Benutzerhandbuch - Deutsch

FUJITSU

WebTransactions V7.5

Konzepte und Funktionen

Ausgabe August 2010

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Bei den meisten IT-Anwendern ist über die Jahre hinweg eine heterogene System- und Anwendungslandschaft entstanden: Mainframes stehen neben Unix- und Windows-Systemen, PCs neben Terminals. Unterschiedliche Hardware, Betriebssysteme, Netze, Datenbanken und Anwendungen werden parallel betrieben. Auf den Mainframe-Systemen und auch auf Unix- oder Windows-Servern existieren oft komplexe und funktional mächtige Anwendungen. Sie sind meist mit erheblichen Investitionen entwickelt worden und stellen in der Regel zentrale Geschäftsprozesse dar, die nicht ohne weiteres durch neue Software ersetzt werden können.

Die Integration vorhandener heterogener Anwendungen in ein einheitliches und transparentes IT-Konzept ist die zentrale Herausforderung der modernen Informationstechnik. Flexibilität, Investitionsschutz und Offenheit für neue Technologien sind dabei von entscheidender Bedeutung.

1.1 Charakterisierung des Produkts

Mit dem Produkt WebTransactions bietet Fujitsu Technology Solutions einen best-of-breed Web-Integration-Server, mit dem eine breite Palette geschäftsrelevanter Anwendungen in kürzester Zeit Browser- und Portal-fähig gemacht werden können. WebTransactions ermöglicht einen schnellen und kostengünstigen Zugang über Standard-PCs und mobile Endgeräte wie Tablett PCs, PDAs (Personal Digital Assistant) und Mobile Phones.

WebTransactions deckt alle Facetten ab, die typischerweise in einem Web-Integrationsprojekt auftreten: von der automatischen Bereitstellung der ursprünglichen „Legacy Oberfläche“ über die grafische Aufbereitung und die Anpassung der Arbeitsabläufe bis hin zu einer umfassenden Frontend-Integration mehrerer Anwendungen. WebTransactions bietet eine hoch-skalierbare Laufzeitumgebung und eine komfortable grafische Entwicklungsumgebung.

Sie können in einer ersten Integrationsstufe folgende Anwendungen und Inhalte über WebTransactions in einer direkten Umsetzung an das WWW anbinden und so Ihren Nutzern intern und extern einfacher zur Verfügung stellen:

– Dialoganwendungen im BS2000/OSD
– MVS- bzw. z/OS-Anwendungen
– systemübergreifende Transaktionsanwendungen auf Basis von openUTM
– dynamische Web-Inhalte

Der Benutzer greift im Internet oder Intranet mit einem Web-Browser seiner Wahl auf die Host-Anwendung zu.

Durch Nutzung modernster Technologie bietet WebTransactions als zweite Integrationsstufe an, die - oftmals noch alphanumerische - Oberfläche der bestehenden Host-Anwendung durch eine attraktive grafische Oberfläche zu ersetzen oder zu ergänzen. Außerdem kann die Host-Anwendung mit WebTransactions auch funktional erweitert werden, ohne dass Eingriffe auf der Host-Seite erforderlich wären (Dialog-Reengineering).

In einer dritten Integrationsstufe können Sie unter der einheitlichen Oberfläche des Browsers unterschiedliche Host-Anwendungen miteinander verknüpfen. Dabei ist es möglich, beliebige vormals heterogene Host-Anwendungen, beispielsweise MVS- oder OSD-Anwendungen miteinander zu verknüpfen oder mit beliebigen dynamischen Web-Inhalten zu kombinieren. Welche Datenquelle ursprünglich die Daten liefert, ist für den Endnutzer nicht mehr sichtbar.

Zusätzlich können Sie den Leistungsumfang und die Funktionalität von WebTransactions-Anwendungen durch eigene Clients beliebig erweitern. Dazu stellt Ihnen WebTransactions ein offenes Protokoll und Schnittstellen (APIs) bereit.

Parallel zum Zugriff über WebTransactions kann weiterhin auch über „herkömmliche“ Terminals oder Clients auf die Host-Anwendungen oder dynamische Web-Inhalte zugegriffen werden. So können Sie eine Host-Anwendung schrittweise ans Web anschließen und die Wünsche und Bedürfnisse unterschiedlicher Nutzergruppen berücksichtigen.

1.2 Liefereinheiten von WebTransactions

Folgende Tabelle gibt einen Überblick über die Liefereinheiten und die Plattformen, für die WebTransactions derzeit verfügbar ist:

Liefereinheit	**WebTransactions-Plattform**	**unterstütztes Protokoll**
WebTransactions for OSD	BS2000/OSD Solaris Linux Windows	9750
WebTransactions for MVS	Solaris Linux Windows	3270
WebTransactions for openUTM	BS2000/OSD Solaris Linux Windows	UPIC

Tabelle 1: Liefereinheiten von WebTransactions

Jede Liefereinheit besteht aus dem WebTransactions-Laufzeitsystem, jeweils mit einem speziellen Host-Adapter, über den die Kommunikation mit der Host-Anwendung läuft, und dem Host-Adapter für dynamische Web-Inhalte. Jede Liefereinheit enthält die Entwicklungsumgebung WebLab, mit der Sie eine Host-Anwendung an das WWW anbinden und die Host-Formate optisch aufbereiten und auch funktional erweitern können.

Durch die modulare Architektur von WebTransactions und die Ausrichtung auf offene Standards sind neue Host-Adapter für zusätzliche Anwendungstypen bei Bedarf auf Anfrage schnell und kostengünstig realisierbar - auch innerhalb von Projektlösungen.

1.2.1 Anwendungs-spezifische Host-Adapter

Die Host-Anwendungen kommunizieren in der Regel über ein proprietäres Protokoll, das den anwendungs-spezifischen Host-Adaptern und damit den einzelnen Liefereinheiten ihren Namen gibt.

WebTransactions for openUTM

WebTransactions for openUTM realisiert die Web-Anbindung von openUTM-Host-Anwendungen, die für FHS oder Formant entwickelt wurden oder über die Client-Server-Schnittstelle UPIC arbeiten. WebTransactions for openUTM steht auf den WebTransactions-Plattformen BS2000/OSD, Solaris, Linux und Windows zur Verfügung.

Unter Solaris, Linux und Windows kann mit beliebigen Web-Servern zusammengearbeitet werden. Unter BS2000/OSD ist für WebTransactions der Web-Server von Apache erforderlich.

Auch die openUTM-Host-Anwendung kann auf jeder dieser Plattformen ablaufen. Der Host-Adapter für die Kommunikation mit der Host-Anwendung basiert auf openUTM-Client (UPIC), d.h., WebTransactions und Host-Anwendung können jeweils auf unterschiedlichen Plattformen ablaufen.

Mit WebTransactions for openUTM wird das BS2000-Programm `IFG2FLD` ausgeliefert, mit dem die Beschreibungen der Host-Formate aus der IFG-Bibliothek in Formatbeschreibungsquellen umgesetzt werden. Aus den Formatbeschreibungsquellen können mit WebLab automatisch Templates generiert werden. Diese generierten Templates bilden die Basis für die individuelle Gestaltung einzelner Formate.

WebTransactions for OSD

WebTransactions for OSD realisiert die Web-Anbindung von beliebigen BS2000/OSD-Dialoganwendungen (TIAM, DCAM). Dabei kann es sich auch um openUTM-Anwendungen handeln.

WebTransactions for OSD ist auf den WebTransactions-Plattformen BS2000/OSD, Solaris, Linux und Windows verfügbar und arbeitet dort jeweils mit beliebigen Web-Servern zusammen. Lediglich unter BS2000/OSD ist für WebTransactions der Web-Server von Apache erforderlich.

Der Host-Adapter, über den die Kommunikation mit der Host-Anwendung läuft, basiert auf einer 9750-Emulation. WebTransactions kann so wie ein 9750-Terminal mit der Host-Anwendung kommunizieren. Mit Hilfe eines so genannten Automask-Templates wird der 9750-Datenstrom 1:1 in HTML umgesetzt. Die gebräuchlichsten Tasten eines 9750-Terminals wie DUE, K- und F-Tasten werden im Browser auf Knöpfe abgebildet, alle weiteren Spezialtasten und die programmierbaren Tasten auf Auswahllisten oder ebenfalls auf Knöpfe. Mit aktuellen Browsern können Sie die Host-Anwendung auch über die Tastatur steuern.

Für eine weitergehende Ausgestaltung der Web-Oberfläche stehen folgende Wege offen:

– Eine globale, auf die gesamte Oberfläche wirkende Aufbereitung kann durch Veränderung des Automask-Templates erreicht werden.

– Eine individuelle Aufbereitung einzelner Formate ist durch ein spezielles Capture-Verfahren möglich. Damit werden „Schnappschüsse“ von einzelnen Formaten aufgezeichnet und in entsprechende Templates umgesetzt. Diese bilden dann die Basis für eine individuelle Aufbereitung.

– Um festgelegte Bereiche der Formate mit einem gleichen Layout zu versehen, z.B. mit einer Knopf-Leiste, kann bei der Generierung des Automask- und der formatspezifischen Templates ein Master-Template zu Grunde gelegt werden.

Die 9750-Emulation ist in WebTransactions for OSD integriert.

Mit WebTransactions for OSD wird das BS2000-Programm `IFG2FLD` ausgeliefert, mit dem die Beschreibungen der Host-Formate aus der IFG-Bibliothek in Formatbeschreibungsquellen umgesetzt werden. Aus den Formatbeschreibungsquellen können mit WebLab automatisch Templates generiert werden. Diese generierten Templates bilden dann die Ausgangsbasis für die individuelle Gestaltung einzelner Formate.

WebTransactions for MVS

WebTransactions for MVS realisiert die Web-Anbindung von beliebigen MVS- bzw. z/OS-Dialoganwendungen.

WebTransactions for MVS ist auf den WebTransactions-Plattformen Solaris, Linux und Windows verfügbar und arbeitet dort jeweils mit beliebigen Web-Servern zusammen.

Der Host-Adapter, über den die Kommunikation mit der Host-Anwendung läuft, basiert auf einer 3270-Emulation. WebTransactions kann so wie ein 3270-Terminal mit der Host-Anwendung kommunizieren. Mit Hilfe eines so genannten Automask-Templates wird der 3270-Datenstrom 1:1 in HTML umgesetzt. Spezielle Tasten eines 3270-Terminals werden im Browser auf Knöpfe bzw. auf eine Liste abgebildet. Mit einem aktuellen Browser können Sie die Host-Anwendung auch über die Tastatur steuern.

Wie bei WebTransactions for OSD stehen für eine weitergehende Ausgestaltung der Web-Oberfläche folgende Wege offen:

– Eine globale, auf die gesamte Oberfläche wirkende Aufbereitung kann durch Veränderung des Automask-Templates erreicht werden.

– Eine individuelle Aufbereitung einzelner Formate ist durch ein spezielles Capture-Verfahren möglich. Damit werden „Schnappschüsse“ von einzelnen Formaten aufgezeichnet und in entsprechende Templates umgesetzt. Diese bilden dann die Basis für eine individuelle Aufbereitung.

– Um festgelegte Bereiche der Formate mit einem gleichen Layout zu versehen, z.B. mit einer Knopf-Leiste, kann bei der Generierung des Automask- und der formatspezifischen Templates ein Master-Template zu Grunde gelegt werden.

Die 3270-Emulation ist in WebTransactions for MVS integriert.

1.2.2 Host-Adapter für dynamische Web-Inhalte

WebTransactions für dynamische Web-Inhalte realisiert den Zugriff auf beliebige Web-Inhalte über das HTTP-Protokoll und wird mit allen Liefereinheiten als zusätzlicher Host-Adapter ausgeliefert.

WebTransactions für dynamische Web-Inhalte ist auf den WebTransactions-Plattformen BS2000/OSD, Solaris, Linux und Windows verfügbar und kann auf beliebige interne oder externe Web-Server über das HTTP-Protokoll zugreifen.

Der Host-Adapter, über den die Kommunikation mit der Web-Anwendung läuft, basiert auf dem HTTP-Protokoll. WebTransactions unterstützt die Verschlüsselung der Nachrichten über HTTPS sowie Client- und Server-Authentifizierung über Zertifikate. WebTransactions kann so, wie beispielsweise ein Browser, mit der Web-Anwendung kommunizieren. Es besteht dann die Möglichkeit, die von der Web-Anwendung empfangenen Daten (HTML oder XML) entweder 1:1 an den Browser weiterzureichen oder sie zu verändern, z.B. nur Ausschnitte zu verwenden, die Daten umzugruppieren oder mit weiteren Daten zu mischen.

WebTransactions für dynamische Web-Inhalte unterstützt den Zugriff auf Web-Services. Durch den Import der Web-Service-Beschreibungen in WSDL (**W**eb **S**ervice **D**escription **L**anguage) wird ein einfacher und transparenter Mechanismus zur Nutzung entfernter Dienste über SOAP (**S**imple **O**bject **A**ccess **P**rotocol) zur Verfügung gestellt.

1.3 Dokumentation zu WebTransactions

Zusätzlich zum vorliegenden Handbuch enthält die Dokumentation zu WebTransactions folgende Einheiten:

- Ein Referenz-Handbuch, das für alle Liefereinheiten gilt und die WebTransactions Template-Sprache WTML beschreibt:

  Template-Sprache

  Nach einem Überblick über WTML finden Sie

  – die lexikalischen Elemente, die in WTML verwendet werden.

  – die klassenunabhängigen globalen Funktionen, wie z.B. `escape()` oder `eval()`.

  – die eingebauten Klassen und Methoden, wie z.B. die Klassen `Array` oder `Boolean`.

  – die WTML-Tags, die die WebTransactions-spezifischen Funktionen enthalten.

  – die WTScript-Anweisungen, die Sie in den WTScript-Bereichen angeben können.

  – die Klassen-Templates, mit denen Sie die Auswertung gleichartiger Objekte automatisieren können.

  – die Master-Templates, die von WebTransactions als Schablone verwendet werden und für ein einheitliches Layout sorgen.

  – eine Beschreibung der Java-Integration, mit der Sie eigene Java-Klassen in WebTransactions instanziieren und der Userexits, mit denen Sie eigene C/C++-Funktionen integrieren können.

  – die mit WebTransactions fertig ausgelieferten UserExits.

  – die XML-Konvertierung für die portable Darstellung von Daten für die Kommunikation mit externen Anwendungen über XML-Nachrichten und die Konvertierung von WTScript-Datenstrukturen in XML-Dokumente.

- Jeweils ein Benutzerhandbuch für jeden Host-Adapter mit speziellen Informationen, zugeschnitten auf den Typ der Partneranwendung:

  Anschluss an openUTM-Anwendungen über UPIC

  Anschluss an OSD-Anwendungen

  Anschluss an MVS-Anwendungen

  Alle Handbücher zu den Host-Adaptern enthalten eine ausführliche Beispielsitzung. Sie beschreiben

  – die Installation von WebTransactions mit dem jeweiligen Host-Adapter.

  – das Einrichten und Starten einer WebTransactions-Anwendung.

  – die Umsetzungs-Templates für die dynamische Umsetzung der Formate auf die Oberfläche eines Web-Browsers.

  – die Bearbeitung von Templates.

  – die Steuerung der Kommunikation zwischen WebTransactions und den Host-Anwendungen über verschiedene Attribute des Systemobjekts.

  – die Behandlung asynchroner Nachrichten und die Druckfunktionen von WebTransactions.

- Ein Benutzerhandbuch, das für alle Liefereinheiten gilt und die Möglichkeiten des HTTP-Host-Adapters beschreibt:

  Zugriff auf dynamische Web-Inhalte

  Das Handbuch beschreibt

  – wie Sie mit WebTransactions auf HTTP-Server zugreifen und deren Ressourcen nutzen.

  – die Einbettung des SOAP-Protokolls (Simple Object Access Protocol) in WebTransactions und den Anschluss von Web-Services über SOAP.

- Ein Benutzerhandbuch, das für alle Liefereinheiten gilt und das offene Protokoll und die Schnittstellen für die Client-Entwicklung für WebTransactions beschreibt:

  Client-APIs für WebTransactions

  Das Handbuch beschreibt

  – das Konzept der Client-Server-Schnittstelle von WebTransactions.

  – die Klasse `WT_RPC` und die Schnittstelle `WT_REMOTE`. Ein Objekt der Klasse `WT_RPC` repräsentiert eine Verbindung zu einer fernen WebTransactions-Anwendung, die auf der Server-Seite über die Schnittstelle `WT_REMOTE` abgewickelt wird.

  – Das Java-Package `com.siemens.webta`, das für die Kommunikation mit WebTransactions ausgeliefert wird.

- Ein Benutzerhandbuch, das für alle Liefereinheiten gilt und das Web-Frontend von WebTransactions beschreibt, das den Zugriff auf allgemeine Web-Services ermöglicht:

  Web-Frontend für Web-Services

  Das Handbuch beschreibt

  – das Konzept des Web-Frontends für objektorientierte Backend-Systeme.

  – die Generierung von Templates für den Anschluss von allgemeinen Web-Services an WebTransactions.

  – den Test und die Weiterentwicklung des Web-Frontends für allgemeine Web-Services.

1.4 Konzept und Zielgruppe dieses Handbuchs

Das vorliegende WebTransactions-Handbuch „Konzepte und Funktionen“ soll als Einstieg in die konkrete Arbeit mit WebTransactions dienen und richtet sich an alle, die mit WebTransactions noch nicht vertraut sind.

In diesem Handbuch stehen nicht die syntaktischen Feinheiten einzelner Anweisungen oder die Schnittstellen-Details im Vordergrund. Vielmehr werden alle grundlegenden Konzepte von WebTransactions erklärt, die für alle Liefereinheiten gleich sind. Wenn Sie darauf aufbauen, werden Sie sich in den übrigen Handbüchern der WebTransactions-Reihe leicht zurechtfinden.

Im Kapitel 2 werden Ihnen zunächst die Eigenschaften und die Funktionsweise von WebTransactions kurz vorgestellt. Im Kapitel 3 finden Sie die für WebTransactions wesentlichen Definitionen, darunter auch eine ausführliche Beschreibung des Objekt-Konzepts. Im Kapitel 4 steht der dynamische Ablauf von WebTransactions-Sitzungen im Mittelpunkt. Kapitel 5 beschreibt, wie Sie WebTransactions administrieren. Im Kapitel 6 wird WebLab, die Entwicklungsumgebung von WebTransactions, vorgestellt und deren wesentliche Funktionalität beschrieben.

Die ausführlichen Verzeichnisse am Ende des Handbuchs - Abkürzungen, Fachwörter, Literatur und Stichwörter - sollen Ihnen den Umgang mit diesem Handbuch erleichtern.

Gültigkeit der Beschreibung

Diese Dokumentation gilt für alle Liefereinheiten - gleichgültig auf welcher WebTransactions-Plattform sie jeweils eingesetzt werden.

1.5 Neue Funktionen

Dieser Abschnitt bietet einen allgemeinen Überblick über wesentliche Neuerungen der WebTransactions-Version 7.5 und verweist auf die genaue Beschreibung.

Art der Neuerung	Beschreibung
Globale Systemobjekt-Attribute: – Neues Systemobjekt-Attribut `LT_REPLACE_STRING` – Neues Systemobjekt-Attribut `PREVENT_EXIT_SESSION`	– Seite 85 – Seite 86
Globale Funktionen: – Neue Funktion `moveFile()` – Neue Funktion `copyFile()` – Neue Funktion `isRequestWaiting()` – Neuer Parameter `all` in `listFolder()`	WebTransactions-Handbuch „Templatesprache“
Eingebaute Klassen und Methoden: – Neue Methode `String.fromCharCode` – Neue Methode `string.charCodeAt` – Neue Methode `WT_Filter.dataObjectToFormattedXML` – Änderung bei der Ausgabe der Methode `toString()` an Objekten und Arrays: bessere Serialisierung/Deserialisierung	WebTransactions-Handbuch „Templatesprache“
Exceptions: Neue Attribute `strLine`, `strColumn` und `strText` am Exception-Objekt	WebTransactions-Handbuch „Templatesprache“
C/C++-Userexits: Neues Argument `SendMail`	WebTransactions-Handbuch „Templatesprache“
WebTransactions for openUTM: – Neues Host-Datenobjekt-Attribut `Unicode` – Neues Attribut `Unicode` an `WT_HOST_MESSAGE`	WebTransactions-Handbuch „Anschluss an openUTM-Anwendungen über UPIC“
WebTransactions for OSD: – Neuer Wert für das Systemobjekt-Attribut `HOST_CHARSET`: `9763-UNICODE` – Neuer Wert für das Systemobjekt-Attribut `TERMINAL_TYPE: UTF-8` – Änderung der Systemobjekt-Attribute `END_MARK` und `LZE_CHAR`	WebTransactions-Handbuch „Anschluss an OSD-Anwendungen“

Art der Neuerung	Beschreibung
WebTransactions für dynamische Web-Inhalte: – Neues Systemobjekt-Attribut COMMUNICATION_FILE_NAME – Neues Systemobjekt-Attribut COMMUNICATION_FILE_TYPE – Neues Systemobjekt-Attribut METHOD	WebTransactions-Handbuch „Zugriff auf dynamische Web-Inhalte“
Web-Frontend für Web-Services: Die Unterstützung von Business Objekten entfällt.	

1.6 Allgemeine Lösungen

Zur Version 7.5 von WebTransactions werden eine Reihe von allgemein verwendbaren Lösungen kostenlos zur Verfügung gestellt. Beispiele dafür sind:

Der Anschluss an das SAP Enterprise Portal
: Ein von SAP zertifiziertes Business Package ermöglicht die nahtlose Integration von WebTransactions-Anwendungen in das mySAP Enterprise Portal.

Der Anschluss an Portale, die JSR168 unterstützen
: Ein Portlet gemäß der JSR168-Spezifikation ermöglicht die nahtlose Integration von WebTransactions-Anwendungen in das entsprechende Portal.

Ein mobiles Portal mit WebTransactions
: Mit einem WTBean können Sie schnell ein personalisierbares Portal zu Ihren Host-Anwendungen erstellen. Sie erhalten eine für PCs geeignete Portaloberfläche und parallel dazu eine für PDA (**P**ersonel **D**igital **A**ssistant, z.B. Pocket LOOX) optimierte Oberfläche für mobile Zugriffe.

Die Anbindung der BS2000-Konsole
: Ein Zugang zur BS2000-Konsole erlaubt die Administration des BS2000 über einen Browser oder über einen PDA (z.B. Pocket LOOX).

Weitere Informationen zu speziellen Lösungen finden Sie auf der WebTransactions-Homepage (*ts.fujitsu.com/products/software/openseas/webtransactions.html*) . Dort stehen Ihnen die Lösungen auch zum Download zur Verfügung.

1.7 Darstellungsmittel

Diese Dokumentation verwendet die folgenden Darstellungsmittel:

Auszeichnung	Bedeutung
`dicktengleiche Schrift`	feste Teile, die genau in dieser Form ein- oder ausgegeben werden, wie z.B. Schlüsselwörter, URLs, Dateinamen
kursive Schrift	variable Teile, für die Sie konkrete Angaben einsetzen müssen
fette Schrift	Zitate, die genauso am Bildschirm oder in der grafischen Oberfläche angezeigt werden, sowie Menübefehle
[ ]	optionale Angaben. Die eckigen Klammern selbst dürfen Sie nicht angeben.
{*alternative1* | *alternative2* }	alternative Angaben. Einen der Ausdrücke innerhalb der geschweiften Klammern müssen Sie auswählen. Die einzelnen Ausdrücke sind durch senkrechte Striche voneinander getrennt. Die geschweiften Klammern selbst dürfen Sie nicht angeben
...	optionale ein oder mehrmalige Wiederholung des vorhergehenden Elements
	wichtige Hinweise und weiterführende Informationen
►	Aufforderungszeichen, wenn Sie etwas tun sollen.
	verweist auf weiterführende Informationen

2 Leistungsüberblick

Dieses Kapitel beschreibt die unterschiedlichen Einsatzmöglichkeiten von WebTransactions – von automatischer 1:1 Umsetzung bis hin zur Integration mehrerer Host-Anwendungen – und die wesentlichen Funktionen von WebTransactions mit ihren Nutzungsmöglichkeiten (siehe hierzu auch Abschnitt „Funktionsumfang von WebTransactions“ auf Seite 29).

2.1 Einsatzmöglichkeiten

Längst hat sich das Internet als entscheidende Netz-Technologie der Datenverarbeitung, der kommerziellen wie der nicht-kommerziellen, etabliert. Die Analysten der führenden Unternehmensberatungen gehen davon aus, dass in wenigen Jahren nur noch diejenigen Unternehmen im Wettbewerb erfolgreich sein werden, die sich bei der Abwicklung ihrer Geschäftsprozesse der Internet-Technologie bedienen. Der Hauptgrund für diese Entwicklung: Die Nutzung der Internet-Technologie verschafft nicht nur den Zugang zum globalen „elektronischen Marktplatz“ Internet mit seiner grafischen Oberfläche World Wide Web, sondern bietet auch die Möglichkeit, mit ein und derselben Technologie unternehmensinterne oder unternehmensübergreifende Prozesse im Rahmen eines Intranets bzw. eines Extranets abzuwickeln.

Heute existiert in den meisten Unternehmen eine heterogene Systemlandschaft: Geschäftsrelevante Anwendungen und Daten liegen verteilt - je nach Bedarf - auf PC-Servern, Abteilungsrechnern oder Mainframes. Standardsoftware und Individuallösungen koexistieren ebenso wie eine Vielzahl von Betriebssystemen und Datenhaltungen.

Für den reibungslosen Ablauf der Geschäftsprozesse müssen alle Anwendungen und Daten überall dort sofort verfügbar sein, wo sie benötigt werden. Außerdem sollen sie - im Sinne einer Weiterentwicklung und Optimierung der Prozesse - beliebig miteinander verknüpft werden können.

Um diese Aufgaben zu lösen, benötigen Sie Software, die auf der vorhandenen Systemlandschaft aufsetzt und die Anwendungen, in denen hohe Investitionen stecken, ohne Änderung im Internet/Intranet zur Verfügung stellt.

Mit WebTransactions erhalten Sie ein etabliertes, in unterschiedlichen Branchen und Szenarien erfolgreich eingesetztes Produkt, das ein mehrstufiges Leistungsangebot aufweist.

Automatische 1:1 Umsetzung (Browserbasierte Emulation)

WebTransactions stellt Konverter für die Umsetzung proprietärer Maskenformate in browserfähige HTML-Dokumente zur Verfügung. Diese automatisch umgesetzten HTML-Seiten entsprechen dem Look & Feel des Terminals.

Statt über Terminals oder PCs mit Terminalemulationen kann über beliebige Web-Browser auf Host-Anwendungen und andere Datenquellen zugegriffen werden, firmenweit im Intranet oder auch weltweit im Internet. Zugangs- und Zugriffschutz-Mechanismen Ihrer Host-Anwendungen stehen selbstverständlich auch beim Web-Zugriff zur Verfügung. Der Web-Zugriff kann auch parallel zum Terminal-Zugriff angeboten werden.

Verteilprobleme hinsichtlich Client-Software treten nicht auf, Web-Browser sind universell verfügbar und nicht an bestimmte Plattformen gebunden.

Nähere Informationen dazu finden Sie in den Benutzerhandbüchern der jeweiligen Host-Adapter.

Gestaltung der Oberfläche (GUIfication)

Aufbauend auf dem Ergebnis der 1:1-Umsetzung können Sie die Web-Oberfläche weiter ausgestalten: Alle HTML- und JavaScript-Möglichkeiten stehen Ihnen offen.

Ihre bewährten Host-Anwendungen können Sie so mit einem modernen Outfit versehen und an das Corporate Design des Unternehmens anpassen. Es besteht die Möglichkeit, Online-Hilfestellungen zu integrieren. Damit werden die Host-Anwendungen auch für neue Nutzer- und Kundengruppen attraktiv, die den Umgang mit dem PC gewöhnt sind und auf die das Layout alphanumerischer Maskensysteme eher kompliziert und altmodisch wirken würde.

Andererseits fühlen sich IT-Experten, die bereits über Jahre hinweg mit den Host-Anwendungen arbeiten und beim Umgang mit IT den Einsatz minimalistischer aber effektiver Mittel gewohnt sind, vom Overhead einer komfortablen, grafische Effekte nutzenden Benutzerführung eher gegängelt. WebTransactions ermöglicht es, unterschiedliche Nutzergruppen desselben Services mit unterschiedlichen Oberflächen zu versorgen.

Nähere Informationen dazu finden Sie im Abschnitt „Templates bearbeiten“ auf Seite 183 und in den Benutzerhandbüchern der jeweiligen Host-Adapter.

Gestaltung der Dialogabläufe (Interface Reengineering)

WebTransactions bietet Ihnen aber nicht nur die Möglichkeit zu einem „Face Lifting“ Ihrer Host-Anwendungen. Sie können auch die Dialogabläufe neu gestalten. Die starre 1:1 Zuordnung zwischen HTML-Seite und Host-Format wird aufgebrochen: Mit WebTransactions können Sie die von den Host-Anwendungen vorgesehenen Dialogstrategien aktiv steuernd verändern: Ein-/Ausgabe-Elemente können ausgefiltert oder hinzugefügt werden, Dialogschritte lassen sich zusammenfassen oder auffächern.

Sie können so die Arbeitsabläufe der Host-Anwendungen benutzerfreundlicher gestalten, die Akzeptanz erhöhen und auch die Effizienz der Online-Dialoge steigern.

Nähere Informationen dazu finden Sie in den Abschnitten „Dialog zwischen WebTransactions und Host-Anwendung“ auf Seite 116 und „Dialog zwischen WebTransactions und Browser“ auf Seite 118.

Anwendungsintegration (Business Process Reengineering)

Die Dynamik des Marktes stellt hohe Anforderungen an die Flexibilität der Geschäftsprozesse. Die IT-Infrastruktur ist mit der Aufgabe konfrontiert, die Änderungen optimal zu unterstützen. Häufig erschwert jedoch die Komplexität über Jahre gewachsener, heterogener IT-Systeme eine flexible und effiziente Anpassung: Bei jeder Änderung muss ein Puzzle wechselseitiger Bezüge und Abhängigkeiten beachtet und behandelt werden. Dies führt zu hohen Kosten aber nur selten zu befriedigenden Ergebnissen. Die Integration vorhandener heterogener Systeme in ein einheitliches und transparentes IT-Konzept ist deshalb eine zentrale Herausforderung der modernen Informationstechnik.

WebTransactions ermöglicht eine solche Integration. Unterschiedliche Host-Anwendungen auf unterschiedlichen Host-Plattformen können am Web-Server in einer WebTransactions-Anwendung integriert und mit einer gemeinsamen Web-Oberfläche ausgestattet werden. Die bestehenden Host-Anwendungen werden so akzeptiert, wie sie sind. Eine Änderung der Anwendungslogik ist nicht erforderlich. Die Host-Anwendungen werden von WebTransactions über ihre bestehenden und bewährten Schnittstellen angesprochen.

Die Integrationsleistung wird an einer klar abgegrenzten und zentralen Stelle erbracht, in den Templates und Konfigurationsdateien von WebTransactions.

Für den „End-Anwender“ am Web-Browser wird in der Regel gar nicht sichtbar, welche Host-Anwendungen die angeforderten Services jeweils erbringen und auf welchen Host-Plattformen sie ablaufen. Er arbeitet mit der WebTransactions-Anwendung, die Komplexität der dahinter liegenden IT-Infrastruktur bleibt verborgen.

Nähere Informationen dazu finden Sie im Abschnitt „WebTransactions-Dialoganwendung starten“ auf Seite 100 und in den Benutzerhandbüchern der jeweiligen Host-Adapter.

Portale

In vielen Unternehmen werden verschiedene Informationen in Portalen bereitgestellt, um sie möglichst unabhängig von Ort und Zeit zugänglich zu machen. Eine zentrale Frage beim Aufbau eines Portals ist, ob alle relevanten Anwendungen und Daten vom Portal aus erreicht werden. Für das mySAP Enterprise Portal, das Portal von Oracle und alle Portale, die JSR168 und WSRP unterstützen, bietet WebTransactions eine Out-of-the-Box-Integration für BS2000/OSD- und z/OS-Anwendungen.

Auf der anderen Seite kann man mit WebTransactions auch ein eigenes Portal bauen. Ein mitgelieferter Oberflächenbaustein liefert rollenspezifische Portaloberflächen. Die im Produkt enthaltene, voll funktionsfähige Beispielinstallation erlaubt es dem Portal-Benutzer, ohne Vorkenntnisse schnell und einfach den personalisierten Zugang zu seinen Anwendungen innerhalb des Portals zu definieren und zu verwenden. Eine kleine Portallösung mit allen wichtigen Zugängen zu Anwendungen und Daten ist dadurch mit geringem Aufwand verfügbar. Darüber hinaus bietet das WebTransactions-Portal eine für PDAs (z.B. den Pocket Loox) optimierte Bedienung und ist voll roaming-fähig.

Hinweise zu WebTransactions-Anwendungen, die für Portale erstellt werden, finden Sie im Abschnitt „Templates für Portaleinsatz gestalten“ auf Seite 191.

Verteilte Anwendungen

XML liefert die Basis für weitere Integrationsmöglichkeiten: Mehrere verteilte WebTransactions-Anwendungen können zu einer übergreifenden Gesamtlösung zusammengeführt werden. Die Daten der beteiligten WebTransactions-Anwendungen werden als XML-Dokumente über das HTTP- oder HTTPS-Protokoll ausgetauscht.

Ein naheliegender Einsatzfall hierfür ist z.B.: Die Filialen eines Unternehmens verarbeiten mit WebTransactions-Filialanwendungen die lokal gewonnenen Daten. Die Datenbestandteile, die filial-übergreifend zentral genutzt werden sollen, können im Rahmen einer Web-Integrationslösung nach definierten Kriterien kombiniert und über eine gemeinsame Web-Oberfläche zugreifbar gemacht werden. Der Einsatz kooperierender WebTransactions-Instanzen sorgt hier für minimale Netzbelastung.

Nähere Informationen dazu finden Sie im WebTransactions-Handbuch „Zugriff auf dynamische Web-Inhalte“.

Einsatz von Client-Programmen

In heterogenen Verfahrenslandschaften stehen neue und altbewährte IT-Lösungen oft nebeneinander. Mit dem neuen Client-Konzept von WebTransactions können neue und alte IT-Lösungen zusammengeführt werden. Das Schlüsselwort dazu lautet `WT_REMOTE`. Über diese offene Schnittstelle können beliebige Programme auf die Ressourcen (Objekte und Methoden) von WebTransactions-Anwendungen zugreifen und deren Funktionalität nutzen. Eine beliebige Anwendung sendet hierzu Anfragen als mehrteilige HTTP-Nachrichten in einem speziellen Format an den Web-Server. Dieser leitet die Nachrichten an die `WT_REMOTE`-Schnittstelle von WebTransactions weiter, wo sie interpretiert und abgearbeitet werden.

Der wesentliche Unterschied zum Browser als Standard-Client einer WebTransactions-Anwendung ist der, dass nicht nur ein Zugriff auf die HTML-Seiten der WebTransactions-Anwendung möglich ist, sondern auch ein direkter Zugriff auf die ferne Anwendung nach Art eines Remote Procedure Call. Damit wird es für Clients erstmals möglich, aktiv auf eine WebTransactions-Anwendung Einfluss zu nehmen und diese von außen zu steuern, bzw. die Funktionalität einer WebTransactions-Anwendung für eigene Zwecke zu verwenden.

Nähere Informationen dazu finden Sie im WebTransactions-Handbuch „Client-APIs für WebTransactions“.

Lastverteilung durch Cluster

Nutzerzahlen und Nutzungsfrequenz nehmen im Internet und vor allem im Intranet stetig zu. Nicht selten entstehen Situationen, in denen ein einzelner, noch so leistungsfähiger Integrations-Server nicht mehr in der Lage ist, allen Endbenutzern einer Web-Integrationslösung die erforderlichen kurzen Antwortzeiten zu garantieren bzw. überlastungsbedingte Systemstillstände zu verhindern.

WebTransactions hat hinsichtlich Performance und Ausfallsicherheit die passende Antwort: Mehrere Integrations-Server können zu einem so genannten Cluster zusammengefasst werden. Diese Integrations-Server müssen nicht notwendigerweise von demselben Typ sein. Alle von WebTransactions unterstützten Integrationsplattformen, d.h. Windows, OSD, Solaris und Linux, können beliebig miteinander kombiniert werden. Der Cluster wird über die Administrations- und Entwicklungsumgebung definiert und eingerichtet.

Ein oder mehrere am Cluster beteiligte Server übernehmen die Rolle des Cluster Controllers, auf denen die Cluster-Definition hinterlegt wird. Die anderen Integrations-Server, auch Cluster Member genannt, erhalten je eine Kopie der WebTransactions-Anwendung bzw. Templates.

Zur Laufzeit verteilt der Controller die Gesamtlast auf die einzelnen Integrations-Server des Clusters gemäß der festgelegten Strategie. Sie können zwischen verschiedenen Verteilstrategien wählen, vom einfachen „Round Robin“ bis hin zu einer an der aktuellen Auslastung der einzelnen Rechner orientierten „Last-Balance“. Für den Endnutzer gibt es - abge-

sehen von kürzeren Antwortzeiten - keinerlei Unterschied zwischen einer einzelnen Server-Konfiguration und dem Einsatz eines Clusters. WebTransactions basierte Integrationslösungen sind somit problemlos skalierbar.

Nähere Informationen dazu finden Sie im Abschnitt „Cluster-Konzept“ auf Seite 153.

Blade Server Unterstützung

Die Blade Server Systeme der Fujitsu Technology Solutions sind in hohem Maße flexibel, skalierbar, servicefreundlich und hochverfügbar. Sie bieten eine hohe Rechenleistung pro Raumeinheit bei erheblich reduziertem Energieverbrauch gegenüber herkömmlichen Servern. Mit der optionalen Deployment Software ist eine automatisierte Multi-Server-Installation möglich.

WebTransactions kann problemlos auf den Blades eines Blade Servers installiert und mit Hilfe eines Images schnell auf vielen Blades verfügbar gemacht werden. Erstellung und Nutzung von WebTransactions-Anwendungen unterscheiden sich hier nicht gegenüber einem normalen Server.

Die Erstellung und Administration eines WebTransactions-Clusters auf einem Blade Server ist so einfach wie eine Einzelinstallation. Ein möglicher Einsatzfall dazu liegt z.B. bei einer Anwendung vor, die mit sehr vielen Clients betrieben wird, so dass eine Lastverteilung auf mehrere Blades notwendig ist.

Nähere Informationen dazu finden Sie im Abschnitt „WebTransactions auf einem Blade Server“ auf Seite 160.

Application Mobility

Mit WebTransactions können Sie auf die Daten Ihrer Host-Anwendung zu jeder Zeit und von jedem Ort aus zugreifen. Mit der Automask für kleine Displays können geschäftskritische Anwendungen auf BS2000/OSD oder z/OS problemlos auch für PDAs zur Verfügung gestellt werden.

Durch die komfortablen Möglichkeiten des Session Roaming ist es zudem möglich, den Zugriff auf eine Sitzung zwischen verschiedenen Client-Geräten zu übergeben oder nach einer vorübergehenden Unterbrechung der Verbindung nahtlos weiterzuarbeiten.

2.2 Funktionsumfang von WebTransactions

Dieser Abschnitt beschreibt die Funktionen, die in allen Liefereinheiten von WebTransactions zur Verfügung stehen.

Entwicklungsumgebung WebLab

Die Entwicklungsumgebung WebLab ist das Werkzeug, mit dem Sie Ihre Host-Anwendung einfach und schnell an das WWW anbinden. Dabei ist es unerheblich, ob es sich bei Ihrem WebTransactions-Server um einen Windows-Rechner, einen Unix-Rechner oder eine BS2000-Anlage handelt, da WebLab WebTransactions-Anwendungen sowohl lokal als auch remote erzeugen und verwalten kann. Die Daten werden über das HTTP- oder HTTPS-Protokoll von dem WebTransactions-Server zu Ihrem PC übertragen und auf demselben Weg zurückgeschrieben.

Auch die individuelle Programmierung und der Test der Templates gestaltet sich mit WebLab komfortabel, da Sie die Templates über das Netz bearbeiten können. WebLab verwaltet parallel zur Editiersitzung eine WebTransactions-Sitzung in einem Web-Browser, damit Sie sich die Auswirkungen Ihrer Änderungen unmittelbar ansehen können.

WebLab bietet Ihnen einen direkten Zugriff auf die Objekte von WebTransactions und erlaubt es, WTML-Tags dialoggestützt in Ihr Template einzufügen. Um Ihnen Ihre Arbeit zu erleichtern, stellt WebLab eine Reihe von Assistenten bereit, mit denen sich Tätigkeiten, die erfahrungsgemäß bei der Integration Ihrer Host-Anwendung in das Intranet/Internet häufig vorkommen, einfach und komfortabel durchführen lassen.

Eine Einführung in die Funktionsweise und die von WebLab bereitgestellte Funktionalität finden Sie im Kapitel „Die Entwicklungsumgebung WebLab“ auf Seite 167 und im Kapitel „Für Neueinsteiger“ in der WebLab-Hilfe. Dort werden auch einige Einsatzszenarien vorgestellt, die Ihnen den Einstieg in die Arbeitsweise von WebLab erleichtern.

WTML-Templates

WTML-Templates (oder kurz Templates) sind der Dreh- und Angelpunkt jeder WebTransactions-Anwendung. Sie werden mit WebLab aus den Formatdefinitionen der Host-Anwendungen automatisch erzeugt und können - wenn gewünscht - mit HTML- und WTML-Sprachmitteln individuell programmiert werden. Zur Laufzeit werden sie von WebTransactions ausgewertet und legen das Layout der im Browser angezeigten HTML-Seiten fest, steuern die WebTransactions-Anwendung und definieren die Kommunikationsschritte mit den Host-Anwendungen.

Bei der Liefereinheit WebTransactions for openUTM wird für jedes Format automatisch ein entsprechendes Template erzeugt. Bei den Liefereinheiten OSD und MVS Systems gibt es ein Automask-Template, das im Standardfall die Darstellung aller Bildschirmformate übernimmt. Aber auch hier gibt es die Möglichkeit, individuelle Templates zu generieren (Capture-Verfahren).

Nähere Informationen dazu finden Sie in Kapitel „Was ist eine WebTransactions-Anwendung“ auf Seite 41 und in den Benutzerhandbüchern der jeweiligen Host-Adapter.

Master-Template als Schablone für die Template-Generierung

Master-Templates werden von WebTransactions bei der Generierung der Automask und der format-spezifischen Templates als Schablone verwendet und sorgen für ein einheitliches Layout. Das Master-Template-Konzept zeigt seine Stärke vor allem bei Host-Anwendungen, bei denen viele Formate einen ähnlichen Aufbau haben: z.B. eine feste Einteilung in Kopfzeile, Arbeitsbereich und Fußzeile.

In solchen Fällen genügt es, den Aufbau einmal im Master-Template festzulegen und dieses Master-Template bei der Generierung sowohl der formatspezifischen Templates als auch des Automask-Templates als Schablone zuzuweisen. Alle generierten Templates erhalten dann automatisch den gewünschten Aufbau.

Weitere Informationen über die Einsatzmöglichkeiten finden Sie im Abschnitt „Master-, Klassen- und Modul-Templates“ auf Seite 55 und im Benutzerhandbuch der jeweiligen Host-Adapter.

Template-Sprache WTML für individuelle Programmierung

Aufsetzend auf den generierten Templates können Sie die Web-Oberfläche grafisch ausgestalten und - wenn gewünscht - auch die Dialogabläufe neu strukturieren.

Für die individuelle Programmierung der Templates steht Ihnen die Template-Sprache WTML (WebTransactions Markup Language) zur Verfügung. WTML besteht aus zwei Teilen, zum einen aus den WTML-Tags, mit denen Sie die für WebTransactions spezifischen Teile in eine HTML-Seite einbetten, und zum anderen aus WTScript, einer server-seitigen Script-Sprache, mit der sich unter anderem Host-Anwendungen steuern und Daten einer Host-Anwendung verarbeiten lassen.

WTML ermöglicht es somit, nicht nur das Erscheinungsbild der Oberfläche, sondern auch die Verarbeitungslogik individuell zu gestalten: Sie können mit WTML den Dialog mit der Host-Anwendung aktiv steuern und auch mehrere Host-Anwendungen unter einer Web-Oberfläche integrieren.

WTML-Tags sind sehr eng an HTML angelehnt. WTScript bietet eine Vielzahl von JavaScript-ähnlichen Sprachmitteln für die Programmierung von server-seitigen Verarbeitungsschritten. Dies bedeutet, dass Sie für die Template-Logik ohne Sprachbruch die gleichen Sprachmittel verwenden können, wie für die Programmierung der client-seitigen Präsentationslogik.

HTML und client-seitiges JavaScript sind nicht Bestandteil von WTML, sondern werden unverändert an den Browser durchgereicht. WebTransactions ist somit offen für künftige HTML- und JavaScript-Erweiterungen.

Die Template-Technik ist so flexibel, dass Sie damit sogar beliebige dynamische HTML-Strukturen auch ohne dahinter liegende Host-Anwendung realisieren können.

Nähere Informationen dazu finden Sie im Abschnitt „WTML-Templates“ auf Seite 51 und im WebTransactions-Handbuch „Template-Sprache“.

Unterstützung von XML

WebTransactions bietet verschiedene, komfortable Möglichkeiten, XML-Daten zu analysieren. Damit können Sie Ressourcen, die in XML vorliegen, einfach und schnell in Ihre WebTransactions-Anwendung integrieren.

Daneben kann auch aus internen Datenstrukturen XML erzeugt werden. Damit können Sie beispielsweise interne Zustände speichern und beim erneuten Starten einer WebTransactions-Sitzung wieder einlesen.

Unterstützung von Unicode

Die anwendungsspezifischen Host-Adapter WebTransactions for OSD und WebTransactions for openUTM unterstützen Unicode. Sie erzeugen Daten in UTF-8-Codierung, die zum Browser und zurück durchgereicht werden.

Die folgenden Komponenten bzw. anwendungsspezifischen Host-Adapter von WebTransactions unterstützen Unicode nicht:

– der WebTransactions-Kern

  Der Interpreter für die Template-Sprache WTML (WebTransactions Markup Language und WTScript) unterstützt Unicode nicht:
  – WTScript-Templates werden weiterhin als ISO-8859-Zeichen interpretiert.
  – In Zeichenketten wird weiterhin jedes Byte als ein Zeichen interpretiert.

– WebTransactions for MVS

– WebLab (Unicode-Zeichen können nicht dargestellt werden)

Weitere Informationen zur Unicode-Unterstützung finden Sie in den WebTransactions-Handbüchern „Anschluss an OSD-Anwendungen“ und „Anschluss an openUTM-Anwendungen über UPIC“.

WTBeans als wiederverwendbare Komponenten

Für die Programmierung der Templates stellt WebTransactions wiederverwendbare Komponenten, die WTBeans, bereit. WTBeans entsprechen einem ganzen oder einem Teil eines WTML-Dokuments und werden deswegen in standalone und inline WTBeans unterschieden. WTBeans bieten sowohl Unterstützung für die Darstellung der Daten im Browser als auch für die Kommunikation mit den Host-Anwendungen.

Jedes WTBean enthält eine Beschreibung seiner Eigenschaften. Aus diesen Eigenschaften erzeugt WebLab eine grafische Oberfläche, über die Sie die Eigenschaften des WTBean bearbeiten können.

Weitere Informationen über die Einsatzmöglichkeiten finden Sie im Abschnitt „WTBeans“ auf Seite 57.

Unterstützung verschiedener Oberflächen-Stile und Landessprachen

Einer Dialogoberfläche können verschiedene Template-Folgen (Oberflächenstile) zugeordnet werden, zwischen denen am Browser beliebig gewechselt werden kann. So kann eine Host-Anwendung z.B. weiterhin in ihrer bisherigen Darstellung für daran gewöhnte Bearbeiter angeboten werden und parallel hierzu auch in grafisch aufwändiger gestalteten Varianten. Neben verschiedenen Oberflächenstilen ist auch der parallele Einsatz von unterschiedlichen Landessprachen möglich.

Alle Oberflächen-Varianten werden zentral auf dem WebTransactions-Rechner verwaltet und sind nach Änderungen sofort wieder für alle Clients (Web-Browser) verfügbar.

Nähere Informationen dazu finden Sie im Abschnitt „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61.

Einfaches Konzept zur Integration mehrerer Host-Anwendungen

Mit dem Konzept der Kommunikationsobjekte bietet WebTransactions eine einfache und übersichtliche Möglichkeit zur Integration unterschiedlicher Host-Anwendungen unter einer gemeinsamen Web-Oberfläche. Für jede Host-Verbindung wird ein separates Kommunikationsobjekt erzeugt. Alle verbindungsspezifischen Daten - gleichgültig ob Steuerungsinformationen oder ausgetauschte Benutzerdaten - werden in diesem Objekt abgelegt. So ist selbst das Handling von mehreren parallel offenen Verbindungen zu unterschiedlichen Hosts für den Programmierer leicht überschaubar. Für den Endnutzer am Web-Browser bleibt dies meist völlig verborgen, d.h. er merkt gar nicht, dass er mit unterschiedlichen Hosts arbeitet.

Nähere Informationen dazu finden Sie im Abschnitt „Host-Kommunikationsobjekt wt_Host.Comobj - Verwalten einer Host-Anbindung“ auf Seite 93 und in den Benutzerhandbüchern der jeweiligen Host-Adapter.

Sicherheits-Funktionen

Die Sicherheits-Mechanismen der Host-Anwendungen (Zugangsschutz, Zugriffschutz, Restart) stehen selbstverständlich auch beim Zugriff über WebTransactions zur Verfügung. Darüber hinaus können Sie den Zugang zu den WebTransactions-CGI-Programmen durch die Schutzmechanismen der Web-Server-Software oder die Zugriffsrechte des Dateisystems einschränken.

Zusätzlich bietet WebTransactions ein eigenes Sicherheits- und Benutzerkonzept, um den Zugang zu den WebTransactions-Anwendungen für Entwickler und Administratoren zu reglementieren. Editier- und Administrationsberechtigungen können eigenen Kennungen zugewiesen werden.

Das Administrationsprogramm ist eine selbstständige WebTransactions-Anwendung und eng mit WebLab verzahnt, um die nötige Zugriffskontrolle zu gewährleisten.

Nähere Informationen hierzu finden Sie im Kapitel „WebTransactions-Server“ auf Seite 141.

Sitzungsmanagement

WebTransactions übernimmt die Verwaltung der laufenden Sitzungen. Es hält fest, von welchem Client die Anfrage kommt und mit welcher Host-Anwendung der Benutzer arbeiten will.

Mit dem Administrationsprogramm von WebTransactions können Sie sich z.B. einen Überblick über die aktuell laufenden Sitzungen verschaffen, die temporären Dateien einzelner Sitzungen (z.B. Trace-Dateien) anzeigen lassen, Sitzungen beenden oder die gesamte WebTransactions-Anwendung sperren bzw. entsperren. Die Administration ist denkbar einfach und erfolgt über eine eigene WebTransactions-Anwendung.

Nähere Informationen zur Administration der WebTransactions-Anwendungen finden Sie im Abschnitt „WebTransactions-Anwendung administrieren“ auf Seite 137.

Java-Integration

Mit Hilfe der Java-Integration von WebTransactions können Sie den Funktionsumfang Ihrer WebTransactions-Anwendung erweitern. In Verbindung mit WTScript, der an JavaScript angelehnten Script-Sprache von WebTransactions, erschließt Ihnen die Java-Integration folgende Möglichkeiten:

- Java-Objekte und somit auch Java-Arrays erzeugen und benutzen
- Java-Methoden aufrufen
- Attribute von Java-Objekten lesen und ändern
- WTScript-Operatoren auf Java-Objekte anwenden
- An Java angelehnte Ausnahmenbehandlung (Exception Handling)

Damit Java-Programme oder -Anwendungen auf dem WebTransactions-Rechner ablaufen können, muss auf diesem Rechner eine Java-Laufzeitumgebung (Java Virtual Machine, JVM) verfügbar sein. Für die einzelnen WebTransactions-Integrationsplattformen (Windows, Solaris, Linux, BS2000/OSD) stehen jeweils spezielle Java-Laufzeitumgebungen zur Verfügung.

Nähere Informationen hierzu finden Sie im WebTransactions-Handbuch „Template-Sprache“.

Anschluss von Web-Services über das SOAP-Protokoll

WebTransactions ermöglicht den Anschluss an Web-Services via **S**imple **O**bject **A**ccess **P**rotocol (SOAP). Die Einbettung des SOAP-Protokolls in WebTransactions basiert auf dem WebTransactions-Host-Adapter HTTP sowie den Definitionen der Protokolle SOAP und WSDL (**W**eb **S**ervices **D**escription **L**anguage). WSDL bietet XML-Sprachregeln für die Beschreibung der Fähigkeiten von Web-Services.

Das XML-basierte SOAP-Protocol realisiert einen einfachen und transparenten Mechanismus, mit dem strukturierte und typisierte Informationen zwischen Rechnern in einer dezentralisierten, verteilten Umgebung ausgetauscht werden können.

SOAP stellt ein modulares Paketmodell sowie Mechanismen zum Verschlüsseln von Daten innerhalb von Modulen zur Verfügung. Dies ermöglicht die unkomplizierte Beschreibung der externen Schnittstellen einer im Web erreichbaren fernen Anwendung (Web-Service).

Nähere Informationen hierzu finden Sie im WebTransactions-Handbuch „Zugriff auf dynamische Web-Inhalte“.

Einsatz selbst-definierter C/C++-Routinen (Userexits)

Selbst-definierte C/C++-Routinen können Sie als Userexits in WebTransactions-Anwendungen integrieren und so die Funktionalität der Host-Anwendungen ergänzen. Dabei ist es auch möglich, mehrere Userexit-Bibliotheken einzusetzen.

Mit Userexits können Sie sich zusätzliche Schnittstellen zum Betriebssystem oder eigenen Anwendungen erschließen (z.B. Dateiverarbeitung oder eigene Anwendungen, für die kein Host-Adapter existiert).

Nähere Informationen hierzu finden Sie im WebTransactions-Handbuch „Template-Sprache“.

Nutzung von LDAP-Verzeichnisdiensten

Für die Unterstützung des Internet-Protokolls für Directory Services LDAP (**L**ightweight **D**irectory **A**ccess **P**rotocol) ist in WTScript die Klasse `WT_LdapConnection` definiert. Über die Methoden dieser Klasse können Sie LDAP-Verzeichnisdienste nutzen, wie z.B.

- Einträge mit Hilfe von benutzerspezifischen Suchkriterien suchen
- Eintrag hinzufügen
- Eintrag löschen
- Eintrag ändern
- Einträge vergleichen

Darüber hinaus enthält die Klasse `WT_LdapConnection` Methoden zum Einrichten und beenden einer LDAP-Sitzung.

Nähere Informationen hierzu finden Sie im WebTransactions-Handbuch „Template-Sprache“.

Zugriff auf dynamische Web-Inhalte

Über den HTTP-Host-Adapter können Sie aus einem Template auf den Inhalt beliebiger im WWW vorhandener Ressourcen zugreifen. HTML- und XML-Ressourcen können Sie dann mit WTScript analysieren.

Nähere Informationen hierzu finden Sie im WebTransactions-Handbuch „Zugriff auf dynamische Web-Inhalte“.

Web-Frontend für Web-Services

WebTransactions realisiert eine Oberfläche für den Zugriff auf allgemeine Web-Services .

Weitere Informationen dazu finden Sie im WebTransactions-Handbuch „Web-Frontend für Web-Services“.

Zugriff von beliebigen Clients auf die Funktionalität von WebTransactions

Normalerweise werden mit WebTransactions Host-Anwendungen auf Dialog-Anwendungen für das WWW umgesetzt, die in einem beliebigen Browser laufen. Die Client-Schnittstelle `WT_REMOTE` ermöglicht es, von beliebigen Clients auf die Resourcen einer WebTransactions-Anwendung zuzugreifen. Für Java-Clients wird diese Funktionalität in einer eigenen Klassenbibliothek zur Verfügung gestellt.

Nähere Informationen hierzu finden Sie im WebTransactions-Handbuch „Client-APIs für WebTransactions“.

2.3 Komponenten von WebTransactions

Folgende Abbildung gibt einen Überblick über die Architektur von WebTransactions. Die integralen Bestandteile der WebTransactions-Liefereinheiten sind durch graue Hinterlegung gekennzeichnet.

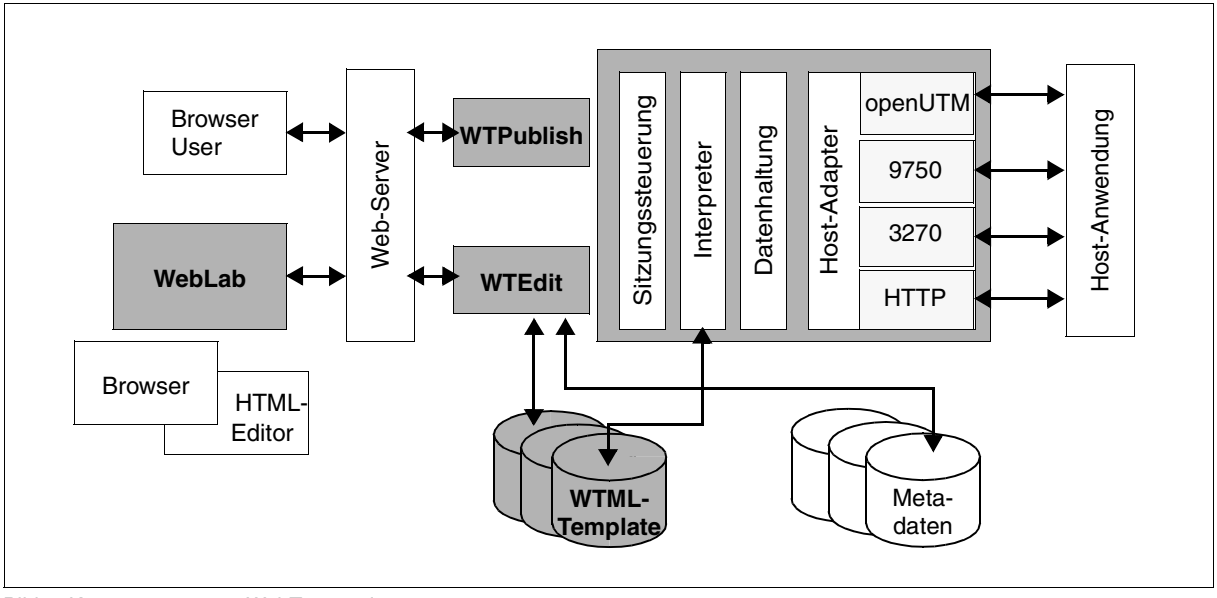


Bild 1: Komponenten von WebTransactions

Basis für die Übertragung von Seiten und Daten im Web ist das Protokoll HTTP (Hypertext Transfer Protocol) bzw. HTTPS (Hypertext Transfer Protocol Secure). Diese standardisierten Protokolle werden für alle Zugriffe auf WebTransactions verwendet, gleichgültig ob es sich um Endbenutzer-Zugriffe oder Administrations-Zugriffe handelt, für die Web-Browser genutzt werden, oder ob der Zugriff über das WebTransactions-Entwicklungssystem WebLab erfolgt.

Mit dem Web-Server kommuniziert WebTransactions über das Common Gateway Interface (CGI), einer normierten Schnittstelle für den Programmaufruf auf einem Web-Server, oder über das Microsoft Internet Server Application Program Interface (ISAPI).

WTPublish, WTEdit

Die Kommunikationsschnittstelle zum Web-Server bilden folgende WebTransactions-Komponenten:
– `WTPublish.exe/WTPublishISAPI.dll` für den „normalen“ Zugriff von End-Benutzern
– `WTEdit.exe` für Editier-Zugriffe

Diese Aufteilung bietet die Möglichkeit für jeweils individuelle Schutzmaßnahmen.

Kern-Komponenten

Den Kern von WebTransactions bilden folgende Komponenten, die für alle Liefereinheiten identisch sind:
– Sitzungssteuerung
– Interpreter für die Template-Sprache WTML (WebTransactions Markup Language und WTScript)
– Datenhaltung für die dynamischen Anwenderdaten

Host-Adapter

Für die Kommunikation mit den Host-Anwendungen stellen die unterschiedlichen Liefereinheiten spezielle Host-Adapter (Kommunikationsmodule) zur Verfügung, die dafür sorgen, dass die Schnittstellen der Host-Anwendungen unverändert genutzt werden können. Durch diese Host-Adapter-Technik sind die Spezifika unterschiedlicher Protokolle für die anderen Komponenten von WebTransactions transparent.

Der Host-Adapter für dynamische Web-Inhalte wird mit jeder Liefereinheit ausgeliefert, damit auch einzelne Host-Anwendungen mit dynamischen Web-Inhalten kombiniert werden können.

Modulare, offene Architektur - neue Host-Adapter

Durch die modulare Architektur von WebTransactions und die Ausrichtung auf offene Standards sind neue Host-Adapter (Kommunikationsmodule) für zusätzliche Host-Anwendungstypen bei Bedarf auf Anfrage schnell und kostengünstig realisierbar - auch innerhalb von Projektlösungen, ohne dass die übrigen Komponenten davon betroffen wären.

Programm IFG2FLD

Mit den Liefereinheiten WebTransactions for OSD und WebTransactions for openUTM wird das Programm `IFG2FLD` ausgeliefert, das die Konvertierung von FHS-Masken vorbereitet. Es erzeugt eine Formatbeschreibungsquelle, die beim Generieren der Templates als Eingabe für WebLab dient.

3 Was ist eine WebTransactions-Anwendung

Eine WebTransactions-Anwendung ist das Bindeglied zwischen einer Host-Anwendung und dem WWW. Sie sorgt dafür, dass die von der Host-Anwendung empfangenen und gesendeten Daten in eine Form gebracht werden, die ein Web-Browser verarbeiten und darstellen kann. Dabei kann es sich um unterschiedliche Host-Anwendungen handeln, die auf verschiedenen Host-Plattformen liegen können.

Eine WebTransactions-Anwendung stellt für die Endnutzer am Browser (ultra thin client) eine Dialogoberfläche zur Verfügung. Für jeden verbundenen Client verwaltet WebTransactions einen Sitzungskontext auf dem Server, sodass ohne zusätzliche Programmierung im zustandslosen Web Sitzungen und damit Dialoge möglich sind. Als Datenquelle und Empfänger können ein oder mehrere eigenständige Host-Anwendungen an die WebTransactions-Anwendung angeschlossen sein.

Die Kommunikation zwischen WebTransactions und der Host-Anwendung erledigt ein so genannter Host-Adapter, der das Protokoll der jeweiligen Host-Anwendung bedient. Entsprechend dem Protokoll, über das der Host-Adapter mit der Host-Anwendung kommuniziert, ist die jeweilige Liefereinheit von WebTransactions benannt.

Der Rechner, auf dem WebTransactions läuft, wird auch WebTransactions-Server oder Integrations-Server genannt. Host-Anwendung und WebTransactions können auf demselben Rechner ablaufen, dies ist aber in der Regel nicht der Fall. Alle Schritte zur Integration Ihrer Host-Anwendungen führen Sie mit WebLab lokal oder remote auf dem WebTransactions-Server aus.

3.1 Bestandteile einer WebTransactions-Anwendung

Eine WebTransactions-Anwendung braucht neben der protokollspezifischen Liefereinheit von WebTransactions ein Basisverzeichnis. In diesem Basisverzeichnis werden alle Dateien Ihrer Anwendung abgelegt, unter anderem:

- Start-Template
- Automask-Template oder formatspezifische WTML-Templates, die die Umsetzung zwischen Host-Anwendung und Browser steuern
- protokollspezifische Konfigurationsdateien

Das Basisverzeichnis muss sich immer auf dem Rechner befinden, auf dem auch WebTransactions abläuft. Sie erstellen das Basisverzeichnis und die erforderlichen Templates mit der Entwicklungsumgebung WebLab, die im Kapitel „Die Entwicklungsumgebung WebLab“ auf Seite 167 beschrieben ist.

Sie können alle Dateien, die bei der Darstellung Ihrer Templates direkt vom Web-Server erreichbar sein müssen, im Basisverzeichnis ablegen. Dazu wird bei der Generierung des Basisverzeichnisses das Unterverzeichnis `wwwdocs` miterzeugt. Dort können Sie Bilder, clientseitige Scripts, etc. ablegen, die vom Browser aus direkt erreichbar sein müssen, siehe hierzu auch Abschnitt „Unterverzeichnis wwwdocs“ auf Seite 65.

Der Aufbau des Basisverzeichnisses ist abhängig von der WebTransactions-Liefereinheit und ist im Abschnitt „Struktur des Basisverzeichnisses“ auf Seite 60 beschrieben.

Um eine Host-Anwendung an das WWW anzuschließen, brauchen Sie neben der Liefereinheit von WebTransactions:

– einen Rechner, auf dem ein Web-Server läuft, als zukünftigen Integrations-Server

– beliebig viele Rechner, auf denen ein Browser läuft, für den Zugriff auf die WebTransactions-Anwendung

Eine WebTransactions-Anwendung kann deswegen auf einem oder mehreren Rechnern mit verschiedenen Betriebssystemen ablaufen: z.B. der Browser als Client auf einem Windows-Rechner, der WebTransactions-Server auf einem Unix-Rechner und die angeschlossene Host-Anwendung auf einem OSD- oder MVS-Mainframe.

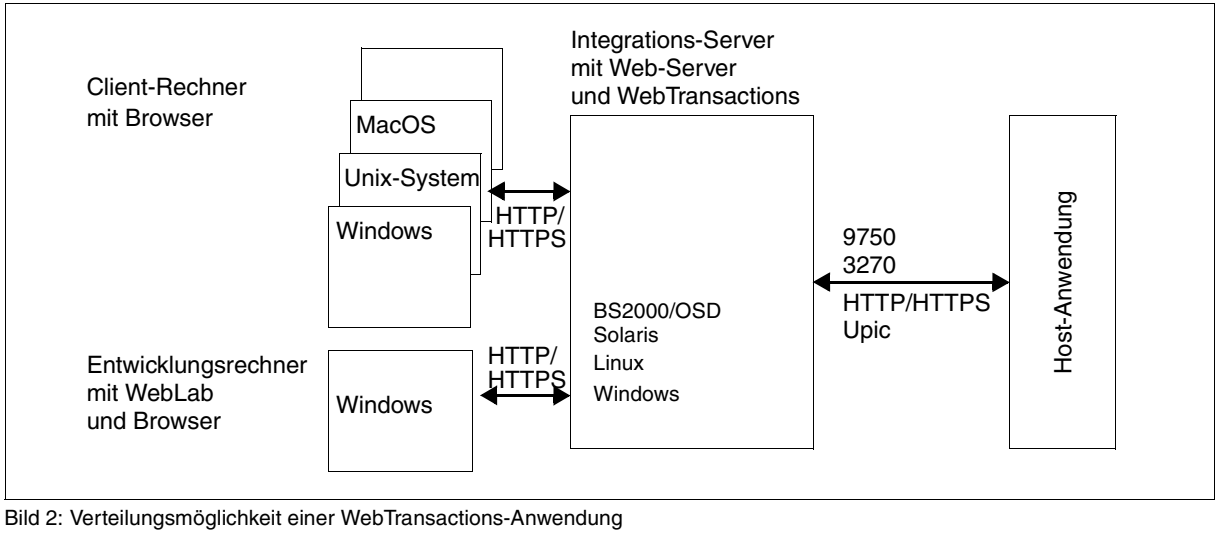


Bild 2: Verteilungsmöglichkeit einer WebTransactions-Anwendung

3.2 WebTransactions-Sitzung

Die Zeitspanne, die ein Benutzer mit einer WebTransactions-Anwendung arbeitet, wird WebTransactions-Sitzung genannt. Zu Beginn jeder solchen Sitzung startet WebTransactions einen Prozess, in dem das WTHolder-Programm läuft. Dieser Prozess bleibt während der gesamten Sitzungszeit bestehen. Die Zuordnung zwischen WTHolder-Programm und dem Browser des Benutzers verwaltet das CGI-Programm `WTPublish`. Die Anzahl der aktiven WTHolder-Prozesse entspricht also der Anzahl der gleichzeitig aktiven Benutzer.

Im Lauf einer WebTransactions-Sitzung können eine oder auch mehrere Verbindungen zu Host-Anwendungen geöffnet und wieder geschlossen werden. Werden in einer WebTransactions-Sitzung mehrere Host-Verbindungen genutzt, so kann dies entweder hintereinander oder auch parallel erfolgen.

Wie Sie eine WebTransactions-Sitzung starten bzw. beenden, ist den Abschnitten „Start-Möglichkeiten“ auf Seite 101 bzw. „Sitzung beenden“ auf Seite 125 beschrieben.

3.2.1 Roaming Sessions

Eine Roaming Session ist eine WebTransactions-Sitzung, die nacheinander oder gleichzeitig von verschiedenen Client-Geräten aus angesprochen werden kann. So kann ein Benutzer z.B. eine Sitzung, die er am Arbeitsplatz begonnen hat, von einem mobilen Gerät aus fortsetzen. Der Weiterbetrieb ist mit diesem Mechanismus auch beim Absturz eines Client-Geräts gewährleistet. Eine Roaming Session unterstützt also einen Wiederanlauf vom Server aus.

Mit WebTransactions kann ein Benutzer beim Starten einer Sitzung festlegen, ob die Sitzung auch von anderen Client-Geräten aus angesprochen werden kann. Dazu ist eine benutzereigene Session Id nötig.

Roaming Sessions können auch in einem Cluster (siehe Abschnitt „Cluster-Konzept“ auf Seite 153) verwendet werden.

WTBean wtcRoaming

Für die Unterstützung von Roaming Sessions wird das WTBean `wtcRoaming` mit ausgeliefert. Dieses WTBean erzeugt ein Start-Template für eine Roaming Session und speichert am Sitzungsbeginn Daten für die Authentifizierung des Benutzers. Wird die Verbindung zu einer Roaming Session erneut hergestellt, so erwartet das WTBean die gleichen Authentifizierungsdaten wie zu Beginn der Sitzung.

Das WTBean `wtcRoaming` ist in der Online-Hilfe von WebLab beschrieben. Dort finden Sie eine Anleitung, wie Sie mit `wtcRoaming` ein Start-Template für eine Roaming Session erzeugen und den Wiederanlauf testen.

WTBean wtcSingleSignOn

Das WTBean `wtcSingleSignOn` stellt die Funktion des Single Sign-On zur Verfügung. Sie müssen sich nur mit Benutzerkennung und Passwort bei WebTransactions anmelden und können die anschließenden Vorgänge mit der entsprechenden Authentisierung starten.

Das WTBean `wtcSingleSignOn` bietet eine besonders komfortable Unterstützung für die Roaming Sessions durch das enthaltene Benutzerkonzept und die Identitätsprüfung.

Sie können das WTBean `wtcSingleSignOn` vom Download-Bereich der WebTransactions-Homepage unter dem Stichpunkt Ready-to-run herunterladen. WTBeans müssen eigens auf dem Rechner, auf dem WebLab läuft, installiert werden.

Für den Regelfall wird empfohlen, zum Erstellen des Start-Templates und die Überprüfung der Authentifizierungsdaten die WTBeans `wtcRoaming` oder `wtcSingleSignOn` zu verwenden. Im folgenden Text ist beschrieben, was Sie beachten müssen, wenn Sie auf den Einsatz der WTBeans verzichten wollen.

Roaming Session starten

Um eine Roaming Session zu starten, die von mehreren Client-Geräten aus angesprochen werden kann, muss der Wert des Attributs `WT_SYSTEM_SESSION`, der die Session Id enthält, mit dem Präfix `R` beginnen (zum Starten einer Sitzung siehe Abschnitt „WebTransactions-Dialoganwendung starten“ auf Seite 100).

Beispiel

```
http://myhost/cgi-bin/WTPublish.exe/startup?
WT_SYSTEM_BASEDIR=C:/Base/App&WT_SYSTEM_FORMAT=start&
WT_SYSTEM_SESSION=R-MyRoamingId
```

Die Session Id für die Roaming Session ist hier `R-MyRoamingId`.

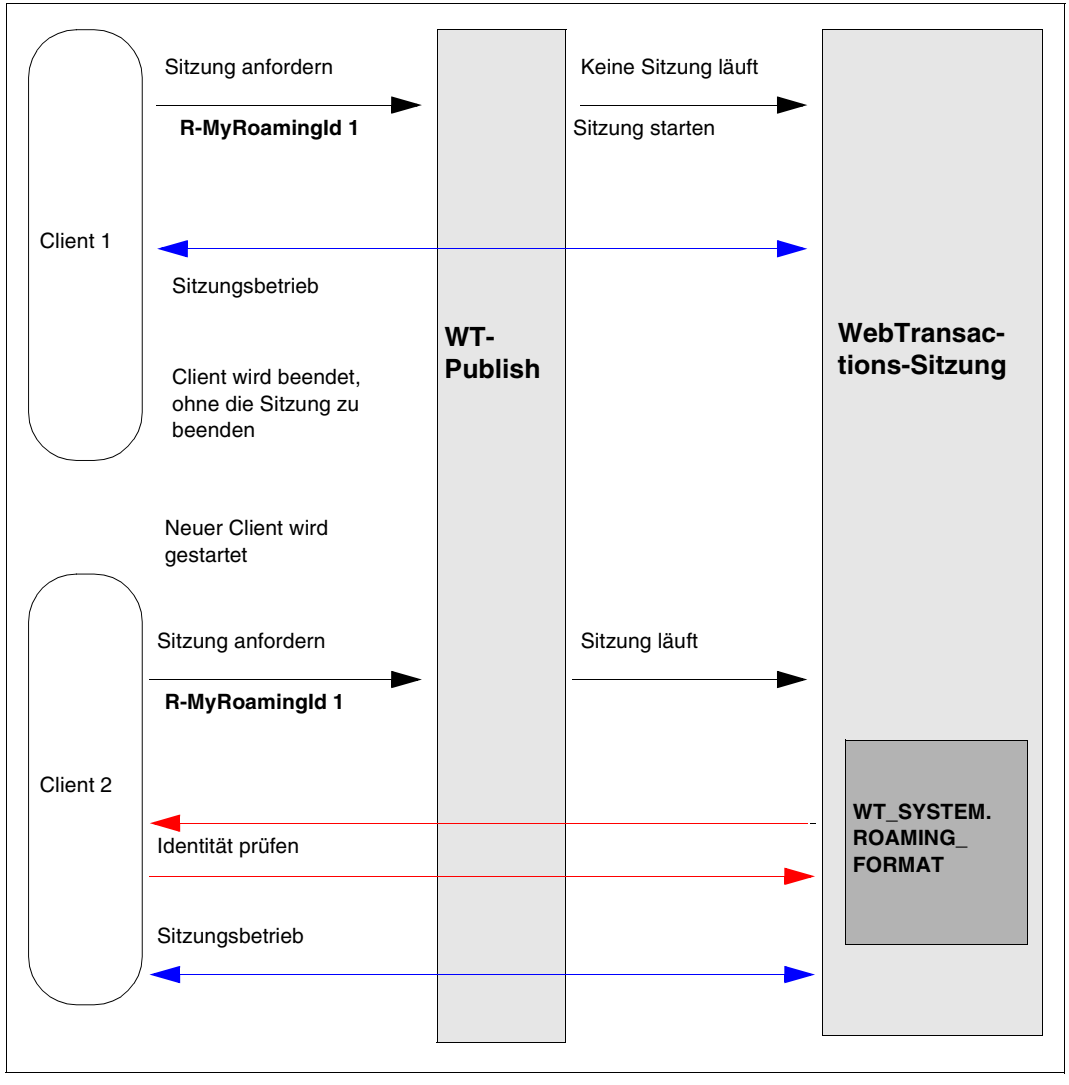
Die Sitzung wird mit dem angegebenen Start-Template gestartet.

Läuft bereits eine Sitzung mit der angegebenen Session Id, wird sie mit der zuletzt synchron erzeugten Seite fortgesetzt.

Damit es nicht zu unberechtigten Zugriffen auf die Sitzung kommen kann, muss in diesem Fall die Authentizität des Benutzers überprüft werden. Hierfür wird das Attribut `WT_SYSTEM.ROAMING_FORMAT` eingesetzt, das mit dem Namen eines Templates versorgt sein muss. In diesem Template muss der berechtigte Zugriff auf die Sitzung kontrolliert werden. Was Sie beim Implementieren eines solchen Templates beachten müssen, ist im Abschnitt „Identität prüfen“ auf Seite 47 beschrieben.

Auf Windows-Plattformen darf jede Session Id einer Roaming Session nur einmal verwendet werden, das heißt, dass zwei Sitzungen nicht mit der gleichen Session Id betrieben werden, auch wenn die Sitzungen in unterschiedlichen Basisverzeichnissen laufen.

Die folgende Abbildung zeigt schematisch den Vorgang beim Starten einer Roaming Session:



Identität prüfen

Bei der Wiederaufnahme einer laufenden Sitzung muss die Authentizität des Benutzers erneut überprüft werden, da es sonst zu unberechtigten Zugriffen auf die Sitzung kommen kann. Hierfür können entweder die mitgeschickten Daten des Requests ausgewertet oder z.B. eine Authentifizierungs-Seite geschickt werden.

Beim Wieder-Eintritt in eine Sitzung setzt WebTransactions das Attribut `WT_SYSTEM.ROAMING` auf `true`, damit auch in weiteren Templates der Status zuverlässig geprüft werden kann. Damit lässt sich in einem Template unterscheiden, ob es sich um eine Neuanmeldung oder die Wiederaufnahme einer laufenden Sitzung handelt. Es wird so möglich, zur erneuten Authentifizierung wiederum das Start-Template zu verwenden und anhand des Werts von `WT_SYSTEM.ROAMING` bei einer Wiederaufnahme z.B. verschiedene Initialisierungen entfallen zu lassen.

Wird der Zugriff abgewiesen, so ist Folgendes zu beachten:

– Das Attribut `WT_SYSTEM.SIGNATURE` darf nicht an den Browser weitergegeben werden. Andernfalls wäre ein unberechtigter Zugriff auf die Sitzung möglich.

– Für den Aufruf eines weiteren Formulars sollte das Tag `wtDataform` verwendet werden, da mit `wtDataform` die zugehörige URL automatisch generiert wird.

– Die Attribute `HREF` und `HREF_ASYNC` dürfen nicht verwendet werden.

Verläuft die Prüfung der Identität positiv, muss in der Regel die zuletzt synchron erzeugte Seite der WebTransactions-Anwendung ausgegeben werden.

Beispiel

```
<script>
  document.location = '##WT_SYSTEM.HREF_ASYNC#';
</script>
```

Die Ausgabe der zuletzt synchron erzeugten Seite ist nicht sinnvoll, wenn z.B. auf der Client-Seite ein anderes Gerät eingesetzt wird. In diesem Fall muss die Seite neu generiert werden. Dabei kann gegebenenfalls die Ausführung der `wtOnReceive`-Scripts unterdrückt werden. Das Template zur erneuten Prüfung der Authentizität könnte z.B. den Stil entsprechend des aktuell verwendeten Client-Geräts umschalten. Eine erneute Generierung im veränderten Stil lässt sich dann durch Ausgabe des folgenden Scripts herbeiführen.

Beispiel

Das Mitschicken von `WT_DISPOSE_RECEIVE_SCRIPTS=true` im folgenden Script unterdrückt die Ausführung aller `wtOnReceive`-Scripts.

```
<script>
  document.location = '##WT_SYSTEM.HREF#&WT_DISPOSE_RECEIVE_SCRIPTS=true';
</script>
```

Roaming Sessions in WebLab

Jede Sitzung, die mit Hilfe von WebLab gestartet wurde (siehe Abschnitt „Sitzung starten“ auf Seite 174), ist zugleich eine Roaming Session.

WebLab bietet Ihnen die Möglichkeit, den Wiederanlauf einer Roaming Session zu testen. Sie verwenden dazu den Befehl **Steuerung/Roaming testen**. Mit diesem Befehl wird das Dialogfeld **Roaming testen** angezeigt. Hier können die zusätzlichen Startparameter verändert werden, um z.B. falsche Anmelde-Informationen zu testen.

Das Dialogfeld **Roaming testen** ist in der Online-Hilfe von WebLab beschrieben.

Roaming Sessions in einem Cluster

Roaming Sessions können auch in einem WebTransactions-Cluster (siehe Abschnitt „Cluster-Konzept“ auf Seite 153) verwendet werden.

Damit eine Roaming Session innerhalb eines Clusters identifiziert werden kann, muss die Session Id beim Start einer Cluster-Sitzung mit übergeben werden (siehe Abschnitt „Cluster-Sitzung starten“ auf Seite 158).

Beispiel

```
http://my-server/cgi-bin/WTCluster.exe/my-cluster-id?
WT_SYSTEM.SESSION=R-MyRoamingId
```

3.2.2 Service-Anwendungen

Eine Service-Anwendung ist eine WebTransactions-Sitzung, die abwechselnd von verschiedenen Benutzern aufgerufen werden kann. Service-Anwendungen werden in erster Linie für Dialoge verwendet, die nur aus einem Schritt bestehen. Sie werden z.B. bei häufigen Retrieval-Aktionen auf großen Datenbeständen eingesetzt.

Eine solche Sitzung wird ausschließlich im nicht synchronisierten Dialog betrieben (siehe Abschnitt „Nicht synchronisierter Dialog“ auf Seite 69). Service-Anwendungen starten bei Bedarf automatisch.

Service-Anwendung starten

Sie rufen eine Service-Anwendung auf durch Eingabe der URL am Browser oder durch die Angabe der URL in einem Formular (siehe Abschnitt „WebTransactions-Dialoganwendung starten“ auf Seite 100):

`http[s]://`*machine*`/`*cgiPath*`/WTPublish.exe/startup?`
`WT_SYSTEM_BASEDIR=`*basedir*`&WT_SERVICE_PAGE=`*service-template*`&`*par1*`=`*val1*`&...`

machine
: Internetadresse bzw. symbolischer Name des Rechners, auf dem WebTransactions installiert ist (ggf. mit Portnummer für den HTTP-Server).

cgiPath
: für den dortigen HTTP-Server vereinbarter Pfad (Präfix) für CGI-Programme

basedir
: Basisverzeichnis, unter dem die WebTransactions-Service-Anwendung installiert ist.
basedir ist ein absoluter Pfad (auf Windows mit Laufwerksbezeichner).

service-template
: Template, das den Service enthält. Der Name des Templates muss mit dem Suffix `.service` enden.

Beispiel

```
basedir/config/forms/myService.service
```

par1=val1
: In weiteren Name/Value-Paaren können Sie der Service-Anwendung weitere Parameter mitgeben.

Erhält `WTPublish` den Aufruf einer Service-Anwendung, wird zunächst eine freie WebTransactions-Sitzung gesucht, die für die Bereitstellung von Services gestartet wurde. Session Ids von Service-Anwendungen erhalten das Präfix `WTSVC` und einen fortlaufenden Index.

Beispiel

```
WTSVC-c__basedir_mybasedir-1
WTSVC-c__basedir_mybasedir-2
...
```

Sind alle laufenden Sitzungen beschäftigt, wird eine neue Sitzung für den Service gestartet. Diese arbeitet dann den Request ab und steht anschließend weiter zur Verfügung.

Das Attribut `WT_SYSTEM.TIMEOUT_USER` bleibt auch bei Sitzungen mit Service-Anwendungen wirksam. Dadurch können im Hochlastfall zusätzlich gestartete Sitzungen bei schwächerer Auslastung automatisch beendet werden.

Templates für Service-Anwendungen

Templates, die eine Service-Anwendung implementieren und als `WT_SERVICE_PAGE` referenziert werden sollen, müssen das Suffix `.service` haben.

Da der Dialog in einer Service-Anwendung jeweils nur aus einem Schritt besteht, können keine Informationen von einem Aufruf zum nächsten gespeichert werden. Deshalb sind bei der Programmierung eines solchen Templates die folgenden Punkte zu beachten:

– Die Attribute `WT_SYSTEM.HREF_ASYNC` und `WT_SYSTEM.HREF` werden nicht mit `SESSION`, `SIGNATURE` und `FORMAT_STATE` versorgt.

– `WTDataform` wird immer für den nicht synchronisierten Dialog generiert.

Service-Anwendungen in WebLab

Um eine Service-Anwendung in WebLab zu starten, verwenden Sie wie gewohnt den Befehl **Datei/Sitzung starten** (siehe Abschnitt „Sitzung starten“ auf Seite 174). Im Dialogfeld **Sitzung starten** geben Sie dann als Start-Template das Template für die Service-Anwendung an.

3.3 Templates

Ein Template ist eine Vorlage für die Generierung von spezifischem Code. Es enthält feste Teile, die beim Generieren übernommen werden, und variable Teile, die durch die jeweils aktuellen Werte ersetzt werden.

Dieser Abschnitt beschreibt folgende Template-Arten:

– WTML-Templates, deren Sprachmittel, die Definition von Arbeitsschritten innerhalb des Templates, sowie das Template während des Ablaufs im Dialogzyklus

– Master- und Klassen-Templates zur anwendungsweit einheitlichen Umsetzung bestimmter Bereiche oder Objekte der Formate

– Modul-Templates zur Definition von Klassen, Funktionen und Konstanten global für eine komplette Sitzung

– WTBeans als wiederverwendbare Komponenten für die Template-Programmierung

3.3.1 WTML-Templates

WTML-Templates (oder kurz Templates) sind der Dreh- und Angelpunkt jeder WebTransactions-Anwendung. Zur Laufzeit werden sie von WebTransactions ausgewertet und legen das Layout der im Browser angezeigten HTML-Seiten fest, steuern die WebTransactions-Anwendung und definieren die Kommunikationsschritte mit den Host-Anwendungen.

Die Automask-Templates (OSD, MVS) oder die formatspezifischen Templates (OSD, MVS, openUTM) erzeugen nach Standard-Vorgaben aus jeder Einheit der Host-Benutzeroberfläche (alphanumerisches Format) eine HTML-Seite. Diese automatisch generierten Templates können Sie unverändert einsetzen oder als Vorlage für individuelle Anpassungen verwenden. Für die Programmierung der Templates steht Ihnen die Template-Sprache WTML (WebTransactions Markup Language) zur Verfügung. Besonderen Komfort für die Bearbeitung und den Test von Templates bietet die WebLab-Entwicklungsumgebung.

Jedes formatspezifische Template ist standardmäßig in einer eigenen Datei *formatname*`.htm` im Verzeichnis *basedir*`/config/forms` abgelegt. Für unterschiedliche Layouts und unterschiedliche Sprachen können Sie noch weitere Verzeichnisse mit Templates anlegen, siehe auch Abschnitt „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61.

Sprachmittel in Templates

In den Templates können Sie folgende Sprachmittel verwenden:

- Standard-HTML-Tags, Text, client-seitiges JavaScript und alle weiteren Sprachmittel, die ein Browser interpretieren kann

  In Templates können Sie alle HTML-Tags, also auch `<SCRIPT>`-Tags, und konstanten Text verwenden, die der beim Benutzer eingesetzte Browser interpretieren kann. Damit stehen Ihnen alle JavaScript-Sprachmittel zur Verfügung, die die eingesetzten Browser interpretieren können.

  Da solche Scripts wie die Standard-HTML-Tags nicht von WebTransactions sondern vom Browser interpretiert werden, spricht man auch von client-seitigem JavaScript. Im MS Internet Explorer können Sie auch JScript bzw. VBScript verwenden. Diese client-seitigen JavaScripts werden wie HTML-Tags und fester Text unverändert an den Browser geschickt. Sie gehören aus Sicht von WebTransactions zum HTML-Bereich.

- WTML-Tags

  Mit WTML-Tags können Sie HTML-Seiten dynamisch generieren sowie die Host-Anwendung steuern. Durch sie werden z.B. Werte berechnet oder Informationen von der Host-Anwendung übermittelt.

  Bereiche mit WTML-Tags werden von WebTransactions interpretiert, und das Ergebnis wird an den Browser gesendet. Sie gehören also nicht zu einem HTML-Bereich.

- WTScripts

  WTScripts stehen ähnlich wie client-seitige JavaScripts in Bereichen, die mit speziellen Tags eingeleitet und beendet werden. Statt HTML-`SCRIPT`-Tags verwenden Sie hierfür jedoch WTML-Tags: `wtOnCreateScript` und `wtOnReceiveScript`. Damit zeigen Sie an, dass diese Scripts von WebTransactions und nicht vom Browser ausgeführt werden sollen und signalisieren zusätzlich den gewünschten Ausführungszeitpunkt. OnCreate-Scripts werden ausgeführt, bevor die Seite an den Browser geschickt wird. OnReceive-Scripts werden erst ausgeführt, nachdem die Antwort vom Browser empfangen wurde.

  Wie WTML-Tags werden WTScript-Bereiche von WebTransactions interpretiert, und das Ergebnis wird an den Browser gesendet. WTScript-Bereiche gehören damit nicht zum HTML-Bereich.

- Auswertungsoperatoren

  Durch die Auswertungsoperatoren können Sie die aktuellen Werte von Objekten bzw. Objekt-Attributen abfragen, siehe auch Abschnitt „Objekte – dynamische Daten“ auf Seite 72. Sie können auch beliebige Ausdrücke in Auswertungsoperatoren schreiben. Diese werden ausgewertet und das Ergebnis wird ausgegeben.

Die Sprachmittel der Template-Sprache WTML sind ausführlich im WebTransactions-Handbuch „Template-Sprache“ beschrieben.

OnCreate und OnReceive-Zeitpunkt

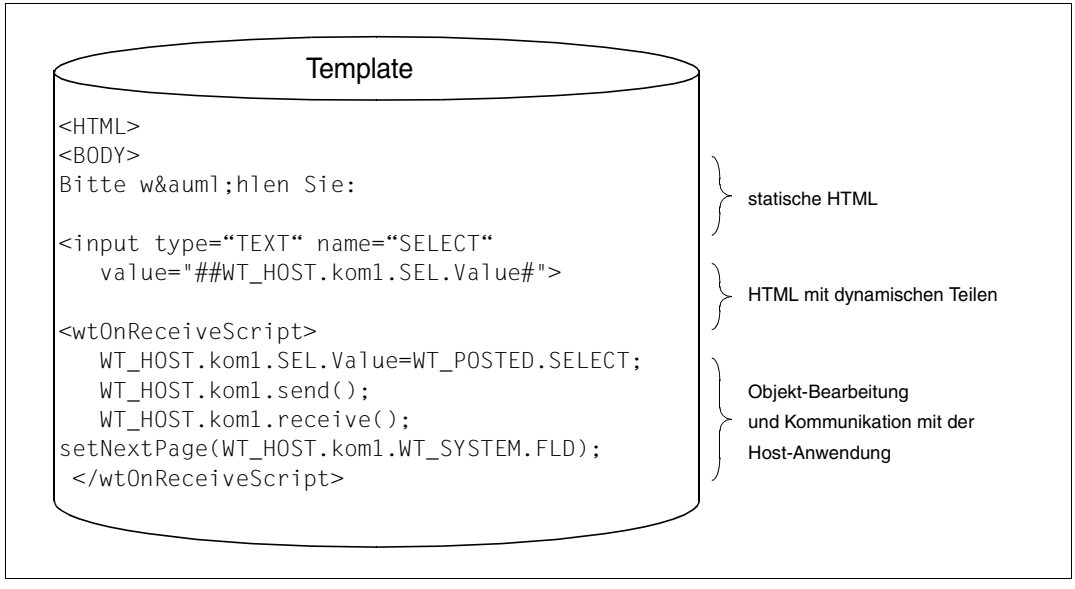
Im Template sind nicht nur die Verarbeitungsschritte definiert, die WebTransactions sofort bei der Generierung der HTML-Seite ausführen soll - also „OnCreate“, sondern auch Verarbeitungsschritte, die erst nach Empfang der vom Browser zurückgeschickten Daten ausgeführt werden sollen („OnReceive“) und auf Benutzereingaben reagieren. Diese OnReceive-Verarbeitungsschritte werden von WebTransactions zunächst zwischengespeichert und erst ausgeführt, nachdem die aktuelle HTML-Seite generiert, zum Browser gesendet und die vom Browser geschickten Daten zurückerhalten wurden.

Dieser Mechanismus ermöglicht es, in einem einzigen Template einen kompletten Dialogzyklus zu definieren (siehe auch Abschnitt „Dialogzyklus“ auf Seite 67):

– Aufbau der HTML-Seite, die zum Browser geschickt wird
– Nachbearbeitung der Daten, die als Antwort auf diese Seite vom Browser zurückgeschickt werden
– Senden der nachbearbeiteten Daten an die Host-Anwendung und Empfang der nächsten Host-Nachricht

Obwohl das Template nur einmal von WebTransactions interpretiert wird, werden die Verarbeitungsschritte zu unterschiedlichen Zeitpunkten wirksam.

Ein Beispiel-Template



Das Template enthält sowohl die HTML-Definitionen für die Ausgabe am Browser, als auch die Anweisungen für die Bearbeitung der Objekte und den Informationsaustausch mit der Host-Anwendung.

Die an den Browser geschickte HTML-Seite sieht folgendermaßen aus:

– Sie enthält als statischen Teil den Text: `Bitte wählen Sie:`

  Das Eingabefeld wird erzeugt. Es heißt `SELECT` und wird mit dem Wert vorbelegt, der im Host-Datenobjekt `SEL` im Attribut `Value` steht.

– Mit dem WTML-Tag `<wtOnReceiveScript>` wird ein WTScript-Bereich eingeleitet. In diesem sind Verarbeitungsschritte definiert, die zum OnReceive-Zeitpunkt ausgeführt werden sollen. Also erst dann, wenn der Browser die Benutzereingabe zurückgeliefert hat.

  Durch eine einfache Zuweisung wird der Wert, der im Attribut `SELECT` des Posted-Objekts `WT_POSTED` abgelegt ist, auf das Attribut `Value` des Host-Datenobjekts `SEL` übertragen. Der vom Benutzer eingegebene Wert liegt nun im Host-Datenobjekt.

Es folgt ein `send`-Aufruf, mit dem das aktuelle Host-Datenobjekt `SEL` zusammen mit allen anderen zu dieser Nachricht gehörenden Daten zur Host-Anwendung gesendet wird. Anschließend wird mit einem `receive`-Aufruf die nächste Host-Nachricht empfangen. Mit der Anweisung `setNextPage` wird das nächste zu verarbeitende Template bestimmt, das im Systemobjekt-Attribut `FLD` zur Verfügung gestellt wird.

3.3.2 Master-, Klassen- und Modul-Templates

Master- und Klassen-Templates dienen der anwendungsweit einheitlichen Umsetzung bestimmter Bereiche oder Objekte der Formate. Mit Modul-Templates können Klassen, Funktionen und Konstanten global für eine komplette Sitzung definiert werden.

3.3.2.1 Master-Templates

Master-Templates werden von WebTransactions bei der Generierung der Automask und der formatspezifischen Templates als Schablone verwendet und sorgen für ein einheitliches Layout.

Das Master-Template-Konzept zeigt seine Stärke vor allem bei Host-Anwendungen, bei denen viele Formate einen ähnlichen Aufbau haben, z.B. eine feste Einteilung in Kopfzeile, Arbeitsbereich und Fußzeile. In solchen Fällen genügt es, den Aufbau einmal im Master-Template festzulegen und dieses Master-Template bei der Generierung sowohl der formatspezifischen Templates als auch des Automask-Templates als Schablone zuzuweisen. Alle generierten Templates erhalten dann automatisch das einheitliche Layout.

Master-Templates können wie jedes andere Template feste HTML-Bereiche sowie beliebige WTML-Tags und WTScripts enthalten. Zusätzlich stehen in Master-Templates spezielle Master-Template-Tags zur Verfügung – kurz MT-Tags genannt –, die im WebTransactions-Handbuch „Template-Sprache“ beschrieben sind.

Für die einzelnen Liefereinheiten von WebTransactions werden eigene Master-Templates mit ausgeliefert, die Sie individuell anpassen, aber auch unverändert einsetzen können. Bei der Generierung geben Sie an, welches Master-Template verwendet werden soll.

3.3.2.2 Klassen-Templates

Mit Klassen-Templates können Sie die Auswertung von Host-Datenobjekten automatisieren. Statt für jedes Host-Datenobjekt dieselben Anweisungen immer wieder zu schreiben, definieren Sie entsprechende Klassen-Templates.

Ein Klassen-Template in WebTransactions enthält für die gesamte Objektklasse (z. B. Eingabe- oder Ausgabefeld) gültige, immer wiederkehrende Anweisungen. Klassen-Templates werden durchlaufen, wenn auf ein Host-Datenobjekt der Auswertungsoperator oder die toString-Methode angewendet wird.

Klassen-Templates werden vollständig in das rufende Template eingeschoben. In Klassen-Templates können Sie die gleichen Sprachmittel einsetzen wie in allen anderen Templates.

Klassen-Templates haben den Dateinamensuffix `.clt` (class template) und sind wie die normalen Templates im Standardfall im Verzeichnis *basedir*`/config/forms` abgelegt. Auch für Klassen-Templates können Sie unterschiedliche Stile und unterschiedliche Sprachen realisieren, wobei dieselbe Suchstrategie wie für normale Templates angewandt wird, siehe hierzu auch Abschnitt „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61.

Weitere Informationen zu Klassen-Templates finden Sie im WebTransactions-Handbuch „Template-Sprache“.

3.3.2.3 Modul-Templates

In Modul-Templates können Sie Klassen, Funktionen und Konstanten für WTScript definieren, auf die Sie während der gesamten WebTransactions-Sitzung in WTScripts und Auswertungsoperatoren zugreifen können. So können Sie z.B. WTScript-Funktions- und Klassenbibliotheken importieren. Achten Sie darauf, dass Klassen, Funktionen und Konstanten in `wtOnCreate`-Scripts definiert werden müssen.

Ein Modul-Template wird mit Hilfe der Funktion `import()` geladen. WebTransactions sucht das entsprechende Template gemäß eingestellter Sprache und eingestelltem Stil (siehe Abschnitt „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61).

3.3.3 WTBeans

Für die Programmierung der Templates stellt WebTransactions wiederverwendbare Komponenten, die WTBeans, bereit. Es wird zwischen inline und standalone WTBeans unterschieden:

- inline WTBeans entsprechen einem Teil eines WTML-Dokuments
- standalone WTBeans sind ein eigenständiges WTML-Dokument

WTBeans lassen sich zusätzlich nach ihrer Funktion unterscheiden:

- WTBeans für die Dialogführung mit dem Browser
  Die Komponenten steuern den Rahmen eines oder mehrerer Dialogschritte. Sie legen z.B. fest, ob der Dialog in einem einfachen Fenster, in mehreren Fenstern oder in verschiedenen Frames geführt wird.
- WTBeans für Dialogelemente für den Browser
  Die Komponenten dienen hauptsächlich der Darstellung von Daten im Browser, z.B. von Oberflächen, die durch mehrere HTML-Tags und client-seitige JavaScripts definiert sein können.

  Beispiel

  Das WTBean `wtcPopupDate` stellt ein Kalenderfenster zur Verfügung. Darin können Sie das Datum komfortabel mit der Maus auswählen.

- WTBeans für die Verarbeitung
  Die Komponenten erzeugen keine Oberfläche, sondern enthalten ein Stück Verarbeitungslogik, wie z.B. Kommunikation mit einem Host oder Anschluss an eine Userexit-Bibliothek.

Mit den WTBeans bietet WebTransactions eine Sammlung von wiederverwendbaren Komponenten, mit denen Sie die Kommunikation sowohl mit dem Browser als auch mit der Host-Anwendung steuern können. Bei der Auslieferung von WebTransactions werden nur die WTBeans für die Kommunikation mit der Host-Anwendung fest installiert. Siehe hierzu auch die Beschreibung in den protokoll-spezifischen Handbüchern.

Weitere WTBeans können Sie vom Download-Bereich der WebTransactions-Homepage unter dem Stichpunkt Ready-to-run herunterladen. WTBeans müssen eigens auf dem Rechner, auf dem WebLab läuft, installiert werden, siehe hierzu auch die zugehörige Dokumentation.

Bestandteile von WTBeans

Jedes WTBean hat einen eindeutigen Namen. Da dieser Name auch als Variablenname verwendet wird, wird zwischen Groß- und Kleinschreibung unterschieden. Die Namen beginnen mit `wtc` (für **WebTransactions C**omponent).

Beispiel

```
wtcStartOSD
```

Jedes WTBean wird durch eine Beschreibungsdatei definiert, die einem Template entspricht. Der Name der Beschreibungsdatei setzt sich zusammen aus dem Namen des WTBean und dem Suffix `.wtc`. Neben der Beschreibungsdatei können zu einem WTBean noch weitere Dateien gehören, wie z.B. Bilder, Dateien mit ausgelagerten Scripts etc.

Beispiel

```
wtcStartOSD.wtc
```

Jedes WTBean kann über Eigenschaften auf die jeweilige WebTransactions-Anwendung zugeschnitten werden. Aus der Beschreibungsdatei wird dazu für WebLab eine Oberfläche generiert, damit Sie die Eigenschaften komfortabel bearbeiten können.

Die Beschreibungsdatei eines WTBean enthält:

– den Prototypen einer HTML-Seite, eines Templates oder eines Template-Ausschnitts

– eine Beschreibung der verwendeten Ressourcen

– eine Beschreibung der Oberfläche für die Bearbeitung der Parameter

– eine Liste der Hilfsdateien und die Zielpfade im Basisverzeichnis

Nach der Installation befinden sich die mit ausgelieferten WTBeans im Unterverzeichnis `wtcCollection` des Installationsverzeichnisses von WebLab, und dort im entsprechenden Unterverzeichnis `inline` oder `standalone`. Diese Verzeichnisse enthalten wieder Unterverzeichnisse mit dem Namen der entsprechenden WTBeans, in denen dann die Beschreibungsdateien liegen.

Sobald Sie das erste WTBean mit WebLab genutzt haben, wird im Basisverzeichnis das Unterverzeichnis `wtcUsage` angelegt. Nach `wtcUsage` werden alle Beschreibungsdateien der WTBeans kopiert, die Sie für Ihre Template-Programmierung nutzen. Somit können Sie auch von anderen Rechnern aus diese Komponenten mit WebLab bearbeiten.

3.4 Struktur des Basisverzeichnisses

Ein Basisverzeichnis (*basedir* in verschiedenen Syntaxangaben) enthält alle Dateien, die zum Anbinden einer Host-Anwendung an das Web nötig sind.

Informationen zur Struktur von Basisverzeichnissen, die nur für eine einzelne Liefereinheit gelten, finden Sie im entsprechenden Benutzerhandbuch.

WTHolder (Windows- / Unix-Plattform)

Das Basisverzeichnis enthält einen Link mit Namen `WTHolder`, der auf das entsprechende Programm im Installationsverzeichnis zeigt:

Windows-Plattform *install_dir*`/lib/WTHolder.exe`

Unix-Plattform *install_dir*`/lib/WTHolder`

Für jeden Benutzer einer WebTransactions-Anwendung wird das WTHolder-Programm in einem eigenen Prozess gestartet. Das WTHolder-Programm bleibt während der gesamten Sitzung aktiv. Es steuert die Verarbeitung der Templates und die Kommunikation zwischen Browser, WebTransactions und Host-Anwendung. Das WTHolder-Programm sorgt auch dafür, dass bei Bedarf Module aus den gemeinsam nutzbaren Bibliotheken nachgeladen werden.

Optional: gemeinsam nutzbare Bibliotheken (Windows- / Unix-Plattform)

WebTransactions stellt die einzelnen Host-Adapter als gemeinsam nutzbare Bibliotheken zur Verfügung (z.B. `WTCommOSD.dll` unter Windows, auf der Unix-Plattform `WTCommOSD.so`). Unter OSD sind diese Bibliotheken nicht vorhanden; hier ist diese Funktionalität in `WTHolder` integriert.
Auch die mit ausgelieferten Userexits sind in solchen Bibliotheken enthalten.

Diese gemeinsam nutzbaren Bibliotheken werden beim Erzeugen des Basisverzeichnisses **nicht** automatisch im Basisverzeichnis angelegt. *WebTransactions* verwendet zur Laufzeit standardmäßig die Bibliotheken im Installationsverzeichnis. Im Basisverzeichnis sind Kopien dieser Bibliotheken nur dann sinnvoll, wenn vom Installationsverzeichnis abweichende Versionen verwendet werden sollen. Das ist z.B. dann der Fall, wenn Sie die ausgelieferte Userexit-Bibliothek mit eigenen Userexits ergänzt haben, oder ganz neue Userexit-Bibliotheken erzeugt haben.

Statisch gebundene WTHolder-Programme (BS2000/OSD)

Unter POSIX können keine gemeinsam benutzten Bibliotheken eingesetzt werden. Deshalb sind die Host-Adapter statisch in die WTHolder-Programme eingebunden (`WTHolderHTTP`, `WTHolderUTM`, `WTHolder...`).

Aus diesem Grund müssen auch Userexits fest eingebunden sein (siehe WebTransactions-Handbuch „Template-Sprache“).

Unterverzeichnisse

Das Basisverzeichnis enthält außerdem die Unterverzeichnisse `config`, `msg`, `tmp`, `wtcUsage` und `wwwdocs`, die in den folgenden Abschnitten beschrieben werden.

Sie können darüberhinaus weitere Unterverzeichnisse selbst anlegen, z.B. um Master-Templates oder statische HTML-Seiten zu archivieren.

3.4.1 Unterverzeichnis config

Das Verzeichnis `config` enthält im Unterverzeichnis `forms` Templates, welche die Oberfläche realisieren.

3.4.1.1 Unterverzeichnis forms

Die generierten Templates werden standardmäßig im Unterverzeichnis `config/forms` abgelegt. Dieses Unterverzeichnis ist immer vorhanden und sollte für jedes individuell angepasste Format ein Template enthalten. Die in `forms` abgelegten Templates bilden den Standardstil und die Standardsprache der individuell angepassten Templates.

Im Unterverzeichnis `forms` liegen auch die vordefinierten Templates, die von WebTransactions zur Verfügung gestellt werden.

3.4.1.2 Unterverzeichnisse für Stil- und Sprachvarianten

Sie können für ein und dieselbe logische Oberfläche einer Host-Anwendung Stil- und Sprachvarianten realisieren, mit denen Sie jeweils individuelle Nutzergruppen ansprechen. Diese Template-Varianten liegen in zusätzlichen Verzeichnissen.

In den Verzeichnissen können dann auch die entsprechenden Start-Templates stehen. Das Start-Template ist das erste Template, das nach dem Start einer WebTransactions-Sitzung gelesen wird (siehe Abschnitt „WebTransactions-Dialoganwendung starten“ auf Seite 100). Damit WebTransactions ein Start-Template in einem anderen Stil oder einer anderen Sprache findet, müssen Sie beim Starten der Sitzung die Attribute `STYLE` und `LANGUAGE` des Systemobjekts mit dem jeweils gültigen Wert versorgen. So steuern Sie, welches Layout WebTransactions verwendet.

Unterschiedliche Stil-Varianten (WT_SYSTEM.STYLE)

Für unterschiedliche Stil-Varianten gilt Folgendes:

– die Templates für den Standard-Stil müssen im Unterverzeichnis `config/forms` abgelegt sein.

– die Templates für einen anderen Oberflächenstil müssen in einem Unterverzeichnis liegen, das Sie (parallel zu `forms`) unter `config` einrichten und dessen Namen Sie frei wählen können. Im Attribut `STYLE` des Systemobjekts geben Sie diesen Namen dann an.

  Sie können beliebig viele Stil-Verzeichnisse einrichten. Die Templates in diesen Stilen realisieren z.B. Varianten mit mehr oder weniger Grafikelementen für unterschiedliche Browser-Ausstattungen.

Unterschiedliche Sprachen (WT_SYSTEM.LANGUAGE)

Wenn Sie Oberflächen im gleichen Stil aber in unterschiedlichen Sprachen anbieten wollen, legen Sie unterhalb der Stil-Verzeichnisse zusätzliche Verzeichnisse für die Sprach-Varianten an. Den Namen für ein Sprachverzeichnis können Sie frei wählen. Im Attribut `LANGUAGE` des Systemobjekts geben Sie diesen Namen an. Damit können Sie steuern, welche Sprache Sie verwenden wollen.

Suchstrategie

Sie brauchen nicht alle Templates in allen Stil- oder Sprachvarianten anzubieten. WebTransactions verwendet eine eigene Suchstrategie, um das passende Template zu finden.

Die Auswahl eines Templates wird über die Systemobjekt-Attribute `BASEDIR`, `STYLE`, `LANGUAGE`, `FORMAT` und `DEFAULT_FORMAT` gesteuert. WebTransactions sucht das passende Template nach folgender Strategie:

BASEDIR-Wert`/config/{`*STYLE-Wert*`|forms}[/`*LANGUAGE-Wert*`]/`*FORMAT-Wert*`[.htm]`

– `BASEDIR` gibt das Basisverzeichnis an
– an das Basisverzeichnis wird immer automatisch der Namensteil `config` angehängt
– danach folgt der Pfadteil aus dem Systemobjekt-Attribut `STYLE`. Wenn in `STYLE` kein Wert steht, gilt die Voreinstellung `forms` für diesen Pfadteil.
– danach folgt der Pfadteil aus dem Systemobjekt-Attribut `LANGUAGE`. Wenn in `LANGUAGE` kein Wert steht, entfällt dieser Pfadteil.
– der Dateiname des Templates steht im Systemobjekt-Attribut `FORMAT`. Das Suffix `.htm` kann entfallen.

Zunächst sucht WebTransactions nach dem Template in folgendem Dateiverzeichnis:

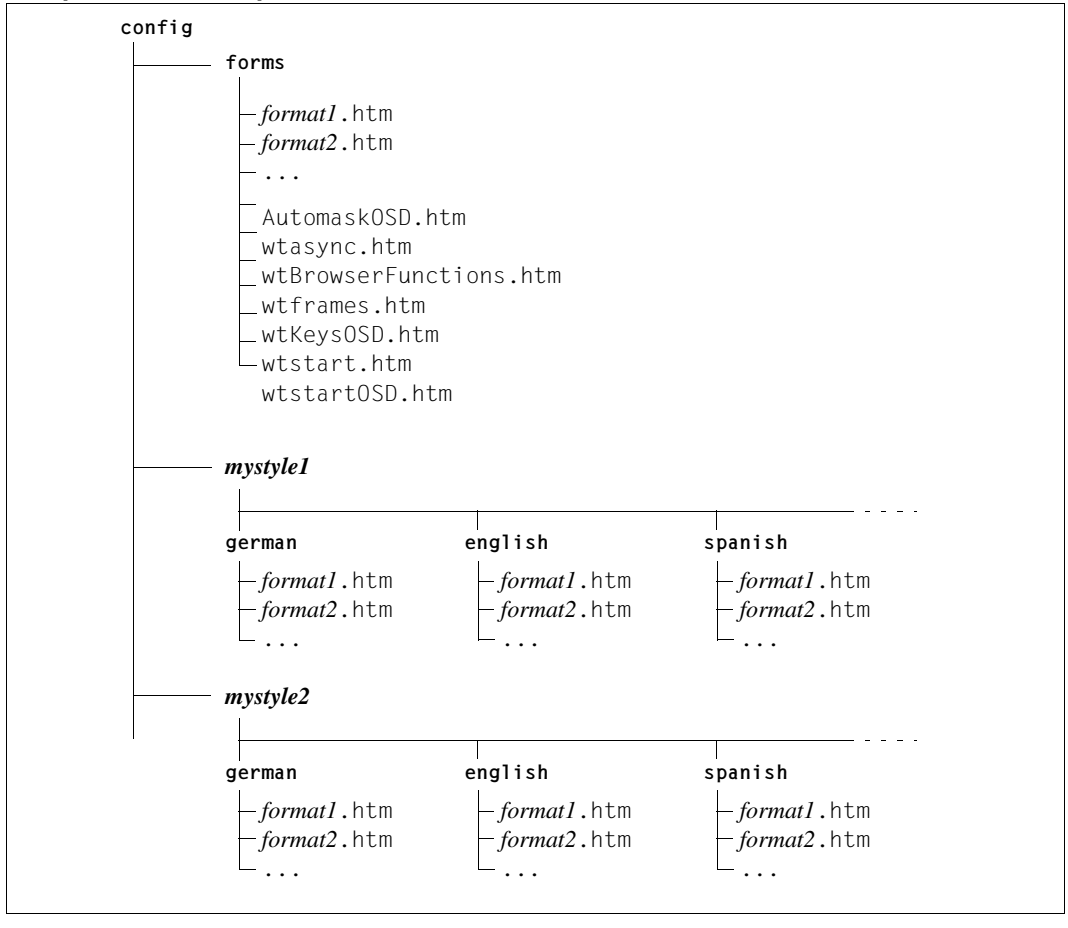
1. *BASEDIR-Wert*/config/*STYLE-Wert*/*LANGUAGE-Wert*

Da es aber nicht notwendig ist, jedes Template in jeder Variante auszugeben, sucht WebTransactions, falls das Template nicht gefunden wurde, in dem nächstmöglichen Verzeichnis nach folgender Reihenfolge:

2. *BASEDIR-Wert*/config/*STYLE-Wert*
3. *BASEDIR-Wert*/config/forms/*LANGUAGE-Wert*
4. *BASEDIR-Wert*/config/forms

Wird das in FORMAT spezifizierte Template in keinem dieser Verzeichnisse gefunden, versucht WebTransactions das in DEFAULT_FORMAT angegebene Template einzulesen. Dabei wird die gleiche Suchstrategie verwendet.

Beispiel: Stil- und Sprachvarianten



3.4.2 Unterverzeichnis msg

Das Unterverzeichnis `msg` enthält Links auf die Meldungsdateien im Installationsverzeichnis (`errmsgs`, `errmsgs*`) sowie auf die Dateien für die Anzeige der Meldungen am Browser, die Fehlermeldungstemplates `loctmpl` und `errtmpl`:

– `loctmpl` verwendet WebTransactions für die Darstellung von Fehlern, deren Ursache einer bestimmten Stelle eines Templates zugeordnet werden kann. Dabei wird jeweils der Name des betreffenden Templates, die Zeilennummer und die Spalte angegeben.

– `errtmpl` dient zur Darstellung aller Fehler, deren Ursache nicht in einem Template lokalisierbar ist.

Alle Dateien gibt es für die deutschen Meldungen (mit Suffix `.de`) und die englischen Meldungen (ohne Suffix).

Wenn Sie die Meldungen für eine WebTransactions-Anwendung mit WebLab ändern, wird beim Abspeichern eine Kopie mit den Änderungen in *basedir*`/msg` abgelegt. Der Verweis auf die entsprechende Datei im Installationsverzeichnis wird gelöscht.

Die geänderten Meldungen würden sonst für alle WebTransactions-Anwendungen auf dem Server gelten. Dasselbe gilt für die Dateien `loctmpl` und `errtmpl`.

Mit WebTransactions lassen sich die Meldungen in verschiedenen Sprachen zur Verfügung stellen. Die Sprache können Sie über das Attribut `LANGUAGE` des Systemobjekts einstellen. WebTransactions verwendet dann jeweils die Meldungsdatei `errmsg.`*lang* (bzw. `errmsg*.`*lang*) sowie die Fehlermeldungstemplates `loctmpl.`*lang* und `errtmpl.`*lang*. Dabei entspricht *lang* dem aktuellen Wert des Systemobjekt-Attributs `LANGUAGE`.

3.4.3 Unterverzeichnis tmp

Dieses Unterverzeichnis dient der Aufnahme temporärer Dateien. Für jede WTHolder-Task wird hier eine Datei *session*`.info` mit Informationen über die Sitzung abgelegt. Diese Informationen können dann über die Administration ausgewertet werden.

Außerdem wird für jede Verbindung in `tmp` ein Unterverzeichnis *session* angelegt. Dieses enthält evtl. Tracefiles sowie temporäre Dateien, die zur Laufzeit erzeugt werden. Hier können z.B. HTML-Dateien zur Unterstützung von Frames oder dynamisch erzeugte Grafiken liegen, die in die HTML-Seite mit eingebunden werden sollen.

Der Name *session* entspricht jeweils dem Wert des Attributs `SESSION` des Systemobjekts.

3.4.4 Unterverzeichnis wtcUsage

Dieses Unterverzeichnis wird nicht bei der Generierung des Basisverzeichnisses angelegt, sondern erst dann, wenn Sie die WTBeans nutzen. In `wtcUsage` werden die Beschreibungsdateien der verwendeten WTBeans gespeichert.

3.4.5 Unterverzeichnis wwwdocs

Zu einer WebTransactions-Anwendung können neben den Templates, die im Basisverzeichnis liegen, auch Dateien gehören, die zum Aufbau der HTML-Seite benötigt werden und direkt vom Web-Server erreichbar sein müssen (beispielsweise JavaScript-Dateien oder Bilddateien). Für diese Dateien exisiert im Basisverzeichnis das Verzeichnis `wwwdocs`, auf das der Web-Server zugreifen kann.

Wenn Sie alle Dateien Ihrer Anwendung im Basisverzeichnis speichern, können Sie die Anwendung bequem von einem Rechner zum anderen transferieren
(siehe Abschnitt „WebTransactions-Anwendung transferieren und verteilen“ auf Seite 214).

Das Verzeichnis `wwwdocs` existiert physikalisch unter dem Dokumentverzeichnis des Web-Servers. Im Basisverzeichnis wird ein symbolischer Link angelegt, der auf `wwwdocs` im Dokumentverzeichnis des Web-Servers zeigt. Das bedeutet, dass alle Dateien, die Sie im Basisverzeichnis im Unterverzeichnis `wwwdocs` ablegen, physikalisch im Dokumentverzeichnis des Web-Servers liegen. In WebLab werden sie wie ein Teil des Basisverzeichnisses behandelt und können deshalb auch für den Transfer mit in dem Archiv verpackt und auf dem Zielrechner entpackt werden.

`wwwdocs` und der symbolische Link im Basisverzeichnis werden bei der Generierung des Basisverzeichnisses angelegt. `wwwdocs` hat folgende Grundstruktur, die Sie nach eigenen Erfordernissen erweitern können:

```
wwwdocs
  |
  |-applet          Applets
  |-class           Java-Klassendateien
  |-html            HTML-Seiten, z.B. Aufrufseiten
  |-image           Bilddateien
  |-javascript      clientseitige JavaScripts
  |-style           Stylesheet-Definitonen
```

Auf Dateien in wwwdocs zugreifen

Für die Template-Programmierung können Sie die Dateien im Verzeichnis `wwwdocs` über das Systemobjekt-Attribut `WWWDOCS_VIRTUAL` ansprechen. `WWWDOCS_VIRTUAL` entspricht dem virtuellen Pfad, unter dem die Basisverzeichnisse Ihrer WebTransactions-Anwendungen liegen. Diesen vergeben Sie bei der Administration des WebTransactions-Servers für das Verzeichnis, siehe hierzu auch den Abschnitt „WebTransactions-Server verwalten“ auf Seite 150.

Beispiel

```
<img src="##WT_SYSTEM.WWWDOCS_VIRTUAL#/image/bild.gif">
```

Direkt können Sie die Dateien unter `wwwdocs` im Dokumentverzeichnis des Web-Servers folgendermaßen ansprechen:

document-root/virtual path/basedir/`wwwdocs`*/dateiname*

document-root	Pfad des Dokumentverzeichnisses des Web-Servers; diesen Pfad geben Sie bei der Installation von WebTransactions an
virtual path	virtueller Pfad, den Sie bei der Administration des WebTransactions-Servers für das Verzeichniss vergeben, unter dem die Basisverzeichnisse Ihrer WebTransactions-Anwendungen liegen, siehe hierzu auch den Abschnitt „WebTransactions-Server verwalten“ auf Seite 150.
basedir	Name des jeweiligen Basisverzeichnisses

3.5 Dialogzyklus

Eine Sitzung lässt sich als eine Folge von Dialogzyklen betrachten. Abhängig von der Dialogart (synchronisiert, nicht synchronisiert oder remote) werden in einem Dialogzyklus verschiedene Phasen durchlaufen.

Vertiefende Informationen zum Dialogzyklus entnehmen Sie dem Abschnitt „Dialog zwischen WebTransactions und Browser“ auf Seite 118.

3.5.1 Synchronisierter Dialog

Von WebTransactions aus betrachtet, besteht ein **Dialogzyklus** aus folgenden drei Phasen:

1. Template interpretieren / HTML generieren (WebTransactions)

   WebTransactions interpretiert das Template und generiert daraus die nächste HTML-Seite, die an den Browser geschickt werden soll. Die im Template enthaltenen HTML-Tags werden dabei unverändert in diese HTML-Seite übernommen. Für die Dynamik sorgen die Verarbeitungsschritte, die im Template in speziellen WTML-Tags und WTScripts definiert sind und dynamische Daten in die statischen HTML-Teile einfließen lassen. Sie dienen dazu, die Host-Daten in HTML umzusetzen oder mit der Host-Anwendung zu kommunizieren. So entsteht aus HTML-Bereichen und WTML-Bereichen eine HTML-Seite, die an den Browser geschickt wird.

   Nicht alle der im Template definierten Verarbeitungsschritte werden jedoch bereits beim Generieren der HTML-Seite (zum „OnCreate“-Zeitpunkt) wirksam:
   In OnReceive-Scripts definierte Verarbeitungsschritte werden von WebTransactions zunächst zwischengespeichert und erst in der 3. Phase des Dialogzyklus ausgeführt.

2. Daten eingeben oder Auswahl treffen (Benutzer am Web-Browser)

   Der Benutzer kann nun in der im Web-Browser angezeigten Seite die gewünschten Einträge machen. Vom Web-Browser wird die HTML-Seite an den HTTP-Server zurückgeschickt, der sie wiederum an WebTransactions weiterleitet (englisch posted). WebTransactions prüft die gesendeten Daten auf Aktualität. Wurde eine ältere Seite aus der Sitzung empfangen, wird die in Phase 1 generierte Seite noch einmal an den Browser geschickt. Sind die empfangenen Daten aktuell, wird mit ihnen die Phase 3 ausgeführt.

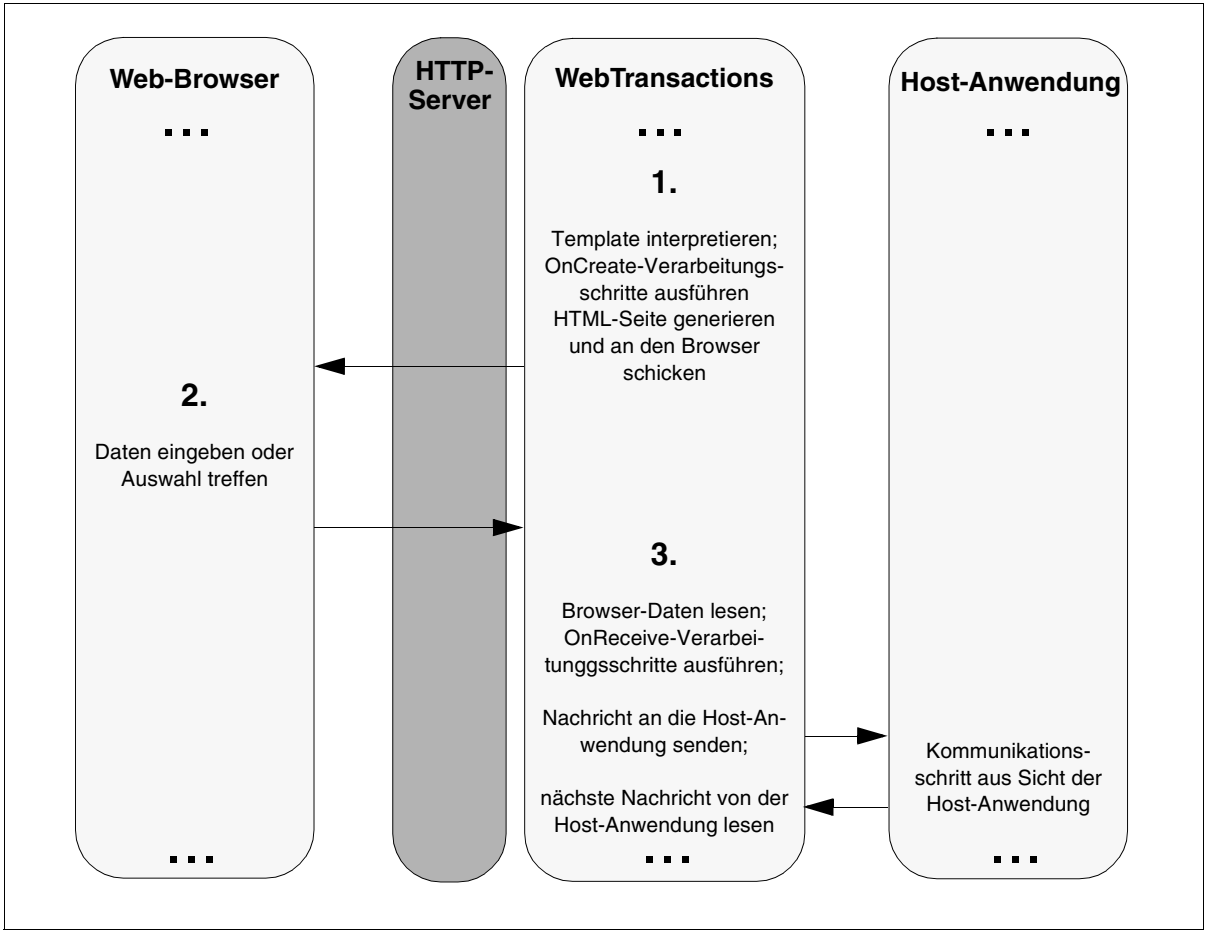
3. OnReceive-Verarbeitungsschritte ausführen (WebTransactions)

   WebTransactions führt nun die in Phase 1 zwischengespeicherten OnReceive-Verarbeitungsschritte aus. Die OnReceive-Verarbeitungsschritte bilden z.B. die vom Browser geschickten Daten auf das Format der Host-Anwendung ab und schicken die Daten an-

schließend zur Host-Anwendung. Im letzten OnReceive-Verarbeitungsschritt wird in der Regel die nächste Nachricht von der Host-Anwendung gelesen. Das Folge-Template wird durch die Funktion `setNextPage()` festgelegt, die den Namen des nächsten zu verarbeitenden Templates in das Systemobjekt-Attribut `WT_SYSTEM.FORMAT` schreibt.

Sie können den standardmäßig von der Host-Anwendung gesteuerten Dialogablauf aber auch aktiv ändern (siehe Abschnitt „Aktiver Dialog“ auf Seite 117).

Folgende Abbildung zeigt schematisch einen synchronisierten Dialog-Zyklus:



Dieser Dialogzyklus ist Teil des synchronisierten Dialogs, bei dem WebTransactions genau festgelegte Daten vom Browser erwartet, um die anstehenden OnReceive-Regeln zu verarbeiten. Wenn der Benutzer eine andere HTML-Seite als die erwartete zurückschickt, wird die zuletzt generierte HTML-Seite erneut an den Browser geschickt.

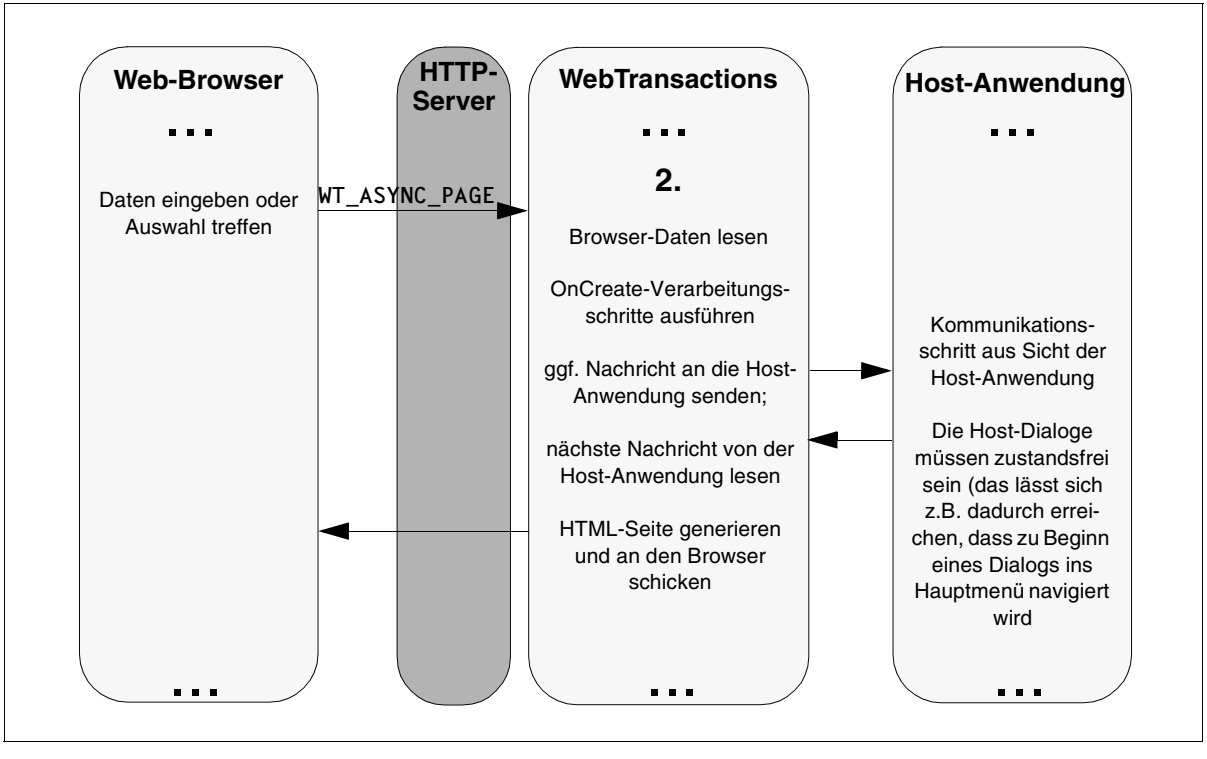
3.5.2 Nicht synchronisierter Dialog

Beim nicht synchronisierten Dialog entfällt die Überprüfung, ob die vom Browser gesendeten Daten aktuell sind (Phase 2 beim synchronisierten Dialog). Ein nicht synchronisierter Dialog besteht aus möglichen Ein-Schritt Dialogen ohne definierte Reihenfolge. Er beginnt mit einer Anfrage des Browsers, in der das auszuführende Template angegeben sein muss. Zusätzlich kann der Browser weitere Daten posten.

Jedes nicht synchronisiert aufgerufene Template generiert nach der Verarbeitung der On-Create-Regeln eine HTML-Seite und schickt diese an den Browser. Damit ist der nicht synchronisierte Dialogschritt abgeschlossen, eine Nachbearbeitung der vom Browser gesendeten Daten mit OnReceive-Regeln entfällt. Das bedeutet, dass die vom Browser gesendeten Daten nur mit OnCreate-Regeln verarbeitet werden können.

Damit reduziert sich der Dialogzyklus beim nicht synchronisierten Dialog auf den Schritt „Template interpretieren / HTML generieren“. Weitere Informationen zum synchronisierten und nicht synchronisierten Dialog finden Sie im Abschnitt „Dialog zwischen WebTransactions und Browser“ auf Seite 118.

Folgende Abbildung zeigt schematisch einen nicht synchronisierten Dialog-Zyklus:



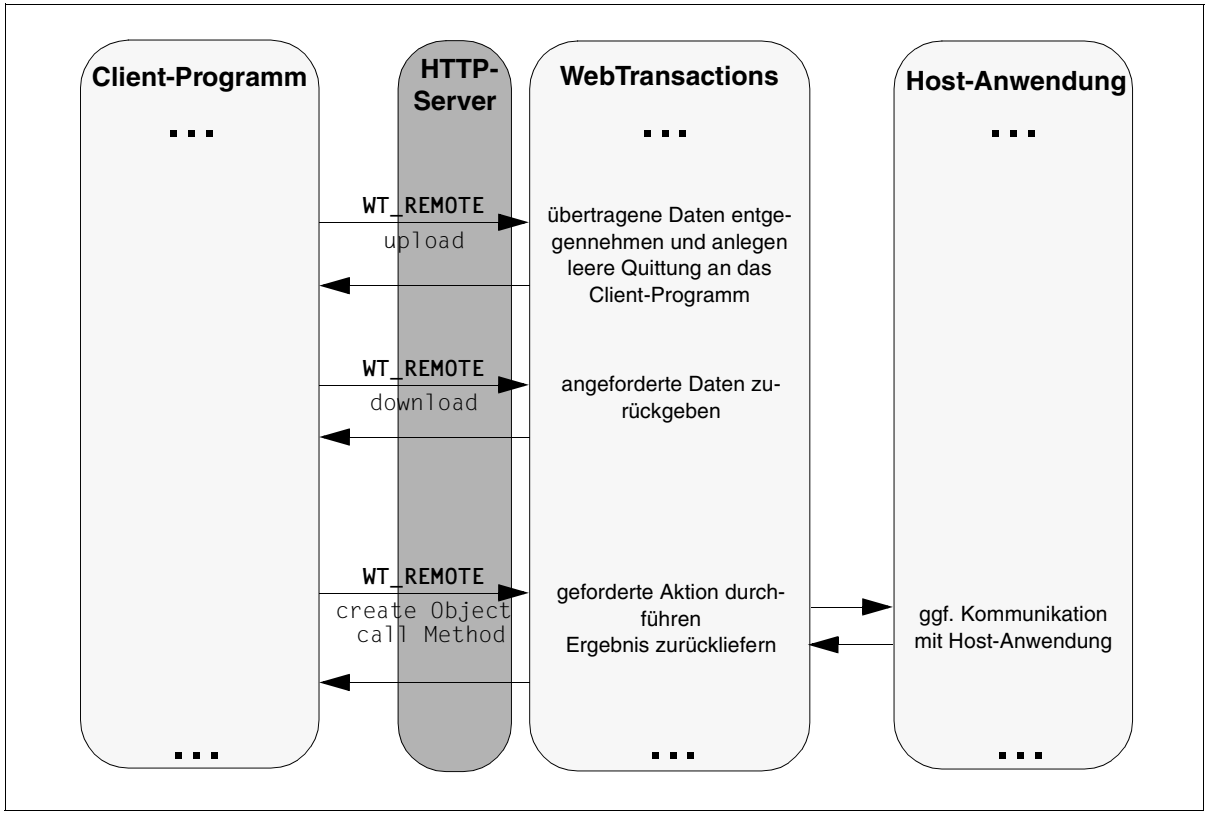
3.5.3 Dialog über Client-Schnittstelle

Bei Zugriffen auf WebTransactions über die Client-Schnittstelle `WT_REMOTE` werden keine HTML-Seiten generiert. Der Browser wird durch ein Client-Programm ersetzt, das mit der WebTransactions-Anwendung Daten austauscht und Programme in der WebTransactions-Sitzung ablaufen lässt. Die einzelnen Zugriffe werden daher auch nicht mit einem Template verknüpft, sondern direkt von Client-Programm aus gesteuert.

Das Client-Programm kann Daten an WebTransactions übertragen. Abhängig vom Inhalt werden diese Daten als neue globale Variablen oder als Attribute von `WT_SYSTEM` oder `WT_HOST` in der WebTransactions-Sitzung angelegt. Umgekehrt kann das Client-Programm auch Daten der adressierten WebTransactions-Sitzung abfragen. Zudem können in der WebTransactions-Sitzung eingebaute oder selbst definierte Konstruktoren, Methoden und Funktionen ausgeführt werden. Das Ergebnis eines solchen Aufrufs wird zurück an das Client-Programm übertragen.

Die Aktionen, die zugehörigen Daten und die Ergebnisse, die zwischen Client-Programm und WebTransactions ausgetauscht werden, sind in einer XML-basierten Sprache codiert. Weitere Informationen zum Aufbau der Nachrichten finden Sie im WebTransactions-Handbuch „Client-APIs für WebTransactions“.

Folgende Abbildung zeigt den prinzipiellen Ablauf der unterschiedlichen Aktionen:



3.6 Objekte – dynamische Daten

WebTransactions unterstützt ein leistungsfähiges und einfaches Objektmodell, mit dem die zwischen Browser und Host-Anwendung ausgetauschten Daten zum Entwicklungszeitpunkt modelliert und die Kommunikationsvorgänge während des Ablaufs gesteuert werden. WebTransactions strukturiert die Daten in folgender Objekthierarchie:

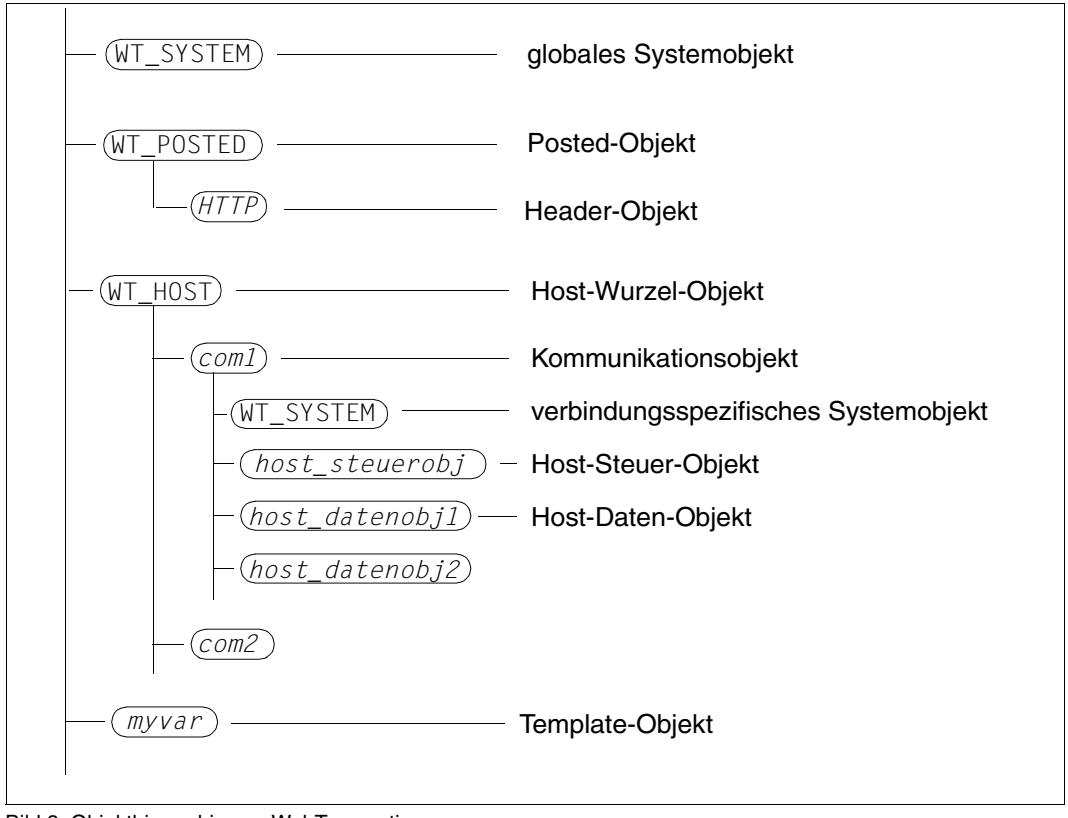


Bild 3: Objekthierarchie von WebTransactions

Die Rahmen in der Grafik zeigen, dass es sich um Objekte handelt, die wiederum Objekte enthalten. Kursiv geschriebene Werte zeigen an, dass die Namen der Objekte vom konkreten Anwendungsfall abhängen.

Während eines Dialogzyklus - siehe dazu auch Abschnitt „Dialogzyklus“ auf Seite 67 - werden die von WebTransactions definierten Objekte folgendermaßen verwendet:

– Das globale Systemobjekt (`WT_SYSTEM`) enthält Attribute, die über die gesamte WebTransactions-Sitzung Gültigkeit haben. WebTransactions verwendet sie unter anderem, um Informationen über die aktuelle Sitzung zu verwalten.

– Das Posted-Objekt (`WT_POSTED`) enthält die Eingaben des Benutzers.

– Die Kommunikationsobjekte enthalten die Daten der zum aktuellen Zeitpunkt bestehenden Verbindungen. Kommunikationsobjekte werden verwendet, um Daten an die Host-Anwendung zu schicken und Daten von der Host-Anwendung zu empfangen.

– Die Host-Datenobjekte dienen dem Datenaustausch zwischen WebTransactions und der Host-Anwendung. Sie repräsentieren die Felder eines Formats der Host-Anwendung bzw. die Datenobjekte (XML-Struktur) einer Partneranwendung, mit der über den HTTP-Adapter kommuniziert wird. Sie werden beim Aufruf der Methode `receive` angelegt und werden beim Aufruf von `send` zur Host-Anwendung übertragen.

– Template-Objekte enthalten kurzlebige Daten, die für die Dauer der Interpretation eines Templates zwischengespeichert werden. Sie werden mit WTML-Sprachmitteln und Funktionen angelegt.

  Zusätzlich liegen Objekte vor, die in Modul-Templates definiert worden sind (siehe Abschnitt „Modul-Templates“ auf Seite 56). Diese Objekte bleiben für die Dauer der gesamten WebTransactions-Sitzung erhalten.

Die folgende Grafik führt die einzelnen Aktionen und die Verwendung der Objekte für einen synchronisierten Dialogzyklus näher aus.

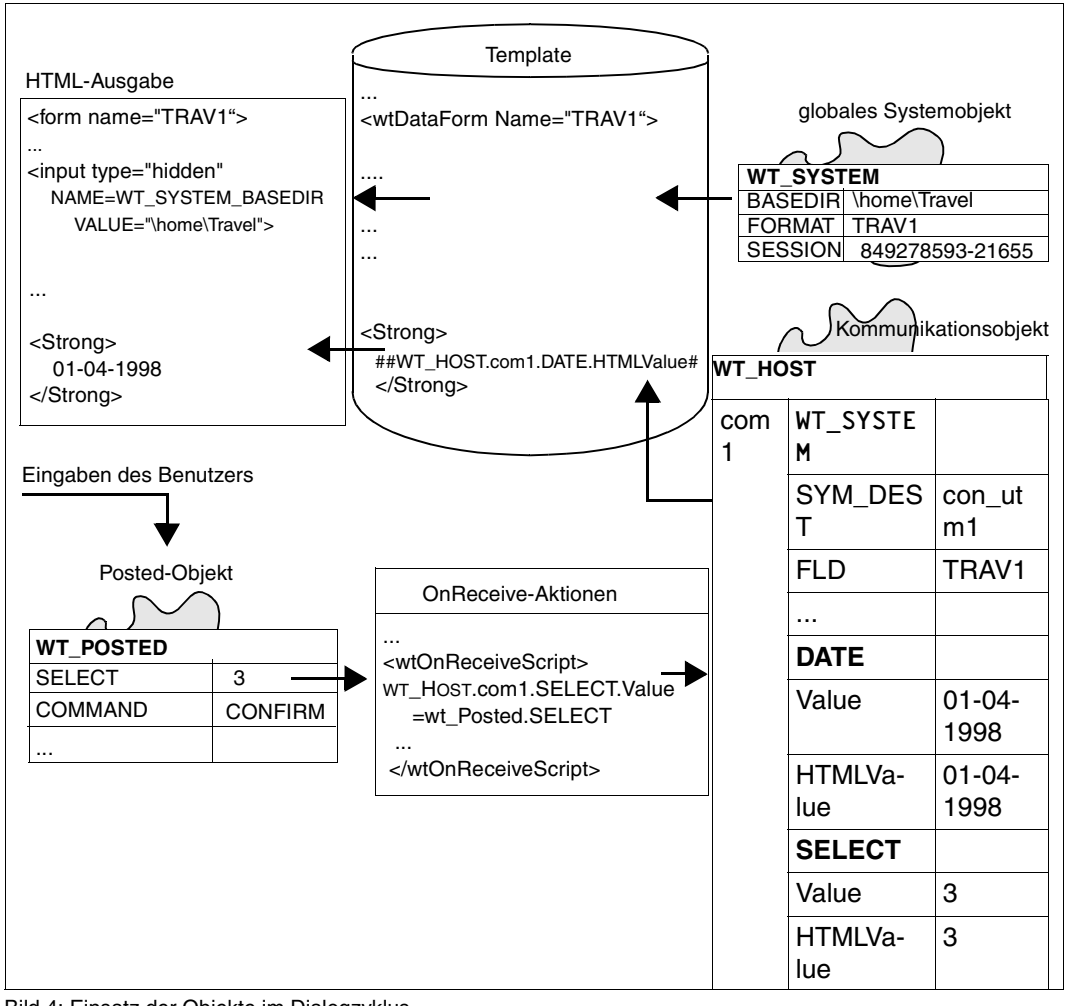


Bild 4: Einsatz der Objekte im Dialogzyklus

Die Bedeutung der unterschiedlichen Objekte, ihre wesentlichen Eigenschaften und Methoden, ihr Zusammenspiel und ihre Lebensdauer werden in den folgenden Abschnitten im Detail beschrieben.

3.6.1 Lebensdauer von Objekten

Die folgende Tabelle zeigt die Lebensdauer der von WebTransactions vordefinierten Objekte:

Objekt	**Objekt-Lebensdauer**
globales Systemobjekt `WT_SYSTEM`	Dauer der Sitzung bei Attributen, die Sie selbst angelegt haben, bis zum Ende der Sitzung oder bis zum expliziten Löschen
Posted-Objekt `WT_POSTED`	Dauer der Sitzung **synchronisiert** die Attribute geben die zuletzt vom Browser empfangenen Daten wieder, sie existieren bis die nächsten vom Browser empfangen werden **nicht synchronisiert** die Attribute enthalten die Daten, mit denen vom Browser aus das asynchrone Template aufgerufen wurde **remote** nur zu Analysezwecken vorhanden
Host-Wurzelobjekt `WT_HOST`	Dauer der Sitzung
Kommunikationsobjekt Attribut von `WT_HOST`	vom Anlegen mit einem Konstruktoraufruf von `WT_Communication` bis zum Ende der Sitzung.
verbindungsspezifisches Systemobjekt Attribut eines Kommunikationsobjekts	vom Anlegen mit einem Konstruktoraufruf von `WT_Communication` bis zum expliziten Löschen oder Ende der Sitzung
Host-Datenobjekt Attribut eines Kommunikationsobjekts	vom Anlegen mit `receive` bis zum nächsten Aufruf dieser Funktion (bei dem ältere Host-Datenobjekte zerstört werden) oder bis `close`
Host-Steuerobjekt Attribut eines Kommunikationsobjekts	vom Anlegen mit `open` bis `close`

Objekt	Objekt-Lebensdauer
Template-Objekte	**synchonisiert/nicht synchronisiert** Die Objekte leben vom expliziten Anlegen im Template bis zum Verarbeitungsende des Templates. **remote** Die Objekte leben vom expliziten Anlegen durch Methoden, Konstruktoren oder upload-Zugriffen bis zum Ende der Kommunikation oder zum expliziten Löschen Objekte aus Modul-Templates Die Objekte leben für die Dauer der Sitzung oder bis zum expliziten Löschen

3.6.2 Sichtbarkeit der Objekte

Die Sichtbarkeit der Objekte und Variablen unterscheidet sich nach der Art des Zugriffs auf WebTransactions, abhängig davon, ob auf die WebTransactions-Anwendung synchronisiert, nicht synchronisiert oder über die Schnittstelle `WT_REMOTE` zugegriffen wird. WebTransactions verwaltet die Posted- und Template-Objekte für jede Zugriffsart getrennt, in WebLab dargestellt durch die verschiedenen Objektbäume:

– Die Objekte, die während eines synchronisierten Zugriffs entstehen, werden im Objektbaum **synchron** angezeigt.

– Die Objekte, die während eines nicht synchronisierten Zugriffs entstehen, werden im Objektbaum **asynchron** dargestellt

– Die Variablen, die während eines Zugriffs über `WT_REMOTE` entstehen, werden im Objektbaum **remote** angezeigt.

Diese getrennte Verwaltung hat auch Auswirkungen auf die Sichtbarkeit und damit auf den Zugriff der Variablen:

– In einem Template, das einen synchronisierten Dialogschritt durchführt, kann nicht auf Template-Variablen und auf das Posted-Objekt zugegriffen werden, die in einem nicht synchronisierten Schritt gültig sind und umgekehrt.

– Es kann durchaus vorkommen, dass zur gleichen Zeit in unterschiedlichen Dialogschritten (synchronisiert, nicht synchronisiert und remote) Variablen gleichen Namens existieren. Diese Variablen enthalten aber unterschiedliche Werte, da sie unabhängig voneinander verwaltet werden.

– Funktionen, die im aktuellen synchronisierten Dialogschritt definiert werden, stehen bei einem nicht synchronisierten Zugriff nicht zur Verfügung.

3.6.3 Globales Systemobjekt wt_System - Sitzungssteuerung und langfristige Datenspeicherung

Das globale Systemobjekt `WT_SYSTEM` wird zu Beginn einer WebTransactions-Sitzung eingerichtet und lebt für die gesamte Dauer der Sitzung. WebTransactions legt Informationen über den aktuellen Zustand der Sitzung in Attributen dieses Objektes ab.

Sie können eigene Attribute zur langfristigen Speicherung anlegen. Diese lassen sich dann in späteren Dialogzyklen wieder abfragen.

Das Systemobjekt gibt es nur einmal für jede Sitzung. Änderungen in einer Dialogart (synchronisiert, nicht synchronisiert, remote) sind auch während der jeweils anderen Dialogarten sichtbar.

3.6.3.1 Langfristige Datenspeicherung

Da das Systemobjekt für die Dauer einer Sitzung lebt, ist es für die längerfristige Speicherung von Daten geeignet. Template-Objekte, die in WTScripts angelegt werden, existieren nur für den jeweiligen Dialogzyklus.

Sie können z.B. ein Benutzerprofil zwischenspeichern, das auf einer HTML-Seite erfasst und auf einer späteren Seite wieder ausgewertet wird.

Eigene Systemobjekt-Attribute werden durch einfaches Zuweisen erzeugt und mit einem Wert belegt. Wenn bereits ein Attribut mit dem gewählten Namen vorhanden ist, wird sein Inhalt überschrieben.

Um Konflikte mit vordefinierten Attributen zu vermeiden, sollten Sie Ihre eigenen Attribute mit einem Unterstrich _ beginnen.

Beispiel

Ein Script legt das Attribut `_NEUES_ATTRIBUT` mit dem Wert `test` an:

```
<wtOnCreateScript>
   WT_SYSTEM._NEUES_ATTRIBUT="test";
</wtOncreateScript>
```

3.6.3.2 Globale Steuerung einer Sitzung

Die Informationen darüber, welches Template als nächstes gelesen wird, welche Sprache für die Oberfläche eingestellt ist, ob Fehlermeldungen unterdrückt werden usw. sind als Attribute des Systemobjekts hinterlegt. Diese Attribute können abgefragt und z.T. auch geändert werden. Sie steuern damit das Verhalten des Kerns von WebTransactions.

Einen Überblick über die Attribute und ihre Wirkung gibt die folgende Tabelle (weitere Attribute werden bei der Kommunikations-Schnittstelle beschrieben). Einige Attribute werden beim Start von WebTransactions gesetzt (start) oder können durch Aktionen in den Temp-

lates verändert oder abgefragt werden (template), um bestimmte Steuerungsfunktionen zu realisieren. Andere Attribute werden als verdeckte Felder („hidden fields“) mit jeder Seite geschickt (hidden).

Im Folgenden werden nur diejenigen Attribute des Systemobjekts beschrieben, deren Bedeutung für alle Protokollvarianten von WebTransactions gleich ist. Attribute, die es speziell für protokollspezifische Anbindungen gibt oder die zumindest für protokollspezifische Anbindungen eine spezielle Bedeutung haben, finden Sie in den Benutzerhandbüchern der einzelnen Liefereinheiten.

Der Einsatz der Attribute wird in der letzten Spalte aufgeführt.

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
BASEDIR	Basisverzeichnis der WebTransactions-Anwendung	Dieses Attribut wird auf der Aufrufseite gesetzt. Es kann im weiteren Verlauf nicht geändert werden. Wert: absoluter Pfadname */pfad/basedir*	hidden start
CGI	CGI-Umgebungs-Variablen	Dieses Attribut hat den Datentyp `object`. Es enthält die angegebenen Attribute, die den Datentyp `string` haben. Sie speichern die Werte der CGI-Umgebung, die beim Aufruf des CGI-Programms WTPublish.exe vom HTTP-Server übergeben werden.	start
AUTH_TYPE		Verfahren, das zur Authentifizierung verwendet wird	
GATEWAY_INTERFACE		CGI-Version des Servers	
HTTP_ACCEPT		Liste von MIME-Typen, die der Client akzeptiert	
HTTP_ACCEPT_CHARSET		Zeichensatz, den der Client akzeptiert	
HTTP_ACCEPT_ENCODING		Information des Browsers, welche Dokumentarten er verarbeiten kann	
HTTP_ACCEPT_LANGUAGE		Sprache, die der Client akzeptiert	
HTTP_USER_AGENT		Bezeichnung des Browsers	
PATH_INFO		zusätzlicher Pfad in der angeforderten URL relativ zu den im WWW-Server eingestellten Root-Verzeichnissen	
PATH_TRANSLATED		Pfad von PATH_INFO entsprechend der Verzeichnisstruktur des Server-Rechners	
QUERY_STRING		an die URL mit ? angehängte Angaben Wird beim Absenden eines HTML-Formulars ein CGI-Script aufgerufen, stehen in dieser Umgebungsvariablen die ausgefüllten Formulardaten.	
CONTENT_LENGTH		Enthält die Anzahl der Zeichen, die beim Aufruf des CGI-Scripts über die POST-Methode übergeben wurden	
CONTENT_TYPE		Enthält beim Aufruf über die POST-Methode den Mime-Type der übergebenen Daten	
REFERER_URL		URL, von der aus WebTransactions gestartet wurde	
REMOTE_ADDR		IP-Adresse des Anfrage-Rechners	

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
REMOTE_HOST		Name des Anfrage-Rechners	
REMOTE_IDENT		Benutzerkennung des Anwenders auf Anfrage-Rechner	
REMOTE_USER		wie REMOTE_IDENT welche der beiden Variablen gesetzt ist, hängt vom Server ab	
REQUEST_METHOD		Methode der HTTP-Anfrage (Post oder Get)	
SCRIPT_NAME		virtueller Pfad von WTPublish.exe	
SERVER_NAME		Name oder IP-Adresse des Servers	
SERVER_PORT		Nr. des Ports, an dem die Anfrage einging	
SERVER_PROTOCOL		Protokollversion/-art	
SERVER_SOFTWARE		Software des HTTP-Servers	
CHARSET	Zeichensatz-Angabe im Feld Content-Type des HTTP-Headers	Ist dieses Attribut gesetzt, dann wird der Inhalt als Wert im Feld Content-Type des HTTP-Header in die generierte Nachricht für den Browser geschrieben. Browser können dynamisch den entsprechenden Zeichensatz einstellen. Ist das Attribut nicht gesetzt, dann generiert WebTransactions folgenden Header: Content-type; text/html; charset=ISO-8859-1 Siehe auch Attribut HTTP_HEADER CHARSET wird vom Attribut HTTP_HEADER übersteuert.	template
COMMUNICATION_ ERROR_FORMAT	Template für einen Fehlerausgang	Tritt in einem OnReceiveScript-Tag bei open, send oder receive Aufrufen ein Fehler auf und hat diese Variable nicht den Wert Leerstring, wird auf das angegebene Template verzweigt. Treten Kommunikationsfehler in einem OnCreateScript auf, so bleibt dieses Attribut unberücksichtigt. Diese Fehler können nur durch Abfrage des Attributs ERROR erkannt und behandelt werden. Werte: Leerstring: kein Fehlerausgang definiert; Dateiname des Templates *template*[.htm] : Fehlerausgang definiert.	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
COMMUNICATION_ ERRORS_DISABLED	Schalter zur Deaktivierung von Meldungen bei Kommunikationsfehlern	Mit diesem Attribut können Sie steuern, ob Kommunikations-Fehlermeldungen an den Browser/Benutzer weitergereicht werden sollen. Im Produktiveinsatz sollten Sie dieses Attribut mit einem Wert belegen, um den Benutzer nicht durch evtl. auftretende Meldungen zu irritieren. Die Fehlerbehandlung sollte im Template erfolgen, z.B. durch Abfragen des Attributs *ERROR* oder Setzen des Attributs *COMMUNICATION_ERROR_FORMAT*. Werte: Leerstring: Weiterreichen von Fehlern Wert ungleich leer: kein Weiterreichen von Fehlern	template
DEFAULT_FORMAT	Voreinstellung für fehlendes Template	Wird das in FORMAT angegebene Template nicht gefunden, so wird versucht, das in DEFAULT_FORMAT angegebene Template zu lesen	template
DIALOG_CONTROL_ FORMAT	Steuerungstemplate	Dieses Attribut wird in den generierten Templates verwendet, um den von der Host-Anwendung geführten Dialog zu unterbrechen und auf ein "übergeordnetes" Template (z.B. Menüschirm) zurückzukehren. Das Template wird durch die Schaltfläche **Suspend** angesprungen.	template
ERROR	Fehlermeldung	Nach Ausführen der Kommunikationsfunktionen hinterlegt WebTransactions in dieser Variablen eine Fehlermeldung, wenn die Aktion fehlgeschlagen ist. Im Erfolgsfall wird die Variable auf Leerstring gesetzt. Der Wert der Variablen kann im Template abgefragt werden (siehe auch COMMUNICATION_ERROR_FORMAT). Eine elegantere Weise auf Fehler während der Kommunikation zu reagieren, ist die Verwendung von Exceptions (siehe WebTransactions-Handbuch „Template-Sprache“).	template
ERROR_LOGFILE	Protokolldatei für Fehler	Ist dieses Attribut mit einem Dateinamen belegt, werden alle Fehlerausgaben von WebTransactions unterdrückt und statt dessen in die angegebene Datei geschrieben. Den Dateinamen geben Sie ohne Pfad an. Sie wird im Basisverzeichnis angelegt.	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
EXIT_SESSION	Sitzungsende	Wenn dieses Attribut vorhanden ist, beendet WebTransactions die Sitzung nach dem Senden der nächsten HTML-Seite zum Browser. Mit der Funktion exitSession() wird dieses Attribut definiert. Wenn das Attribut PREVENT_EXIT_SESSION existiert, wird die WebTransactions-Sitzung trotz Ausführen der Funktion *exitSession()* nicht beendet, siehe auch Seite 86.	template
FORMAT	Name des nächsten Templates (synchronisierter Dialog)	Dieses Attribut bestimmt beim synchronisierten Dialog das nächste zu lesende Template. Es kann in dem URL oder auf der Aufrufseite angegeben werden sowie durch WTML-Tags (Zuweisungen) im Template. Dieses Attribut wird mit der Funktion setNextPage() oder durch eine explizite Zuweisung gesetzt. Werte: Dateiname des Templates *template*[.htm] (siehe Abschnitt „Suchstrategie“ auf Seite 62)	template
FORMAT_STATE	Zustand der aktuellen Seite	Anhand dieses Attributs wird sichergestellt, dass keine Seite außerhalb der vorgeschriebenen Reihenfolge verarbeitet wird. Das Attribut wird automatisch mit einem Wert versorgt, der für jedes HTML-Formular eindeutig ist. So können Fehlbedienungen durch Verwendung der Browser-History behandelt werden. Es kann nicht geändert werden.	hidden
HANDLE	Name des aktuellen Kommunikationsobjekts	Dieses Attribut wird nur noch aus Kompatibilitätsgründen zur V1.0 und V2.0 weiter unterstützt. Ab der Version V3.0 können Sie jedes Kommunikationsobjekt benennen und es direkt ansprechen. Der Wert dieses Attributs stellt ein Kommunikationsobjekt als Standard ein.	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
HREF	Link. Daten für die Erzeugung eines Links, der die laufende WebTransactions-Sitzung synchron aufruft	Dieses Attribut kann im Template für die Definition eines Links verwendet werden. Es enthält den Namen des CGI-Moduls WTPublish.exe und einen Wert zur Identifikation der Seite (*BASEDIR, SESSION, FORMAT, FORMAT_STATE, LANGUAGE, SIGNATURE, TIMOUT_APPLICATION*). Werte: *url_of_webtransactions*&WT_=*value* der HTML-Link <A HREF="##WT_SYSTEM.HREF#"> .</A> ruft die aktuelle WebTransactions-Sitzung, siehe auch Abschnitt „Starten durch Eingabe der URL“ auf Seite 102.	template
HREF_ASYNC	Header für Hyperlink. Daten für die Erzeugung eines Links, der die laufende WebTransactions-Sitzung asynchron aufruft	Enthält Kopf (Scriptname und Name/Value-Paare anderer Systemvariablen) für Hyperlinks innerhalb einer Sitzung. Im Gegensatz zum Attribut HREF öffnet diese URL WebTransactions für einen nicht synchronisierten Dialogzyklus. Das Template dafür muss als zusätzliches Name/Value-Paar mit Namen WT_ASYNC_PAGE angegeben werden. Ein HTML-Link der Form <A HREF= "##WT_SYSTEM.HREF_ASYNC#&WT_ASYNC_ PAGE=myPage"> </A> ruft die aktuelle WebTransactions-Sitzung für einen nicht synchronisierten Dialogzyklus. Wird WT_ASYNC_PAGE nicht angegeben, so gibt WebTransactions noch einmal die zuletzt synchron generierte Seite aus (dieses Verhalten lässt sich z.B. gut nutzen, wenn innerhalb eines Dialogs dynamisch Frames um Dialogseiten geladen oder entladen werden sollen, siehe Beispiel in Abschnitt „Dialog zwischen WebTransactions und Browser“ auf Seite 118).	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
HTTP_DEFAULT_HEADER	HTTP-Header für die gesamte Sitzung	Ist dieses Attribut gesetzt, dann wird der Inhalt als HTTP-Header in die generierte Nachricht für den Browser geschrieben. Das Attribut muss alle erforderlichen Header, Zeilenvorschübe (CR LF) und den abschließenden doppelten Zeilenvorschub (CR LF CR LF) enthalten. Solange HTTP_DEFAULT_HEADER gesetzt ist, wird in jede generierte Nachricht dieser Header geschrieben. WebTransactions generiert dann keine eigenen Header-Felder mehr; der HTTP-Server kann allerdings gegebenenfalls weitere HTTP-Header-Felder ergänzen, wenn diese nicht in HTTP_DEFAULT_HEADER enthalten sind.	template
HTTP_HEADER	HTTP-Header	Ist dieses Attribut gesetzt, dann wird der Inhalt als HTTP-Header in die generierte Nachricht für den Browser geschrieben. Das Attribut muss alle erforderlichen Header, Zeilenvorschübe (CR LF) und den abschließenden doppelten Zeilenvorschub (CR LF CR LF) enthalten. Das Attribut HTTP_HEADER wird nach jeder generierten Seite gelöscht und muss für jede Seite mit speziellen Headern neu gesetzt werden. Ist das Attribut nicht gesetzt, dann generiert WebTransactions folgenden Header: `Content-type; text/html; charset=ISO-8859-1` HTTP_HEADER übersteuert die Attribute CHARSET und HTTP_DEFAULT_HEADER.	template
JAVA_CLASSPATH	Pfad der Java-Klassen	Die Java-Klassendateien mit den Klassenmethoden zur Ausführung innerhalb von WebTransactions werden im Unterverzeichnis java des Basisverzeichnisses erwartet. Soll in einem anderen Verzeichnis gesucht werden, kann der Pfad in diesem Attribut gesetzt werden. Voreinstellung: *basedir*/java	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
JAVA_EXCEPTION	Unterdrückt Fehlermeldungen für nicht abgefangene Ausnahmen	Nicht abgefangene Java-Ausnahmen werden nicht als Fehlermeldungen an den Browser/Benutzer weitergereicht. Wenn Sie das Attribut als Variable vom Typ BOOLEAN definieren, wird es nach fehlerfreier Ausführung der Methoden auf false gesetzt. Sobald eine Ausnahme ausgelöst wird, wird das Attribut auf true gesetzt. Informationen über die Ausnahme werden in der Trace-Datei protokolliert.	template
LANGUAGE	Sprache der Oberfläche	Dieses Attribut wird bei der Suchstrategie für das Template und die Fehlermeldungsdatei ausgewertet. Es wird meist auf der Aufrufseite gesetzt; kann aber jederzeit geändert werden (z.B. durch aktive Auswahl des Benutzers). Siehe auch Abschnitt „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61.	template
LT_REPLACE_STRING	Ersetzen des HTML-Tag-Startzeichens '<'	Dieses Attribut sorgt dafür, dass in allen vom Browser geposteten Daten (Attribute an dem Objekt WT_POSTED) das Zeichen '<' (less than) durch die in LT_REPLACE_STRING enthaltene Zeichenkette ersetzt wird. Dadurch wird verhindert, dass HTML-Tags in dem Inhalt von Eingabefeldern wirksam werden.	template
MAX_NESTING_LEVEL	Maximale Schachtelungstiefe	Mit diesem Attribut steuern Sie die maximale Schachtelungstiefe für einzuschiebende Templates (<wtInclude>, include(), evaluate()) und Funktionsaufrufe. Voreinstellung: 99 Die Funktion forward() setzt die aktuelle Schachtelungstiefe auf 0 zurück.	template
PLATFORM	Systemumgebung von WebTransactions	Mögliche Werte: NT, UNIX, OSD	start

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
POSTED_UNPARSED	nicht aufbereitete Daten vom Browser (Query-String)	WebTransactions stellt die vom Browser empfangenen Daten bzw. den Querystring einer auf WebTransactions verweisenden URL als Objekt WT_POSTED bereit. Wenn Sie die geposteten Daten nicht aufbereitet brauchen, z.B. weil Sie die Daten an einen anderen Web-Server weiterleiten wollen, können Sie dieses Attribut mit einem Namen belegen, der als Wert nicht in den geposteten Daten vorkommen darf. Dann legt WebTransactions unter dem Objekt WT_POSTED ein Objekt mit dem entsprechenden Namen an, in dem die nicht aufbereiteten Daten vom Browser stehen. Beispiel: WT_SYSTEM.POSTED_UNPARSED= “myPost“ Im Objekt myPost stehen dann die empfangenen Name/Wert-Paare vom Browser.	template
PREVENT_EXIT_SESSION	Aufrechterhalten der Sitzung trotz *exitSession()*	Wenn dieses globale Systemattribut existiert, wird die WebTransactions-Sitzung trotz Ausführung der Funktion *exitSession()* nicht beendet. Dieses Attribut ist nützlich beim Debuggen einer Sitzung mit WebLab. Sie müssen nicht die möglicherweise an vielen Stellen stehenden *exitSession()*-Funktionen deaktivieren, um das Beenden der Sitzung zu verhindern. Dadurch bleibt Ihnen der Zugriff auf die Variablen und auf einen gegebenenfalls erstellten SingleStep-Mitschnitt der Sitzung erhalten. Beim Start einer neuen Sitzung wird der gepostete Wert von WT_SYSTEM_PREVENT_EXIT_SESSION nach WT_SYSTEM.PREVENT_EXIT_SESSION übernommen. Somit ist kein Eingriff in die vorhandenen Templates notwendig, allein die mit WebLab übermittelten Startparameter genügen.	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
PROTOCOL	Name des zu verwendenden Host-Adapters	Dieses Attribut wird nur noch aus Kompatibilitätsgründen zur Version 1.0 weiter unterstützt. Ab der Version V2.0 können Sie jeder zu öffnenden Verbindung mit der Anweisung open das Protokoll direkt mitgeben. Der Wert dieser Variablen bestimmt den Host-Adapter, der zum Öffnen einer Verbindung verwendet wird, wenn im Konstruktor WT_Communication der Operand PROTOCOL nicht angegeben ist. Werte: OSD, MVS, UTMV4	template
ROAMING	Wiedereintritt in eine Sitzung	Beim Wiedereintritt in eine Sitzung setzt WebTransactions das Attribut ROAMING auf true. Damit lässt sich in einem Template unterscheiden, ob es sich um eine Neuanmeldung oder die Wiederaufnahme einer laufenden Sitzung handelt. Es wird so möglich, zur erneuten Authentifizierung wiederum das Start-Template zu verwenden und anhand des Werts von ROAMING bei einer Wiederaufnahme z.B. verschiedene Initialisierungen entfallen zu lassen. Siehe auch Abschnitt „Roaming Sessions“ auf Seite 43.	template
ROAMING_FORMAT	Prüfung der Zugriffsberechtigung auf eine Roaming Session	Damit es nicht zu unberechtigten Zugriffen auf eine Sitzung kommen kann, muss die Authentizität des Benutzers mit Hilfe des Attributs ROAMING_FORMAT überprüft werden, wenn bereits eine Sitzung mit der angegebenen Session Id läuft. Dazu muss ROAMING_FORMAT mit dem Namen eines Templates versorgt sein. In diesem Template muss der berechtigte Zugriff auf die Sitzung kontrolliert werden. Siehe auch Abschnitt „Roaming Sessions“ auf Seite 43.	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
SEARCH_HOST_OBJECTS	Suche von Variablen in Hostobjekten	Aus Kompatibilität zur Version 1.0 suchte WebTransactions V2.0 globale Variablen auch immer unter dem voreingestellten Kommunikationsobjekt (s. Attribut HANDLE). Diese Suche wird ab der Version 3.0 standardmäßig nicht mehr durchgeführt. Soll die Suche aus Kompatibilitätsgründen eingeschaltet werden, so ist dieses Attribut auf "YES" zu setzen. Der eingestellte Modus gilt jeweils für einen gesamten Dialogschritt. Eine Änderung des Attributs wirkt sich immer erst für die folgenden Dialogschritte aus Werte: "YES", "NO" Voreinstellung: "NO".	template
SESSION	Session Identifier	Anhand dieses Attributs findet WebTransactions die Task (Prozess/Thread), die der aktuellen Sitzung zugeordnet ist. Dieses Attribut wird zu Beginn der Sitzung erzeugt und auf allen Folgeseiten als "hidden field" mitgeführt. Gleichnamig mit diesem Attribut ist das Verzeichnis unterhalb von *tmp*, in dem temporäre, der Sitzung zugeordnete Dateien abgelegt werden können. Es kann nicht geändert werden.	hidden
SIGNATURE	Unterschrift	Für jede HTML-Seite wird eine Unterschrift erzeugt. Beim Empfang einer Seite mit inkonsistenter Unterschrift wird die Seite mit einer Fehlermeldung abgewiesen.Die Holdertask bleibt in unverändertem Zustand. Das Attribut kann nicht geändert werden.	hidden
STATISTICS	Statistische Informationen über die Sitzung	Das Objekt STATISTICS enthält drei numerische Attribute. BYTES_SENT enthält die Anzahl der in der laufenden Sitzung von WebTransactions an den Browser gesendeten Zeichen, BYTES_RECEIVED die Anzahl der vom Browser empfangenen Zeichen. DIALOG_STEPS enthält die Anzahl der Dialogschritte. Die Daten werden bei jedem Dialogschritt automatisch aktualisiert.	template

Attribut	Bedeutung	Steuerungsmöglichkeit	Einsatz
STYLE	Stil der Oberfläche	Dieses Attribut wird bei der Suchstrategie für das Template ausgewertet. Es bewirkt die Auswahl des Stils, wenn mehrere Sätze von Templates parallel gehalten werden (z.B. mit viel/wenig Grafik, JAVA-Verwendung usw.). Es kann jederzeit geändert werden. Werte: Name des Verzeichnisses *STYLE* (siehe Abschnitt „Suchstrategie“ auf Seite 62)	template
TIMEOUT_APPLICATION	Zeitspanne in Sekunden für Antworten von der WebTransactions-Anwendung	Dieses Attribut gibt an, wie lange maximal auf Antworten auf eine Anfrage an WebTransactions gewartet werden soll, bevor mit einem Timeout die Sitzung beendet wird. Werte: numerischer Wert Voreinstellung 120 sec	start template hidden
TIMEOUT_FORMAT	Template, das bei Timeout der Sitzung noch ausgeführt wird	Dieses Attribut gibt das Template an, das nach Ablauf von TIMEOUT_USER noch ausgeführt wird, bevor die Sitzung beendet wird.	start, template
TIMEOUT_USER	Zeitspanne in Sekunden für Antworten vom Benutzer	Dieses Attribut gibt an, wie lange maximal auf Antwort vom Browser gewartet werden soll, bevor mit einem Timeout die Sitzung beendet wird. Dieses Attribut kann beim Start gesetzt und im weiteren Verlauf geändert werden. Mit NOLIMIT wird der Timer für das Attribut TIMEOUT_USER nicht aktiviert. Werte: numerischer Wert oder NOLIMIT Voreinstellung: 600 Sekunden.	start template
WTML_VERSION	Versionsangabe für WT-Script	Mit der Version 1.2 der Sprache JavaScript hat sich das Verhalten einiger Methoden der eingebauten Klassen geändert. Dieses verbesserte Verhalten wird auch in WebTransactions zur Verfügung gestellt. Wollen Sie für diese Methoden das alte Verhalten einstellen, so setzen Sie den Wert auf 2.0. Werte: 7.5, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0 Voreinstellung: 7.5	template
WWWDOCS_VIRTUAL	virtueller Pfad für Verzeichnis wwwdocs und Basisverzeichnis	Dieses Attribut ist erforderlich, um Ressourcen im Template anzusprechen, die im Verzeichnis wwwdocs stehen. Es wird zur Laufzeit aus der Administration gelesen und kann im weiteren Verlauf einer Sitzung nicht geändert werden.	hidden, start

3.6.4 Posted-Objekt wt_Posted - Daten vom Browser

Im Posted-Objekt werden die Daten, die vom Browser kommen, abgelegt. Es wird im Dialogzyklus nach Empfang der Browserdaten neu aufgebaut. Synchronisierte wie nicht synchronisierte Dialoge (mit dem Browser) beginnen jeweils mit dem Neuaufbau des Posted-Objekts. Damit sind die Daten des Posted-Objekts immer nur im aktuellen Dialogschritt sichtbar und nicht mehr in einem späteren.

Die Attribute des Posted-Objekts und deren Werte entsprechen den Name/Value-Paaren der HTML-Elemente, die vom Browser zurückgeschickt wurden. Dabei bezeichnet *Name* das vom Benutzer manipulierte Objekt/Feld der HTML-Seite und *Value* den dabei entstandenen Wert, der verarbeitet werden soll. Die Attribute können mit dem Namen angesprochen werden, unter dem das Objekt/Feld auf der HTML-Seite definiert wurde.

Das Posted-Objekt ist eine globale Variable vom Datentyp `object`. Die Attribute sind vom Typ `string`. Enthält die Nachricht mehrere Name/Value-Paare mit dem gleichen Namen, wird ein Array dieses Namens angelegt. Die Einträge sind vom Typ `string`. Die toString-Methode (`WT_POSTED.`*`arrayname`*`.toString()`) liefert das erste Attribut zurück.

Beispiel

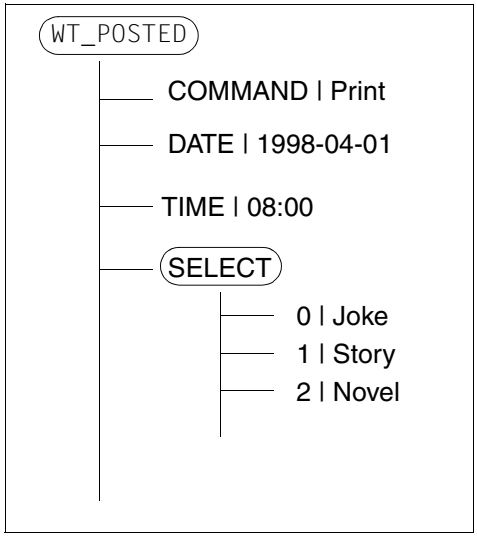
Enthält eine HTML Seite folgende Tags:

```
<input type="text" name="DATE" value="1998-04-01" >
<input type="text" name="TIME" value="08:00" >
<input type="submit" name="COMMAND" value="Print">
<select name="SELECT" multiple>
<option value="Drama">Drama
<option value="Story" selected>Story
<option value="Joke" selected>Joke
<option value="Novel" selected>Novel
<option value="Poem">Poem
</select>
```

Vom Browser werden die folgenden Name/Value-Paare geschickt:

```
COMMAND=Print
TIME=08:00
DATE=1998-04-01
SELECT=Joke
SELECT=Story
SELECT=Novel
```

Dabei entsteht die folgende Objekthierarchie:



Die Rahmen in der Grafik zeigen, dass es sich um Objekte handelt, die wiederum Objekte enthalten.

Das Posted-Objekt kann nur abgefragt werden, und zwar beim Ausführen der onCreate-Scripts (nicht synchronisiert) bzw. der onReceive-Scripts und beim Ausführen des onCreate-Scripts (synchronisiert) des nächsten Templates. Wertzuweisungen werden ignoriert.

Das Posted-Objekt enthält das Objekt `WT_POSTED.HTTP`, unter dem alle HTTP-Header des betreffenden Requests als Attribute angelegt werden. Auch diese Attribute können nur gelesen werden. Um Felder im HTTP-Header der Antwort zu bearbeiten, müssen die Systemobjekt-Attribute `WT_SYSTEM.HTTP_HEADER` (siehe Seite 84) und `WT_SYSTEM.HTTP_DEFAULT_HEADER` verwendet werden (siehe Seite 84).

Mit der Interpretation dieser Browserdaten als Objekt bleibt eine einheitliche Syntax im Template für den Zugriff auf dynamische Daten erhalten.

Beispiel

Gab es auf einer Seite die folgenden drei Knöpfe:

```
<Input Type="SUBMIT" Name="CHOICE" Value="Kunst">
<Input Type="SUBMIT" Name="CHOICE" Value="Unterhaltung">
<Input Type="SUBMIT" Name="CHOICE" Value="Sport">
```

so können Sie die vom Browser zurückgeschickten Werte im nächsten Template folgendermaßen auswerten:

```
Sie haben sich f&uuml;r ##WT_POSTED.CHOICE# entschieden.
```

Bei einem Zugriff auf WebTransactions über die Schnittstelle `WT_REMOTE` wird das Posted-Objekt zu Analysezwecken unterstützt. In WebLab können Sie sich den Steuer- und Datenteil des Posted-Objekts zur Analyse ansehen, der direkte Zugriff auf diese Daten mit Methoden oder Konstruktoren wird nicht empfohlen.

Besondere Attribute an `WT_POSTED`

File upload

Wenn bei WebTransactions ein POST Request eintrifft, dessen Content Type mit `text/` beginnt, wird der Inhalt des Bodys in dem Attribut `BODY` am Objekt `WT_POSTED` gepeichert und steht dort zur weiteren Bearbeitung zur Verfügung. Der Body darf nur druckbare Zeichen enthalten, da ansonsten bei der ersten binären NULL der String als beendet interpretiert wird.

Weitere besondere Attribute

Wenn Sie beim Starten einer Sitzung `/startup` angeben, wird der gepostete Wert einiger Systemobjekt-Attribute `WT_SYSTEM_`*`XXX`* unter `WT_SYSTEM.`*`XXX`* abgelegt, siehe auch Abschnitte „Start-Möglichkeiten“ auf Seite 101 und „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61. Es handelt sich dabei um folgende Attribute:

- `BASEDIR`
- `FORMAT`
- `LANGUAGE`
- `PREVENT_EXIT_SESSION`
- `STYLE`

3.6.5 Host-Wurzelobjekt wt_Host - Verwalten der Verbindungen zu Host-Anwendungen

Alle Objekte, die Informationen über die in einer WebTransactions-Sitzung offenen Verbindungen zu Host-Anwendungen enthalten, werden unter `WT_HOST` zusammengefasst. Für jede Verbindung wird ein eigenes Kommunikationsobjekt angelegt, das die Daten (Host-Datenobjekte, Host-Steuerobjekte, verbindungsspezifisches Systemobjekt) dieser Verbindung enthält.

Sie können in einer WebTransactions-Sitzung mit mehreren gleichzeitig offenen Verbindungen zu verschiedenen Host-Anwendungen und auch Host-Anwendungs-Typen arbeiten.

Das Host-Wurzelobjekt und die Kommunikationsobjekte gibt es nur einmal für jede Sitzung. Änderungen in einer Dialogart (synchronisiert, nicht synchronisiert, remote) sind auch während der jeweils anderen Dialogarten sichtbar.

3.6.5.1 Host-Kommunikationsobjekt wt_Host.*Comobj* - Verwalten einer Host-Anbindung

Da in jeder Sitzung mehrere Verbindungen zu unterschiedlichen Host-Anwendungen sowohl hintereinander als auch parallel geöffnet werden können, muss WebTransactions (und auch der Template-Programmierer) die unterschiedlichen Verbindungen klar unterscheiden können. Deshalb wird für jede Verbindung (explizit oder implizit) ein Kommunikationsobjekt erzeugt. Die Bezeichner für diese Kommunikationsobjekte sind frei wählbar.

Alle Kommunikationsobjekte werden der Übersichtlichkeit halber von WebTransactions unter einer gemeinsamen Wurzel erzeugt, dem Host-Wurzel-Objekt `WT_HOST`.

Ein Host-Kommunikations-Objekt enthält:

- interne Verwaltungsinformationen: Unter anderem die Information, mit welchem Host-Adapter die Verbindung arbeitet. Auf diese Informationen können Sie nicht zugreifen.
- die aktuellen Host-Daten- und Host-Steuerobjekte.
- ggf. das verbindungsspezifische Systemobjekt. Die Attribute dieses Objekts hängen vom Host-Adapter ab und werden in den entsprechenden Handbüchern als Attribute des Systemobjekts beschrieben.

3.6.5.2 Host-Datenobjekte - Daten der Host-Anwendung

Für den Datenaustausch zwischen WebTransactions und Host-Anwendung gibt es Host-Datenobjekte. Sie entsprechen den Daten der Host-Anwendung und speichern die einzelnen Felder eines Bildschirmformats. Diese werden nach dem Empfang einer Host-Nachricht erzeugt und als Attribute des jeweiligen Kommunikationsobjekts abgelegt und stehen dann als Bildschirmabbild zur Verfügung. Sie können dann gelesen, überschrieben und gesendet werden und werden beim Empfang einer neuen Host-Nachricht zerstört.

Für die Darstellung im Browser fließen die Host-Datenobjekte in die HTML-Seite ein. Alle generierten Templates sind so aufgebaut, dass Eingabefelder auf korrespondierende HTML-Oberflächenelemente abgebildet werden
(z.B. `<Input Type="text" ...>`), Ausgabefelder auf reinen HTML-Text.

Senden zur Host-Anwendung und Empfangen von der Host-Anwendung wird durch die Methoden `send` und `receive` der Klasse `WTCommunication` ausgelöst, die im Sprachumfang von WTScript enthalten sind. Sie können beim Generieren der HTML-Seite (Create-Zeitpunkt) oder nach Empfang der Daten vom Browser (Receive-Zeitpunkt) vorkommen. Die Lebensdauer von Host-Objekten ist daher nicht an den Dialogzyklus gebunden. Es können während eines Dialogzyklus, ja sogar in einer Phase des Dialogzyklus (HTML-Generierung oder receive-Methoden-Ausführung) mehrere Generationen von Host-Objekten entstehen und wieder verschwinden, es kann aber auch eine Generation über mehrere Dialogzyklen bestehen bleiben.

Host-Objekte existieren nur, während eine Verbindung zur Host-Anwendung offen ist (siehe WebTransactions-Handbuch „Template-Sprache“, Methode open).

Da die Host-Adapter Host-Objekte unterhalb des entsprechenden Kommunikationsobjekts anlegen, können parallel gleichnamige Objekte existieren. Die Namen sind dadurch eindeutig, dass sie die Host-Objekte mit ihrem vollqualifizierten Namen ansprechen, also mit dem Pfad des Kommunikationsobjekts (`WT_HOST.`*Comobj*`.Host_object.attribute`).

Die genaue Ausprägung der Host-Objekte sowie deren Namensgebung hängt vom jeweiligen Host-Adapter ab und ist im Handbuch für die jeweilige Host-Anbindung beschrieben.

3.6.5.3 Host-Steuer-Objekte - Verwaltungsdaten für ein Format

Zur Steuerung der Host-Anbindung gibt es Host-Steuer-Objekte, die für die gesamte Lebensdauer einer Verbindung existieren. Diese liefern Informationen, die nicht nur ein einzelnes Feld betreffen, sondern das gesamte Bildschirmformat. Dazu gehören z.B. das Feld, in dem sich der Cursor befindet, und die Reihenfolge der Felder eines Bildschirms.

Die genaue Ausprägung der Host-Steuer-Objekte sowie deren Namensgebung hängen vom jeweiligen Host-Adapter ab und ist im Handbuch für die jeweilige Host-Anbindung beschrieben.

3.6.5.4 Verbindungsspezifisches Systemobjekt wt_Host.*Comobj*.wt_System - verbindungsspezifische Steuerungsfunktionen

Die Attribute des verbindungsspezifischen Systemobjekts steuern die Verbindung zu einer Host-Anwendung. Sie sind spezifisch für den verwendeten Host-Adapter und werden in den entsprechenden Handbüchern beschrieben.

Das verbindungsspezifische Systemobjekt wird erzeugt, wenn das Kommunikationsobjekt mit dem Konstruktor der Klasse `WT_Communication` erzeugt wird. Da es sich um ein ganz normales Objekt handelt, kann es innerhalb eines WTScripts mit dem `delete`-Operator wieder gelöscht werden.

Existiert das verbindungsspezifische Systemobjekt, so greift WebTransactions für die Steuerung der Verbindung nur auf die Attribute dieses Systemobjekts zu; gleichnamige Attribute am globalen Systemobjekt werden ignoriert.

3.6.5.5 WTScript und Kommunikations-Objekte

Die folgende Abbildung zeigt anhand eines Beispiels, mit welchen WTScript-Anweisungen Sie eine Verbindung zu einer Host-Anwendung herstellen und welche Objekte dabei von WebTransactions angelegt werden:

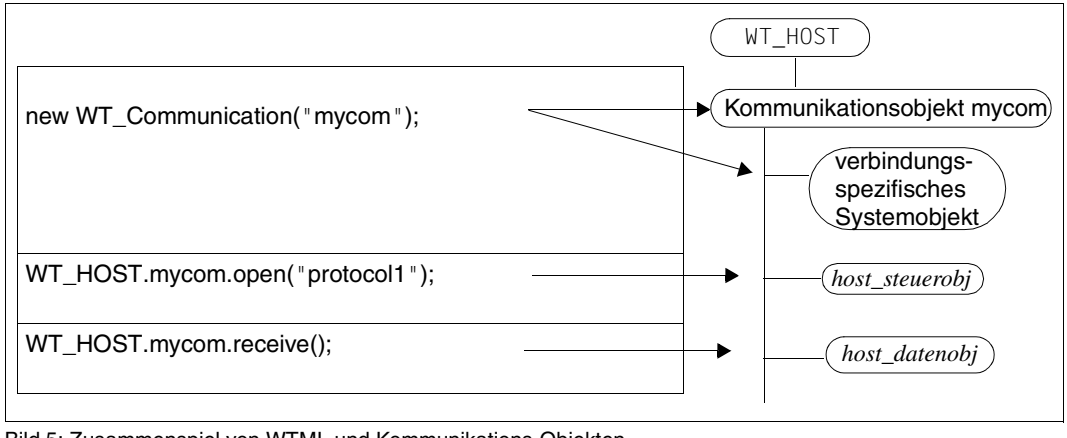


Bild 5: Zusammenspiel von WTML und Kommunikations-Objekten

Der Konstruktoraufruf von `WT_Communication` legt unter `WT_HOST` ein Kommunikationsobjekt und darunter ein verbindungsspezifisches Systemobjekt an. Bei diesem Aufruf legen Sie außerdem den Namen des Kommunikationsobjekts fest.

Alle weiteren Aktionen mit dieser Verbindung rufen Sie als Methoden an diesem Kommunikationsobjekt auf.

Mit dem Methodenaufruf `open()` wird eine Verbindung geöffnet. Bei diesen Aufrufen bestimmen Sie auch, über welchen Host-Adapter die Verbindung hergestellt wird. Bei diesem Aufruf entstehen die Host-Steuer-Objekte.

Mit `receive` werden Daten von der Host-Anwendung empfangen. Bei diesen Aufrufen entstehen die Host-Daten-Objekte.

3.6.6 Template-Objekte – Kurzfristige Zwischenspeicher

Template-Objekte (auch Template-Variablen) dienen dem Template-Programmierer als kurzfristige Speicherungsmöglichkeit für Daten, die in WTScript-Bereichen, WTML-Tags oder Auswertungsoperatoren eines Templates zur Zwischenspeicherung benötigt werden.

Sie können an beliebiger Stelle in diesen Bereichen im Template definiert werden und existieren nach ihrer Definition bis zum Ende des Dialogschritts (synchronisiert oder nicht synchronisiert), bis zum Sitzungsende (remote) oder bis zum expliziten Löschen. Die Template-Objekte der verschiedenen Dialogarten (synchronisiert, nicht synchronisiert und remote) werden getrennt verwaltet und beeinflussen sich gegenseitig nicht.

Template-Objekte werden einfach durch Verwendung eines Namens (Bezeichner, der keinem vordefinierten WTML-Objekt entspricht) in einer Anweisung oder Zuweisung angelegt und können alle in WebTransactions verfügbaren Variablen-Typen annehmen (Näheres dazu im WebTransactions-Handbuch „Template-Sprache“).

Beispiel

```
<wtonCreateScript>                          (1)
a = 5;
b = a + 1;                                  (2)
</wtonCreateScript>
##a# ist um eins kleiner als ##b#           (3)
```

(1) Im Script-Bereich legen Sie das Template-Objekt `a` an und weisen ihm den Wert 5 zu.

(2) Sie addieren den Wert 1 zu dem Wert von `a` und legen das Ergebnis in die Template-Variable `b`, die Sie damit definieren.

(3) Im HTML-Bereich ersetzen Sie mit Hilfe des Auswertungsoperators die Template-Objekte `a` und `b` durch aktuelle Werte.

Wenn Sie diese Anweisungen mit WebTransactions ausführen, enthält die HTML-Seite den Text:

```
5 ist um eins kleiner als 6
```

Template-Objekte können auch selbst wieder Behälter für weitere Objekte/Attribute sein.

So legen Sie z.B mit folgendem WTScript das Template-Objekt `x` an und weisen ihm ein neues Attribut `y` mit dem Wert 1 zu:

```
<wtonCreateScript>
x = new Object();
x.y = 1;
</wtonCreateScript>
```

4 Ablauf einer WebTransactions-Anwendung

Eine WebTransactions-Anwendung ist der Vermittler zwischen Dialog-Anwendungen auf einem Host und der grafischen Oberfläche im Browser. Auf eine WebTransactions-Anwendung kann aber auch über ein Client-Programm und die WT_REMOTE-Schnittstelle zugegriffen werden. Beide Schnittstellen für den Zugriff auf WebTransactions sind in diesem Kapitel beschrieben.

Der Schwerpunkt liegt in diesem Kapitel auf der Arbeit mit der Dialogschnittstelle und den Steuerungsmöglichkeiten für den Template-Programmierer. Die Beschreibung der Dialogschnittstelle umfasst

- die Startmöglichkeiten für eine WebTransactions-Anwendung: Direkte Eingabe der URL, Link oder Formular. Außerdem werden die Angaben, die Sie auf einer Aufrufseite verarbeiten können, beschrieben.
- den Datenaustausch während der Sitzung: HTML-Formular und HTML-Link
- die Steuerungsmöglichkeiten einer WebTransactions-Anwendung: Passiv über die Host-Anwendung oder aktiv über das Template
- den synchronisierten und nicht synchronisierten Dialog
- das Ende einer Sitzung
- Diagnosemöglichkeiten
- den Transfer einer WebTransactions-Anwendung

Danach wird kurz die Client-Schnittstelle WT_REMOTE vorgestellt.

Abschließend wird beschrieben, wie Sie eine WebTransactions-Anwendung vom Browser aus administrieren können.

4.1 Erstellen einer WebTransactions-Anwendung

Um Ihre Host-Anwendung an das WWW anzubinden, gehen Sie folgendermaßen vor:

1. Installieren Sie die erforderliche Liefereinheit von WebTransactions auf Ihrem Integrations-Server. Die Installation der einzelnen Liefereinheit ist im jeweiligen Handbuch beschrieben.
2. Mit jeder Liefereinheit von WebTransactions werden ein Administrationsprogramm und die Entwicklungsumgebung WebLab ausgeliefert, mit der Sie alle weiteren Schritte der Integration ausführen können. Das Administrationsprogramm ist im Kapitel „WebTransactions-Server“ auf Seite 141 beschrieben.
3. Starten Sie WebLab mit dem Befehl **Start/Programme/WebTransactions 7.5/ WebLab**. Das Hauptfenster von WebLab wird am Bildschirm eingeblendet.
4. Wenn Sie zum erstenmal mit WebLab arbeiten, stellen Sie mit dem Befehl **Optionen/Einstellungen/Programme** ein, mit welchem Browser Sie arbeiten wollen.
5. Wählen Sie den Befehl **Administration/Server**, um zuerst das Administrationsprogramm zu starten.
6. Melden Sie sich beim Administrationsprogramm als Benutzer `admin` an. Wenn Sie das Administrationsprogramm zum ersten Mal aufrufen, vergeben Sie ein Passwort für den Benutzer `admin`.
7. Fordern Sie dann die benötigte Anzahl Lizenzen an und tragen Sie sie ein.
8. Legen Sie für jeden Entwickler, der WebTransactions-Anwendungen bearbeiten soll, einen WebTransactions-Benutzer (User) in der Administration an.
9. Legen Sie dann einen Pool für Ihre Basisverzeichnisse an und weisen Sie dem Benutzer `admin` oder auch anderen Benutzern die Zugriffsberechtigung auf diesen Pool zu.
10. Speichern Sie die neue Konfiguration.
11. Beenden Sie das Administrationsprogramm und schließen Sie den Browser.
12. Erzeugen Sie in WebLab mit dem Befehl **Projekt**/**Neu** ein neues Projekt und ein neues Basisverzeichnis. Im Dialogfeld **Basisverzeichnis erzeugen** wählen Sie einen oder mehrere der angebotenen Host-Adapter aus.

13. Erstellen Sie ein individuelles Start-Template. Geben Sie auf der Registerkarte **Verbindungsparameter** die Daten Ihrer Host-Anwendung ein.

    Wenn Sie eine Host-Anwendung mit WebTransactions for openUTM anbinden wollen müssen Sie zusätzlich folgende Schritte ausführen:

    – Loggen Sie sich am Host-Rechner unter der Kennung ein, unter der die Host-Anwendung abgelegt ist, die Sie integrieren wollen.

    – Erstellen Sie mit dem Dienstprogramm `IFG2FLD` aus der IFG-/FHS-Bibliothek die Formatbeschreibungsquellen, aus denen mit WebLab die Templates erzeugt werden. Das Dienstprogramm `IFG2FLD` ist im WebTransactions-Handbuch „Anschluss an openUTM-Anwendungen über UPIC“ beschrieben.

    – Übertragen Sie die Formatbeschreibungsquellen im Textmodus auf den Rechner, auf dem WebLab läuft.

    – Wählen Sie in WebLab den Befehl **Generieren/Templates/aus einer IFG-Bibliothek**, um aus den Formatbeschreibungsquellen die Templates für die Umsetzung zu generieren.

14. Stellen Sie dann mit dem Befehl **Datei/Sitzung starten** mit den Parametern im Dialogfeld **Sitzung starten** eine Verbindung über WebTransactions zur Host-Anwendung her. Geben Sie als Start-Template das erzeugte individuelle Start-Template an. Die Parameter des Dialogfelds **Sitzung starten** sind im Abschnitt „Sitzung starten“ auf Seite 174 und in der Online-Hilfe beschrieben.

    Nach Bestätigen dieses Dialogfeldes wird der eingestellte Browser auf dem Entwicklungsrechner gestartet. Das individuelle Start-Template baut die Verbindung zur Host-Anwendung auf und zeigt das erste Format im Browser an. Je nach Konfiguration der Host-Anwendung kann dies bereits das Start-Format der Host-Anwendung sein (z.B. openUTM) oder eine Emulationsdarstellung (z.B. bei `$Dialog`), in der Sie die Host-Anwendung erst starten.

Mit diesen Schritten haben Sie sowohl Ihre Host-Anwendung an das WWW angebunden als auch schon eine WebTransactions-Anwendung erstellt und gestartet.

Eine ausführlichere Beschreibung zu diesem Thema entnehmen Sie dem Kapitel „Die Entwicklungsumgebung WebLab“ auf Seite 167.

4.2 WebTransactions-Dialoganwendung starten

Eine WebTransactions-Anwendung startet der Benutzer entweder direkt über einen URL oder über eine Aufrufseite am Browser. Auf dieser Aufrufseite weist ein Link oder ein Formular auf das erste Template der WebTransactions-Anwendung, das so genannte Start-Template. Mit dem Start-Template wird die WebTransactions-Anwendung eigentlich erst gestartet. Das folgende Bild zeigt eine schematische Darstellung des Starts einer WebTransactions-Anwendung über eine Aufrufseite.

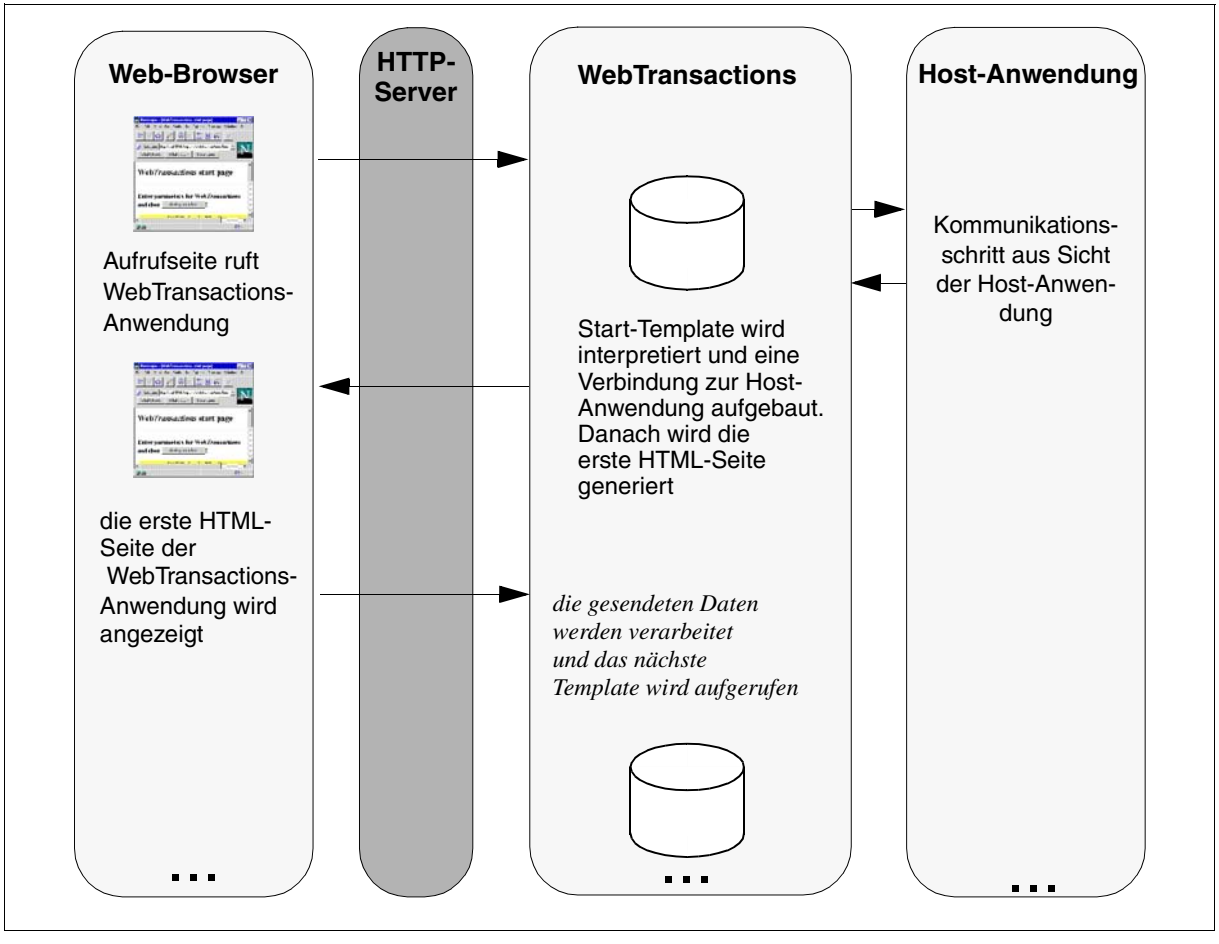


Bild 6: Start einer WebTransactions-Anwendung mit Aufrufseite

Beim Erstellen eines Start-Templates werden Sie in WebLab mit einem WTBean unterstützt. Die Aufrufseite für den Benutzer am Browser müssen Sie selbst erstellen. Neben der Festlegung von Basisverzeichnis und Start-Template für die WebTransactions-Sitzung können auch vor Beginn der Sitzung weitere Daten am Browser erfasst und bereits im ersten Schritt an WebTransactions geschickt werden.

Beachten Sie, dass die Aufrufseite für eine WebTransactions-Anwendung im Dokumentenverzeichnis des Web-Servers liegen muss, damit sie vom Web-Server gefunden wird. Es wird empfohlen, die Aufrufseite im Verzeichnis `wwwdocs/html` abzulegen. Dies hat den Vorteil, dass die Datei beim Transfer der WebTransactions-Anwendung berücksichtigt wird und die Pfade der enthaltenen Programmaufrufe (wtPublish, wtCluster) dabei angepasst werden.

4.2.1 Start-Möglichkeiten

Jede Sitzung beginnt mit einer Benutzereingabe am Browser. Hierfür gibt es verschiedene Möglichkeiten:

– Angabe der URL-Adresse einer WebTransactions-Anwendung im Browser oder Aktivierung eines Links

– Ausfüllen und Abschicken eines Formulars

– Starten über die Schnittstelle `WT_REMOTE`

Die ersten beiden Startarten lassen sich mit WebLab simulieren: Die Angaben zum Start werden dann nicht im Web-Browser, sondern in WebLab gemacht.

Auf allen von WebTransactions unterstützten WebTransactions-Plattformen wird die CGI-Schnittstelle zum HTTP-Server unterstützt. Das CGI-Programm, das Sie in der URL angeben, heißt `WTPublish.exe`. Zusätzlich wird auf einer Windows-Plattform ein Kopplungsbaustein für die ISAPI-Schnittstelle ausgeliefert (`WTPublishISAPI.dll`). Diesen können Sie auf HTTP-Servern einsetzen, welche die ISAPI-Schnittstelle unterstützen (z.B. MS Internet Information Server). Da ISAPI gegenüber CGI die Performance erheblich verbessert, empfiehlt es sich, ISAPI einzusetzen.

Wenn Sie die ISAPI-Schnittstelle zu WebTransactions verwenden, müssen Sie in der entsprechenden URL `WTPublish.exe` durch `WTPublishISAPI.dll` ersetzen.

4.2.1.1 Starten durch Eingabe der URL

Die URL zum Aufruf von WebTransactions ist folgendermaßen aufgebaut:

`http[s]://`*machine*`/`*cgiPath*`/WTPublish.exe/`*basedir*`?`*startTemplate*`[.htm]`

Sie startet WebTransactions auf der angegebenen Maschine in dem entsprechenden Basisverzeichnis.

machine
: Internetadresse bzw. der symbolische Name des Rechners, auf dem WebTransactions installiert ist (ggf. mit Portnummer für den HTTP-Server).

cgiPath
: für den dortigen HTTP-Server vereinbarter Pfad (Präfix) für CGI-Programme.

basedir
: Basisverzeichnis (basedirectory), unter dem die WebTransactions-Anwendung installiert ist.
basedir ist ein absoluter Pfad (auf Windows mit Laufwerksbezeichner)

startTemplate
: erstes Template, das ausgeführt wird. Das Suffix `htm` kann entfallen. Zur Generierung der ersten Seite wird das Template *startTemplate* im Stil `forms` verwendet, also die Datei:

  basedir`/config/forms/`*startTemplate*`[.htm]`

Bei dieser Methode können im ersten Schritt keine weiteren Nutzdaten an WebTransactions geschickt werden.

Diese URL kann direkt im Browser eingegeben werden oder in einem Link – z.B. auf einer Aufrufseite – stehen:

`<a href="http[s]://`*machine/cgiPath/*`WTPublish.exe/`*basedir*`?`*startTemplate*`[.htm]">`
Text oder Bild zum Starten von WebTransactions`</a>`

Starten mit zusätzlichen Werten

Wenn Sie beim Starten zusätzliche Werte mitgeben wollen, muss die URL folgendermaßen aufgebaut sein:

```
http[s]://machine]/cgiPath/WTPublish.exe/startup?
WT_SYSTEM_BASEDIR=basedir&WT_SYSTEM_FORMAT=startTemplate[.htm]
&myData=data...
```

Im Gegensatz zur vorherigen Methode werden Basisverzeichnis und Start-Template als Name/Value-Paare angegeben. Als Namen müssen Sie `WT_SYSTEM_BASEDIR` und `WT_SYSTEM_FORMAT` verwenden. Bei dieser Methode können weitere Name/Value-Paare in den `QUERY_STRING` aufgenommen werden. Diese stellt WebTransactions im ersten Dialogzyklus als Attribute des Posted-Objekts (`WT_POSTED`) zur Verfügung.

Die Namen der Attribute entsprechen dem Namen der Name/Value-Paare. Wenn Ihre URL beispielsweise ein Paar `&Benutzer=Mustermann` enthält, können Sie auf den Wert dieses Paares im Start-Template über `WT_POSTED.Benutzer` zugreifen.

Starten in einem anderen Stil und einer anderen Sprache

Ein spezieller Fall für das Starten mit zusätzlichen Werten ist das Starten einer Sitzung in einer anderen Sprache oder einem anderen Stil: Beide können über bestimmte Name-/Value-Paare festgelegt werden.

```
http[s]://machine/cgiPath/WTPublish.exe/startup/WT_SYSTEM_BASEDIR=
/basedir?WT_SYSTEM_FORMAT=startTemplate&WT_SYSTEM_LANGUAGE=sprache&
WT_SYSTEM_STYLE=stil
```

Dabei lautet der Name für das bevorzugte Stilverzeichnis `WT_SYSTEM_STYLE`, der für das bevorzugte Sprachverzeichnis `WT_SYSTEM_LANGUAGE`.

Eine genaue Beschreibung, wie WebTransactions die zu verwendenden Templates sucht, finden Sie im Abschnitt „Unterverzeichnisse für Stil- und Sprachvarianten“ auf Seite 61.

Wenn Sie eine Sitzung auf diese Weise starten, wird bereits das Start-Template im angegebenen Stil und/oder der angegebenen Sprache gesucht.

Starten auf einem Cluster-Member

Eine Sitzung auf einem WebTransactions-Cluster wird über folgenden Link gestartet, siehe hierzu auch Abschnitt „Cluster-Sitzung starten“ auf Seite 158:

```
http[s]://machine/cgiPath/WTCluster.exe/cluster-id[?WT_SYSTEM_FORMAT
=startTemplate[&param2....]]
```

4.2.1.2 Starten durch HTML-Formular

Ein HTML-Formular zum Aufruf von WebTransactions ist folgendermaßen aufgebaut:

```
<FORM METHOD="POST"
ACTION="[http[s]://machine]/cgiPath/WTPublish.exe/basedir?startTemplate[.htm]">
   additional text and data
</FORM>
```

machine
: Internetadresse bzw. der symbolische Name des Rechners, auf dem WebTransactions installiert ist (ggf. mit Portnummer für den HTTP-Server). Dieser Teil wird vom Browser ergänzt und kann daher entfallen, wenn die Aufrufseite, die diesen Link enthält, von der gleichen Maschine geladen wurde

cgiPath
: für den dortigen HTTP-Server vereinbarter Pfad (Präfix) für CGI-Programme.

basedir
: Basisverzeichnis (basedirectory), unter dem die WebTransactions-Anwendung installiert ist.
basedir ist ein absoluter Pfad (auf Windows mit Laufwerksbezeichner)

startTemplate
: erstes Template, das ausgeführt wird. Das Suffix `htm` kann entfallen. Zur Generierung der ersten Seite wird das Template *startTemplate* im Stil `forms` verwendet, also die Datei:

  basedir`/config/forms/`*startTemplate*`[.htm]`

Innerhalb des Formulars können weitere HTML-Eingabeelemente (z.B. Eingabefelder, Auswahllisten) angegeben werden, die bei dieser Methode im ersten Dialogzyklus an WebTransactions geschickt werden. Diese Daten stehen dann bei der Verarbeitung des Start-Templates als Attribute des Objekts `WT_POSTED` zur Verfügung.

Die Namen der Attribute entsprechen den Namen der Eingabeelemente. Wenn Ihr Formular z.B. ein Eingabefeld `<input type=text name=Benutzer>` enthält, können Sie auf den Wert dieses Feldes im Start-Template über `WT_POSTED.Benutzer` zugreifen.

Starten in einem anderen Stil und einer anderen Sprache

Wenn Sie Ihre WebTransactions-Anwendung bereits mit einem Start-Template aus einem anderen Stil oder einer anderen Sprache beginnen wollen, verwenden Sie folgendes Formular:

```
<FORM METHOD="POST"
ACTION="[http[s]://machine]/cgiPath/WTPublish.exe/startup]">
<input type="hidden" name="WT_SYSTEM_FORMAT" value="startTemplate">
<input type="hidden" name="WT_SYSTEM_BASEDIR" value="basedir">
<input type="hidden" name="WT_SYSTEM_STYLE" value="mystyle">
<input type="hidden" name="WT_SYSTEM_LANGUAGE" value="mylanguage">
additional text and data
</FORM>
```

Im Gegensatz zur vorherigen Methode werden Basisverzeichnis und Start-Template in versteckten Eingabefeldern angegeben. Verwenden Sie als Namen der Eingabefelder `WT_SYSTEM_BASEDIR` und `WT_SYSTEM_FORMAT`.

Starten in einem Cluster-Member

Für den Start einer Sitzung auf einem Cluster über ein Formular müssen Sie die Methode GET verwenden, siehe hierzu auch Abschnitt „Cluster-Konzept“ auf Seite 153:

```
<form method="get"
      action="http[s]://machine/cgiPath/WTCluster.exe/cluster-id">
      [<input type="hidden" name="WT_SYSTEM_FORMAT" VALUE="startTemplate">]
additional text and data
</form>
```

4.2.1.3 Starten über WT_REMOTE

Ein Client-Programm startet eine Sitzung zur exklusiven Nutzung über die Schnittstelle `WT_REMOTE` mit der Methode `START_SESSION`.

Die Client-Programmierung ist ausführlich im WebTransactions-Handbuch „Client-APIs für WebTransactions“ beschrieben.

4.2.2 Templates beim Starten

Beim Starten können Sie einerseits die mitgelieferten Start-Templates einsetzen, andererseits haben Sie die Möglichkeit sich anwendungsspezifische Start-Templates zu erzeugen. Dieser Abschnitt beschreibt die mitgelieferten Start-Templates und zeigt auf, welche Verzweigungsmöglichkeiten in den einzelnen Start-Templates angeboten werden.

4.2.2.1 Allgemeines Start-Template wtstart.htm

Für den Test der WebTransactions-Anwendung werden spezielle Start-Templates mit ausgeliefert. Über diese Start-Templates können Sie die Parameter für die Host-Anwendungen setzen und beliebig viele Host-Anwendungen parallel starten.

Mit dem Start-Template `wtstart.htm`, das von WebTransactions ausgeliefert wird, können Sie

– eine Verbindung zu einer beliebigen Host-Anwendung herstellen

– Verbindung zu mehreren Host-Anwendungen gleichzeitig aufnehmen.

Sie legen auf dieser HTML-Seite die globalen Parameter für Ihre WebTransactions-Sitzung fest. Beim Öffnen der Verbindung wird auf die verbindungsspezifische Aufrufseite verzweigt, auf der Sie die speziellen Parameter für die jeweilige Verbindung eingeben.

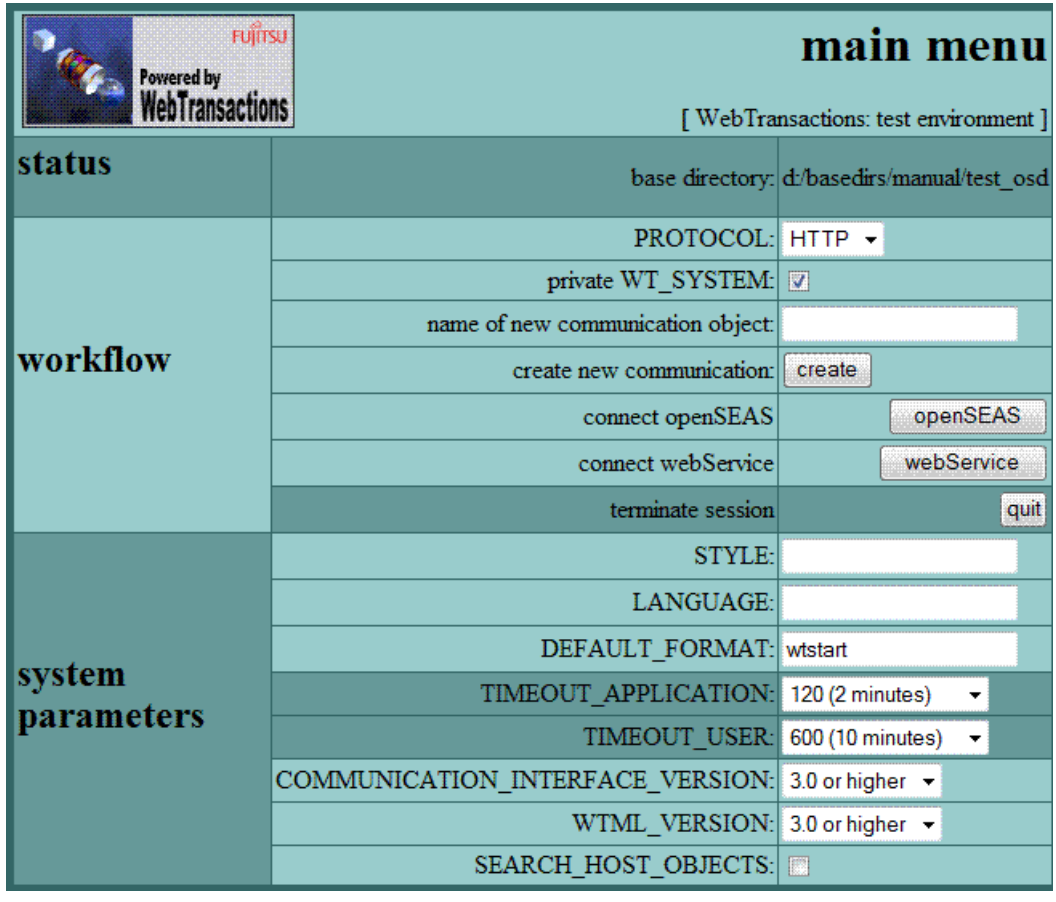
Für eine Produktivanwendung werden Sie i.A. ein eigenes Start-Template erstellen, das die für Ihre WebTransactions-Anwendung wichtigen Daten automatisch ermittelt und ein Minimum an Benutzereingaben benötigt. Beim Erstellen eines Start-Templates werden Sie in WebLab durch ein WTBean unterstützt.

Für das Arbeiten mit mehreren Verbindungen zeigen Ihnen die Beispiele zur Anwendungsintegration (siehe hierzu auch Abschnitt „Zusammenspiel der Start-Templates bei Anwendungsintegration“ auf Seite 111), wie Sie vorgehen können.

Mit dem Host-Adapter UTMV4 können nicht zwei Verbindungen gleichzeitig geöffnet werden. Sie können parallele Verbindungen zu *open*UTM-Anwendungen über den Host-Adapter OSD abwickeln.

`wtstart.htm` wird beim Erzeugen eines Basisverzeichnisses im Unterverzeichnis `config\forms` angelegt. Dieses Start-Template kann zu Beginn einer WebTransactions-Sitzung angegeben werden.

Die folgende Abbildung zeigt die HTML-Oberfläche von `wtstart.htm`:



Alle System-Parameter sind Attribute des Systemobjekts und sind im Abschnitt „Globales Systemobjekt wt_System - Sitzungssteuerung und langfristige Datenspeicherung“ auf Seite 77 beschrieben.

PROTOCOL
Geben Sie den Typ der Host-Anwendung bzw. den Host-Adapter an.

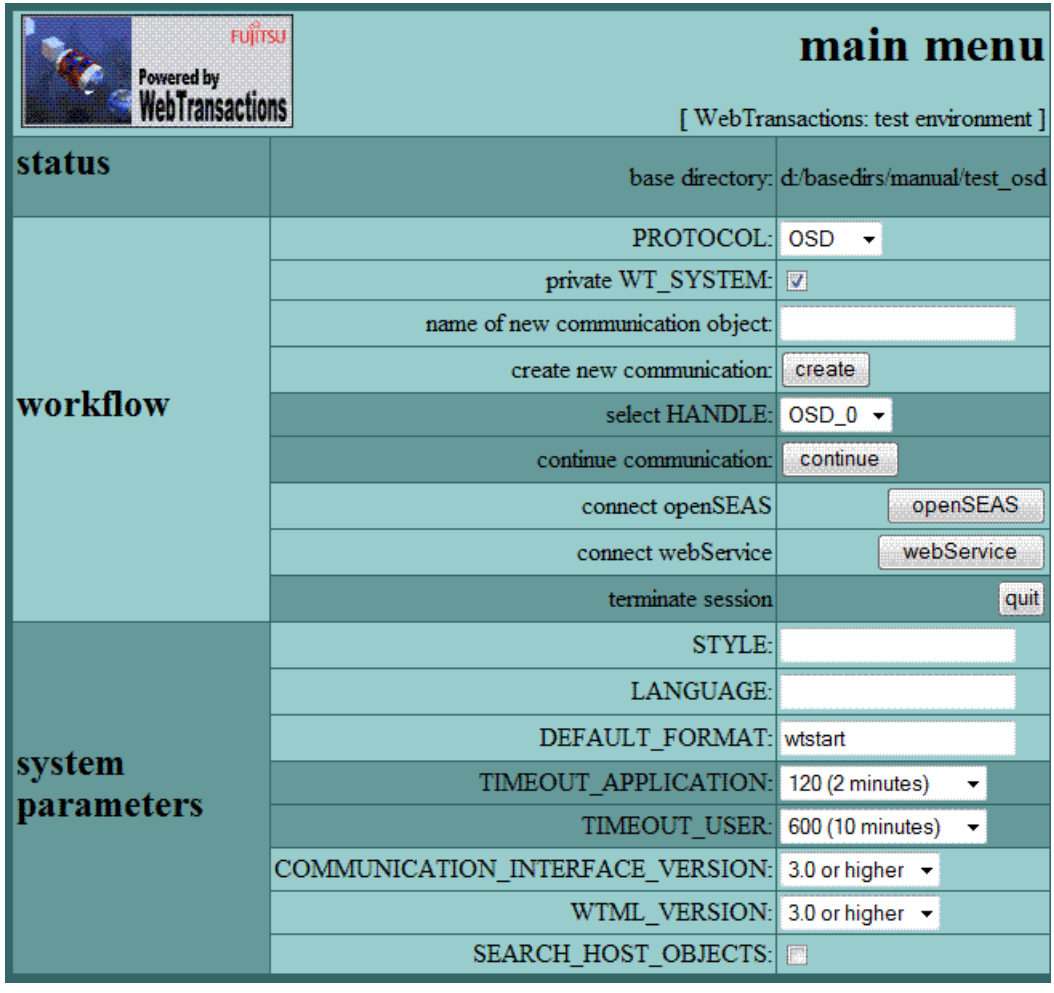
private WT_SYSTEM
Die Option für ein verbindungsspezifisches Systemobjekt unter dem Kommunikationsobjekt ist standardmäßig aktiviert.

name of new communication object
Sie können für jedes Kommunikationsobjekt einen Namen vergeben und das Kommunikationsobjekt im Template auch über diesen Namen ansprechen. Der Name, den Sie hier angeben, muss mit dem Namen des Kommunikationsobjekts in den verwendeten Templates übereinstimmen.

connect webService
Wählen Sie diesen Knopf, wenn Sie ein Web-Frontend für einen Web-Service starten wollen. Im Browser wird das Template `wtconnectWebService.htm` eingeblendet. Dort können Sie die für Web-Services generierten Templates einzeln auswählen.

create new communication
Mit diesem Knopf legen Sie ein neues Kommunikationsobjekt an. Das protokollspezifische Start-Template (`wtstart*.htm`) wird im Browser eingeblendet, in dem Sie die Parameter für die Verbindung zum Host-Rechner eingeben können. Das Zeichen `*` in `wtstart*.htm` steht für den jeweiligen Host-Adapter. Wenn Sie danach im Template `wtstart*.htm` auf den Knopf **goto main menu** klicken, wird wieder `wtstart.htm` eingeblendet, damit Sie ein weiteres Kommunikationsobjekt und damit eine weitere parallele Verbindung anlegen können.

Wenn Sie bereits eine Verbindung gestartet haben und diese z.B. mit dem Knopf **Suspend** unterbrechen, wird ein leicht modifiziertes `wtstart.htm` ausgegeben. Auch dann können Sie weitere Kommunikationsobjekte anlegen.



select HANDLE

Wählen Sie aus der Liste den Namen eines Kommunikationsobjekts (HANDLE) für eine bereits geöffnete Verbindung zu einer Host-Anwendung.

continue communication

Setzen Sie die unter `select HANDLE` ausgewählte Verbindung fort.

`wtstart.htm` verzweigt auf ein verbindungsspezifisches Start-Template, wenn eine Verbindung hergestellt oder der Dialog mit einer Verbindung fortgesetzt wird.

4.2.2.2 Verbindungsspezifische Start-Templates

Es gibt folgende verbindungsspezifische Start-Templates:

- `wtstartHTTP.htm`
- `wtstartUTMV4.htm`
- `wtstartOSD.htm`
- `wtstartMVS.htm`
- `wtconnectOpenSEAS.htm`
- `wtconnectWebServices.htm`

Diese werden beim Erzeugen eines Basisverzeichnisses im Unterverzeichnis `config\forms` angelegt. In diesen speziellen Start-Templates setzen Sie die Start-Parameter und nehmen den Dialog mit der Host-Anwendung auf. Diese Start-Templates sind in den Handbüchern des jeweiligen Host-Adapters beschrieben. `wtstart.htm` verzweigt auf eines dieser Start-Templates, wenn eine Verbindung hergestellt oder der Dialog mit einer Verbindung fortgesetzt wird.

4.2.2.3 Zusammenspiel der Start-Templates bei Anwendungsintegration

Nachdem Sie auf der Aufrufseite eine WebTransactions-Anwendung aufgerufen haben, wird eine WebTransactions-Sitzung gestartet (siehe Abschnitt „WebTransactions-Dialoganwendung starten“ auf Seite 100). Die folgende Grafik zeigt Ihnen die Verzweigungmöglichkeiten, die in den einzelnen Start-Templates angeboten werden:

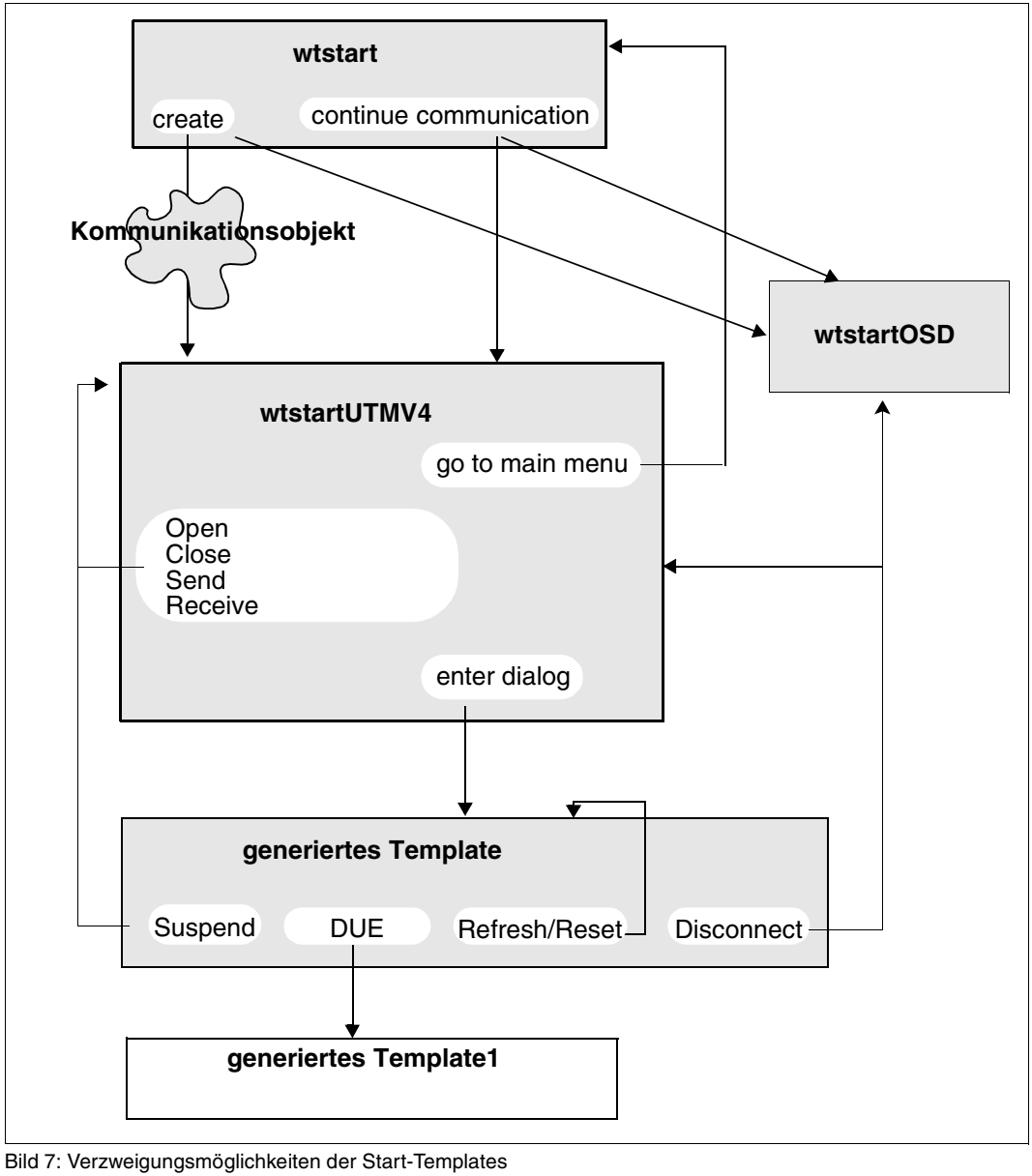


Bild 7: Verzweigungsmöglichkeiten der Start-Templates

`wtstart.htm` legt ein Kommunikationsobjekt `WT_HOST.`*Comobj* für verschiedene Host-Anwendungen an. Dies geschieht durch Auswählen des Knopfs **create**. Dieser löst den Konstruktoraufruf des Kommunikationsobjekts `new WT_Communication` aus (siehe WebTransactions-Handbuch „Template-Sprache“).

Kommt der Endbenutzer ins allgemeine Start-Template zurück, so kann er mit dem Knopf **continue communication** andere bestehende Verbindungen fortsetzen. Dann wird `WT_SYSTEM.FORMAT` mit `wtstart*.htm` versorgt und das verbindungsspezifische Start-Template ausgegeben.

In den verbindungsspezifischen Start-Templates (siehe Benutzerhandbücher zu den einzelnen Host-Adaptern) können Sie mit **go to main menu** zum allgemeinen Start-Template zurückkehren. Diese Option müssen Sie auswählen, wenn Sie eine Verbindung zu einer weiteren Host-Anwendung herstellen. Zu bereits bestehenden Verbindungen können Sie auch direkt über **go to <*commobj*>** wechseln.

Darüberhinaus können Sie Aktionen mit der Verbindung auslösen (Öffnen der Verbindung, Senden von Daten, Empfangen von Daten...). Insbesondere können Sie in den Dialog mit der Verbindung einsteigen. Sie gelangen dann zu einem Template der entsprechenden Host-Anwendung. Wenn Sie mit den generierten Templates von WebTransactions arbeiten, steht Ihnen auf diesen Seiten die Tastaturbelegung als Knopfleiste zur Verfügung.

Zusätzlich zur regulären Tastaturbelegung stehen Ihnen noch folgende Knöpfe in der Knopfleiste zur Verfügung:

– **Reset** zum Zurücksetzen der Eingabefelder. Dabei findet kein Kontakt zur Host-Anwendung statt.

– **Refresh** zum Neuaufbau der Darstellung am Bildschirm

– **Disconnect** zum Beenden der Verbindung zur Host-Anwendung

– **Suspend** zum Unterbrechen des Dialogs mit einer Host-Anwendung (nur wenn das Attribut `WT_SYSTEM.DIALOG_CONTROL_FORMAT` vorhanden ist)

4.3 Datenaustausch während der Sitzung

Nach dem Start einer WebTransactions-Anwendung stellen Sie dem Benutzer im Browser Elemente für Eingaben, Auswahlmöglichkeiten usw. zur Verfügung. Die Werte dieser Benutzer-Reaktionen müssen vom Browser an die WebTransactions-Sitzung zurückgeschickt werden. HTML bietet dafür zwei Möglichkeiten an: das FORM-Tag und den HTML-Link.

4.3.1 FORM-Tag

Die HTML-Tags `<FORM>` und `</FORM>` klammern einen Bereich mit Eingabedaten. Innerhalb dieses Bereichs werden Werte von Dialogelementen (Eingabefelder, Auswahllisten usw.) vom Browser an ein Programm geschickt. Bei WebTransactions nutzt z.B. die Aufrufseite diese Möglichkeit. In den folgenden Dialogzyklen verwenden Sie für diesen Bereich im Template das WTML-Tag `<wtDataform>` und `</wtDataform>` (siehe WebTransactions-Handbuch „Template-Sprache“). Sie brauchen den URL des bearbeitenden CGI-Programms nicht mehr anzugeben. WebTransactions erzeugt aus dem `wtDataform`-Tag auf der generierten Seite ein FORM-Tag, das genau die aktuelle WebTransactions-Sitzung adressiert.

Beispiel

```
<wtDataform>
     Bitte geben Sie Ihre Adressen ein:
     Strasse: <input type=“text“ name=“STREET“> <br>
     Ort:     <input type=“text“ name=“CITY“> <br>

</wtDataform>

<wtOnReceiveScript>
     hostapp.STREET.VALUE=WT_POSTED.STREET;
     hostapp.CITY.VALUE=WT_POSTED.CITY;
</wtOnReceiveScript>
```

4.3.2 HTML-Link

Wollen Sie den Zugriff auf WebTransactions durch einen Link realisieren, so müssen Sie die URL des aufzurufenden Programms selbst angeben und die Benutzereingaben als Name/
Value-Paare an die URL anhängen.

Im Attribut `HREF` des globalen Systemobjekts ist die URL des CGI-Programms von WebTransactions `WTPublish.exe` enthalten. Dieser URL sind bereits alle Name/Value-Paare angefügt, die WebTransactions benötigt, um die aktuelle Sitzung zu identifizieren.

Sie müssen dieser URL also nur noch die Name/Value-Paare anfügen, die den Benutzereingaben entsprechen. D.h., Sie definieren einen Link folgendermaßen:

```
<A HREF="##WT_SYSTEM.HREF#&name1=value1&name2=value2&...">Link</A>
```

`WT_SYSTEM.HREF`
: Dieses Attribut des Systemobjekts enthält die URL des CGI-Programms `WTPublish.exe` und in verschlüsselter Form die versteckten Felder der aktuellen Seite (BASEDIR, FORMAT, FORMAT_STATE, LANGUAGE, SESSION, SIGNATUR, TIMEOUT_APPLICATION).

`name/value`
: Mit diesen Name/Value-Paaren werden die zusätzlichen Werte definiert, die beim Aktivieren des Links mit gesendet werden. Wenn der Link vom Benutzer aktiviert wurde, stehen diese Paare im Posted-Objekt zur Verfügung. Sie dürfen weder Leerzeichen noch die Zeichen & und = enthalten. Hierbei sind folgende Ersatzzeichen zu verwenden:

nicht erlaubte Zeichen	Ersatzzeichen
%	%25
&	%26
=	%3D
+	%2B
Leerzeichen	%20
/	%2F
:	%3A

Links eignen sich für Sprünge auf andere Verarbeitungsschritte in einer WebTransactions-Anwendung (z.B. Rücksprung zum Hauptmenü) und werden häufig zur besseren Gestaltung verwendet.

Außerdem können Sie dem Benutzer bei einer Auswahl z.B. statt einer Drop-Down-Liste Links anbieten, die er aktivieren kann.

Beispiel

Eine Auswahl von Links: Die geposteten Daten werden im Receive-Tag/-Script ausgewertet:

```
<A HREF="##WT_SYSTEM.HREF#&Command=CONFIRM"> Fortsetzung </A>

<A HREF="##WT_SYSTEM.HREF#&Command=QUIT"> Abbruch </A>

<wtOnReceiveScript>
<--
WT_HOST.comobj.Command.Value = WT_POSTED.Command;
!-->
</wtOnReceiveScript>
```

Der HTML-Link ist unabhängig vom FORM-Tag bzw. Dataform-Tag. Gleichwohl kann er auch innerhalb eines Dataform-Tags stehen.

4.4 Dialog zwischen WebTransactions und Host-Anwendung

Die Dialogsteuerung, d.h. die Folge der HTML-Seiten am Browser, entspricht bei einer automatischen 1:1 Umsetzung der Formatfolge der Host-Anwendung (passiver Dialog). Sie können aber das Template ändern und aktiv in den Dialog eingreifen (aktiver Dialog).

Außerdem haben Sie die Möglichkeit, die Reihenfolge der vom Browser ausgegebenen und zurückgeschickten HTML-Seiten überprüfen zu lassen (synchronisierter Dialog) oder nicht (nicht synchronisierter Dialog).

4.4.1 Passiver Dialog

Die einfachste Art der Dialogsteuerung besteht darin, alles der Host-Anwendung zu überlassen. Sie erhalten dann einen Dialogablauf wie bei Terminalbetrieb, bei dem WebTransactions die Daten für die Browser-Ausgabe „durchreicht“. Jeder Dialogzyklus umfasst die folgenden Aktionen:

- WebTransactions baut aus dem aktuellen Template eine HTML-Seite auf. Dabei wird das Format der Host-Anwendung in ein HTML-Formular umgesetzt.
- WebTransactions speichert die Receive-Tags/-Scripts für die verzögerte Auswertung zwischen.
- Der HTTP-Server liest die HTML-Seite und schickt diese an den Browser.
- WebTransactions wartet auf die Daten vom Browser.
- Der Browser schickt seine Daten an das CGI-Programm WTPublish.exe.
- WebTransactions führt dann die Receive-Scripts aus. Dabei werden gepostete, d. h. vom Browser geschickte, Daten auf die Schnittstelle zur Host-Anwendung übertragen und zur Host-Anwendung geschickt. Außerdem wird der nächste Datensatz von der Host-Anwendung empfangen (Setzen der Host-Objekte und send- und receive-Methode).
- Das Folge-Template wird abhängig vom empfangenen Format bestimmt. Dies geschieht mit der receive-Methode. Aus dem Attribut des Systemobjekts ermittelt WebTransactions den Namen des vom Host empfangenen Formats und legt das zugehörige Template als Folge-Template fest. Jetzt beginnt der Dialogzyklus neu.

Beim passiven Dialog werden die beschriebenen Schritte solange ausgeführt, bis im Template die Sitzung beendet wird. Diese passive Dialogsteuerung ist bereits vollständig realisiert und ablauffähig, wenn Sie WebTransactions für eine bestimmte Host-Anwendung konfigurieren (siehe Abschnitt „Einsatzmöglichkeiten“ auf Seite 23). Alle automatischen Umsetzungen können nur einen passiven Dialog realisieren. Für einen aktiven Dialog müssen Sie in den Ablauf, d.h. in die Templates, eingreifen.

4.4.2 Aktiver Dialog

Die Dialogsteuerung muss nicht der Formatfolge der Host-Anwendung entsprechen. Das Template kann vielmehr den Dialog aktiv beeinflussen.

Sie können z.B. mehrere Hostformate in einer HTML-Seite zusammenfassen und damit die Regel - ein Hostformat entspricht einem Template - durchbrechen. Dazu kommunizieren Sie in einem Template so lange mit der Host-Anwendung, bis Sie alle Daten gesammelt haben, die Sie auf der HTML-Seite darstellen wollen.

Das Blättern am Bildschirm ist ein Beispiel für eine solche Kettung (siehe WebTransactions-Handbuch „Template-Sprache“, Kapitel „WTML-Tags“, Abschnitt „DO UNTIL-Schleife“). So werden z.B. alle Felder eines blätterbaren Formats eingelesen. Erst dann wird eine HTML-Seite zum Browser geschickt. Hier wird also am Ende eines Host-Dialogschritts automatisch ein Folgeschritt gestartet, ohne dass der Dialogzyklus von WebTransactions mit einer Browser-Ausgabe abgeschlossen wird.

Sie können aber auch das nächste zu lesende Template aktiv beeinflussen, indem Sie die eingebaute Funktion `setNextPage()` aufrufen. Am Anfang des folgenden Dialogzyklus wird WebTransactions das angegebene Template verwenden.

Diese Technik wenden Sie z.B. an, wenn Sie ein Hostformat in mehreren Seiten am Browser präsentieren wollen. Ein weiterer Anwendungsfall für diese Vorgehensweise tritt z.B. bei der Kommunikation mit mehreren Host-Anwendungen auf, wenn der Endbenutzer mit jeder der Host-Anwendungen interagiert.

Ein weiteres Beispiel dafür, dass ein Dialogzyklus nicht unbedingt je einen Kontakt zur Host-Anwendung und einen Browser-Kontakt umfasst, ist die Möglichkeit, zwischen verschiedenen Stilen oder Sprachen umzuschalten. Dabei werden die Daten noch einmal neu präsentiert, die im vorhergehenden Schritt bereits vom Host empfangen wurden.

An der Oberfläche bieten Sie dem Benutzer Knöpfe zum Umschalten zwischen den Varianten an. Damit wird ein anderes Template für die HTML-Generierung ausgewertet. Diesen Wechsel zwischen Templates programmieren Sie durch ein Receive-Tag/-Script, das dem Attribut STYLE oder LANGUAGE des Systemobjekts den entsprechenden Wert zuweist. Das FORMAT-Attribut im Systemobjekt bleibt unverändert. WebTransactions startet einen neuen Dialogzyklus, ohne Kontakt zur Host-Anwendung aufzunehmen.

Im Extremfall ist es möglich, eine Anwendung mit WebTransactions ganz ohne Host-Anwendung zu programmieren. Die dynamischen Daten, die auf der HTML-Seite vom Benutzer eingegeben werden, könnten in einer Datei gespeichert und z.B. als Katalog von Bestellungen aufgefasst werden.

4.5 Dialog zwischen WebTransactions und Browser

Beim Anbinden von Host-Anwendungen an das Web wird die Benutzeroberfläche der Host-Anwendungen als Schnittstelle verwendet. WebTransactions verhält sich gegenüber der Host-Anwendung wie ein Terminal. WebTransactions muss daher Formate, die von der Host-Anwendung empfangen wurden, entsprechend ausgefüllt wieder zurückschicken. Um sicherzustellen, dass logische Zustände der Host-Anwendung (z.B. offene Transaktionen einer Datenbank oder eines Transaktionsmonitors) nicht verletzt werden, muss die Reihenfolge der empfangenen und gesendeten Formate genau eingehalten werden.

Im Gegensatz dazu erlauben es die Browser, ältere HTML-Seiten erneut anzuzeigen („History“-Funktion) und z.B. darin enthaltene Formulare erneut zu senden. Um Probleme mit der Host-Anwendung zu vermeiden, kann WebTransactions prüfen, ob die vom Browser empfangenen Daten von der zuletzt aufgebauten HTML-Seite stammen oder ob ein „veraltetes“ Formular geschickt wurde. WebTransactions gibt dann eine Fehlermeldung sowie die zuletzt aufgebaute HTML-Seite erneut aus. Auf diese Weise wird der Dialog mit dem Browser erneut synchronisiert. Dieser synchronisierte Dialog ist die häufigste Form, beispielsweise verwenden alle generierten oder mit dem Capture-Verfahren erzeugten Templates den synchronisierten Dialog.

Es gibt allerdings Fälle, in denen dieser strenge Dialog eine unnötige Einschränkung darstellt; so sollte ein in sich abgeschlossener Zwischendialog mit der Host-Anwendung oder eine Verarbeitung ohne Zugriff auf die Host-Anwendung jederzeit möglich sein. WebTransactions bietet dafür einen **nicht synchronisierten Dialog an,** bei dem die vom Browser geschickte Seite nicht auf ihre Aktualität geprüft wird.

4.5.1 Synchronisierter Dialog

Im synchronisierten Dialog mit dem Browser stellt WebTransactions sicher, dass nur Eingaben von der zuletzt generierten HTML-Seite verarbeitet werden. Die Verarbeitung erfolgt in Dialogzyklen, die aus folgenden Schritten bestehen:

– Aufbau einer HTML-Seite aus einem Template

– Benutzereingaben am Browser und

– Verarbeitung der Eingaben (Aktionen oder Anweisungen in WTML-Tags oder WTScripts des Templates zum Receive-Zeitpunkt)

Diese Dialogzyklen reihen sich in definierter Reihenfolge aneinander. Der Benutzer kann sich zwar ältere Ausgaben ansehen und z.B. Daten aus der Vorgeschichte ermitteln (was die Dialogführung gegenüber dem Terminal bereits erweitert), schickt er aber eine dieser HTML-Seiten ab, so erfolgt eine Fehlermeldung und eine erneute Ausgabe der aktuellen Seite.

Sie können die Benutzereingaben vom Browser auf zwei Arten an eine WebTransactions-Sitzung übermitteln:

– als Daten eines HTML-Formulars oder

– als `QUERY_STRING` innerhalb des URL eines Link.

Im Fall des synchronisierten Dialogs verwenden Sie das `DataForm`-Tag ohne den Parameter `AsyncPage` (siehe WebTransactions-Handbuch „Template-Sprache“) oder einen Link mit dem Attribut `HREF` des Systemobjekts (siehe Abschnitt „Starten durch Eingabe der URL“ auf Seite 102). Die folgende Tabelle zeigt das Ergebnis der Generierung:

Template	**HTML-Seite**

`<wtDataForm>`	`<form method=“post“ action=...>`
	`<input type=“hidden“`
	`name=“WT_SYSTEM_FORMAT_STATE“`
	`value = ...> ...`

`<A`
`HREF="##WT_SYSTEM.HREF#">`
`...`
`</A>`
	`<A HREF=“/cgi-bin/WTPublish.exe`
	`?...&WT_SYSTEM_FORMAT_STATE=...">`
	`...`
	`</A>`

WebTransactions zählt die Dialogzyklen mit dem Browser innerhalb einer WebTransactions-Sitzung. Der aktuelle Wert steht als Attribut `FORMAT_STATE` am Systemobjekt sowie als Name/Value- Paar innerhalb des Attributs `HREF` zur Verfügung. Außerdem wird er bei der Umsetzung eines `DataForm`-Tags als verstecktes Feld in die HTML-Seite generiert.

Wenn WebTransactions von einem solchen Formular oder Link angesprochen wird, wird ein Wert für `WT_SYSTEM_FORMAT_STATE` mitgeschickt. WebTransactions kann diesen mit dem aktuellen Zähler für die Dialogzyklen vergleichen und ggf. mit einer Fehlermeldung reagieren. Stimmen die Werte überein, so führt WebTransactions die OnReceive-Tags/-Scripts aus. Dann beginnt der nächste Dialogzyklus mit dem Lesen des Templates, das im Attribut `FORMAT` spezifiziert ist.

4.5.2 Nicht synchronisierter Dialog

Zwar verbieten offene Transaktionen oder andere bedingte Dialoge ein freies Navigieren in der Dialogoberfläche einer Host-Anwendung. Es gibt jedoch immer wieder Situationen, in denen man den strikten Dialog verlassen könnte, ohne den logischen Zustand der Host-Anwendung zu ändern und Schaden anzurichten. So können Sie z.B. auf einer HTML-Seite einen Knopf anbieten, um Hilfeinformation aus der laufenden Host-Anwendung in einem separaten Fenster anzuzeigen. Hierfür ist ein in sich abgeschlossener Zwischendialog notwendig. Oder Sie wollen beim Host-Adapter OSD abfragen, ob asynchrone Nachrichten oder Druckdaten über die laufende Verbindung angekommen sind, ohne jedes Mal die aktuelle Seite vom Browser abzuschicken. Dazu können Sie den nicht synchronisierten Dialog nutzen, bei dem die Einhaltung der Reihenfolge der Interaktionen nicht geprüft wird.

Beim nicht synchronisierten Dialog verwenden Sie einen Link mit dem Attribut `HREF_ASYNC` des Systemobjekts oder das `DataForm`-Tag mit dem Parameter `AsyncPage`. Die folgende Tabelle zeigt das Ergebnis der Generierung:

Template	**HTML-Seite**

`<A HREF="##WT_SYSTEM.HREF_ASYNC#`
`&WT_ASYNC_PAGE=asPage">`
`...`
`</A>`
	`<A HREF="/cgi-bin/WTPublish.exe`
	`?...&`
	`WT_SYSTEM_FORMAT_STATE=IGNORE`
	`&WT_ASYNC_PAGE=asPage">`
	`...`
	`</A>`

`<A HREF="##WT_SYSTEM.HREF_ASYNC#">`
`...`
`</A>`
	`<A HREF="/cgi-bin/WTPublish.exe`
	`?...&`
	`WT_SYSTEM_FORMAT_STATE=IGNORE">`
	`...`
	`</A>`

`<wtDataForm asyncPage="asPage">`	`<form method="post" action=...>`
	`<input type="hidden"`
	`name="WT_SYSTEM_FORMAT_STATE"`
	`value="IGNORE"> ...`
	`<input type="hidden"`
	`name="WT_ASYNC_PAGE"`
	`value="asPage"> ...`

Das Attribut `HREF_ASYNC` enthält das Name/Value-Paar `WT_SYSTEM_FORMAT_STATE=IGNORE`. Bei der Umsetzung eines `DataForm`-Tags mit dem Parameter `asyncPage` wird im versteckten Feld `WT_SYSTEM_FORMAT_STATE` der Wert `IGNORE` hinterlegt, die Angabe für `asyncPage` wird in ein verstecktes Feld `WT_ASYNC_PAGE` umgesetzt.

Wird nun WebTransactions von einem solchen HTML-Formular oder HTML-Link angesprochen, so zeigt der übertragene Wert `IGNORE` für `WT_SYSTEM_FORMAT_STATE` an, dass die Reihenfolge nicht überprüft werden soll. WebTransactions soll eine Anfrage außerhalb des strengen Dialogs beantworten. Welches Template diese Anfrage bearbeiten soll, wird aus dem ebenfalls übertragenen Wert für `WT_ASYNC_PAGE` ermittelt.

Dieses Template muss so gestaltet sein, dass es jederzeit ausgeführt werden kann. Eine Kommunikation mit der Host-Anwendung ist in diesen Templates grundsätzlich möglich, sollte aber im Hinblick auf die synchronen Zugriffe sehr vorsichtig eingesetzt werden. `OnReceive`-Scripts haben in diesen Templates keine Wirkung, da diese eine Nachbearbeitung innerhalb eines synchronisierten Dialogzyklus voraussetzen.

Einen Sonderfall stellt der Link mit `WT_SYSTEM_FORMAT_STATE=IGNORE` ohne Angabe für `WT_ASYNC_PAGE` dar. Hier wird die letzte synchronisierte Seite erneut ausgegeben. Beispiel 2 zeigt den Einsatz dieser Funktion.

Beispiel 1: Hilfe in separatem Fenster anfordern

Eine Host-Anwendung bietet für jedes Eingabefeld online einen Hilfetext an. Dazu wird ein Fragezeichen in das Eingabefeld geschrieben und die Formate abgeschickt. Daraufhin erscheint die erste Hilfeseite. Die Hilfeseiten enthalten 23 Ausgabezeilen und in Zeile 24 ein Kommandofeld zum Blättern (`+` eine Seite vor, `-` eine Seite zurück) bzw. zum Beenden der Hilfe (`QUIT`). Solange weitere Seiten vorhanden sind, ist das Kommandofeld mit „+“ vorbelegt, hat man bis zum Ende geblättert, so wechselt die Vorbelegung in „-“. Nach Beenden der Hilfe durch das Kommando `QUIT` kehrt die Host-Anwendung zur Ausgangsseite zurück.

Auf den HTML-Seiten soll ein Hilfeknopf angeboten werden. Wird er betätigt, so soll die Hilfe zu dem Feld, in dem der Cursor steht, in einem separaten Fenster angezeigt werden. Die Seite selbst soll nicht abgeschickt werden.

Dazu ergänzen Sie in Ihrem Template innerhalb eines HTML-Formulars die Zeile:

```
<INPUT TYPE="BUTTON" NAME="hilfe" VALUE="Hilfe"
onClick="window.open('##WT_SYSTEM.HREF_ASYNC#'+
'&WT_ASYNC_PAGE=HILFE&CURSOR='+
document.forms[0].CURSOR.value,
'Hilfe',
'scrollbars=yes toolbar=0 location=0'+
'status=0 height=400 width=400'+
'left=0 top=0 resizable=1')">
```

Es wird eine Schaltfläche generiert, die ein neues Fenster öffnet und darin die Hilfe anzeigt. Diese wird über einen nicht synchronisierten Zugriff auf WebTransactions erzeugt. Das Template `HILFE` baut die Seite auf (`WT_ASYNC_PAGE=HILFE`). Dem Template wird über das Name/Value-Paar `CURSOR=...` der Name des aktuellen Felds mitgeteilt (es wird davon ausgegangen, dass der Name des aktuellen Felds in `document.forms[0].CURSOR.value` steht). Es folgt das Template `Hilfe.htm`:

```
<HTML>
<HEAD><TITLE>Hilfe</TITLE></HEAD>
<BODY>
<wtonCreateScript>
<!--
  WT_HOST.STD[WT_POSTED.CURSOR].VALUE='?';                    Aktion 1
  first = true;                                               Aktion 2
  while ( first || WT_HOST.STD.E_24_001_04.VALUE != "-" )     Aktion 3
  {
    WT_HOST.STD.send();                                       Aktion 4
    WT_HOST.STD.receive();
    for (i=1;i<24;i++)                                        Aktion 5
      document.writeln( WT_HOST.STD['E_'+i+'_001_80'].HTMLVALUE,
'<br>');
    first = false;
  }
  WT_HOST.STD.E_24_001_04.VALUE ="QUIT";                      Aktion 6
  WT_HOST.STD.send();
  WT_HOST.STD.receive();
//-->
</wtonCreateScript>
<FORM>
<input type="BUTTON" name="CLOSE" value="Schließen"           Aktion 7
onClick="window.close()“>
</FORM>
</BODY>
</HTML>
```

Das Template `HILFE.htm` holt die Hilfeinformation aus der Host-Anwendung und navigiert dann an die Ausgangsposition zurück. Danach kann der synchronisierte Dialog fortgesetzt werden.

Im Detail werden durch `HILFE.htm` folgende **Aktionen** durchgeführt:

Zunächst wird das Feld, in dem der Cursor auf der HTML-Seite steht, im Format mit '`?`' belegt (Aktion 1). Um wenigstens einmal die Schleife zu durchlaufen, wird `first` auf `true` gesetzt (Aktion 2). Über die Methodenaufrufe `send`/`receive` (Aktion 4) wird die erste und alle ggf. folgenden Hilfeseiten in der Schleife (Aktion 3) empfangen. Für jede empfangene Hilfeseite werden die ersten 23 Zeilen mit Informationen in den Ausgabestrom geschrieben (Aktion 5). Durch das Kommando `QUIT` wird auf die Ausgangsseite zurückgekehrt (Aktion 6). Am Ende der Seite wird eine Schaltfläche ergänzt, die es erlaubt, das Hilfefenster zu schließen (Aktion 7).

Beispiel 2: Frame während einer Sitzung laden/entladen

Innerhalb einer WebTransactions-Sitzung gibt es einige HTML-Seiten, die in einem Frame ablaufen sollen. In einem zweiten Steuer-Frame sollen dann z.B. einige Knöpfe zur Bedienung der Host-Anwendung dargestellt werden. Das Frameset soll aber **nicht** für die gesamte Sitzung vorhanden sein, sondern nur für einige HTML-Seiten. Die Seiten müssen also, wenn sie in den Browser geladen werden, prüfen, ob sie in dem Dialog-Frame des Framesets laufen oder nicht. Einige Seiten müssen dann das Frameset laden, wenn sie feststellen, dass es noch nicht vorhanden ist; andere müssen es entladen, wenn es vorhanden ist. Dieses Verhalten lässt sich folgendermaßen realisieren:

- Ergänzen Sie am Anfang aller Templates, die im Frame ablaufen sollen, die folgende Zeile:

  ```
  <wtInclude name = frameOn>
  ```

- Das Template `frameOn.htm` hat folgenden Inhalt:

  ```
  <SCRIPT>
  if( ! parent.ctr )
    self.location.href ="##WT_SYSTEM.HREF_ASYNC#&WT_ASYNC_PAGE=frameset";
  </SCRIPT>
  ```

  Das client-seitige JavaScript prüft, ob das Frameset bereits geladen ist (Existenz des Objekts `parent.ctr` für den Steuer-Frame). Falls ja, ist nichts zu tun, und die Seite wird ganz normal in dem entsprechenden Frame angezeigt. Falls das Frameset noch nicht vorhanden ist, wird es an die aktuelle Position geladen. Dies geschieht durch einen nicht synchronisierten Zugriff auf die WebTransactions-Sitzung mit dem Template `frameset.htm`.

- `frameset.htm` hat folgenden Inhalt:

```
<HTML>
 <FRAMESET ROWS="20%,80%" BORDER=1>
  <FRAME NAME="ctr" SRC="##WT_SYSTEM.HREF_ASYNC#&WT_ASYNC_PAGE=control">
  <FRAME NAME="dlg" SRC="##WT_SYSTEM.HREF_ASYNC#">
 </FRAMESET>
</HTML>
```

  Das Frameset hat zwei Frames. In den ersten `ctr` wird durch einen erneuten nicht synchronisierten Zugriff auf WebTransactions z.B. eine Schaltflächenleiste geladen. Diese könnte aber auch als statische HTML-Seite definiert sein und ohne WebTransactions geladen werden.

  Quelle für den zweiten Frame ist wieder ein nicht synchronisierter Zugriff auf WebTransactions. Diesmal ist aber kein Template angegeben. WebTransactions schickt deshalb die letzte synchronisierte Ausgabe noch einmal. Damit wird die aktuelle Seite in den Frame `dlg` geladen.

- Ergänzen Sie nun noch am Anfang aller Templates, die nicht im Frame ablaufen sollen, die Zeile:

```
<wtInclude name = frameOff>
```

- Das Template `frameOff.htm` hat folgenden Inhalt:

```
<SCRIPT>
if( parent.ctr )
  parent.location.href = "##WT_SYSTEM.HREF_ASYNC#";
</SCRIPT>
```

  Das client-seitige JavaScript prüft, ob das Frameset geladen ist (Existenz des Objekts `parent.ctr` für den Steuer-Frame). Falls nein, ist nichts zu tun und die Seite wird ganz normal angezeigt. Falls das Frameset vorhanden ist, wird die aktuelle Seite in seine Position (`parent.location`) geladen und das Frameset damit entladen. WebTransactions liefert die aktuelle Seite erneut durch einen nicht synchronisierten Zugriff ohne Angabe für `WT_ASYNC_PAGE`.

  Abhängig von den Einstellungen des Browsers kann das beschriebene Verfahren zu Problemen führen, wenn bestimmte URLs aus dem Cache des Browsers geladen werden. Dann kann man den Cache abstellen oder ein zusätzliches Name/Value-Paar mit schrittweise geänderten Werten in die URL aufnehmen, um ein erneutes Laden vom Server zu erzwingen.

4.6 Sitzung beenden

Eine Sitzung wird entweder explizit beendet, durch Timeout oder über WT_REMOTE.

Mit WebLab lässt sich jede mit WebLab gestartete Sitzung explizit beenden.

4.6.1 Explizites Ende einer Sitzung

Das explizite Ende einer Sitzung wird durch den Benutzer am Browser ausgelöst. Das bedeutet, dass Sie dem Benutzer die Möglichkeit zum Beenden geben müssen, z.B. als Beenden-Knopf, wie er in den generierten Templates zur Verfügung gestellt wird.

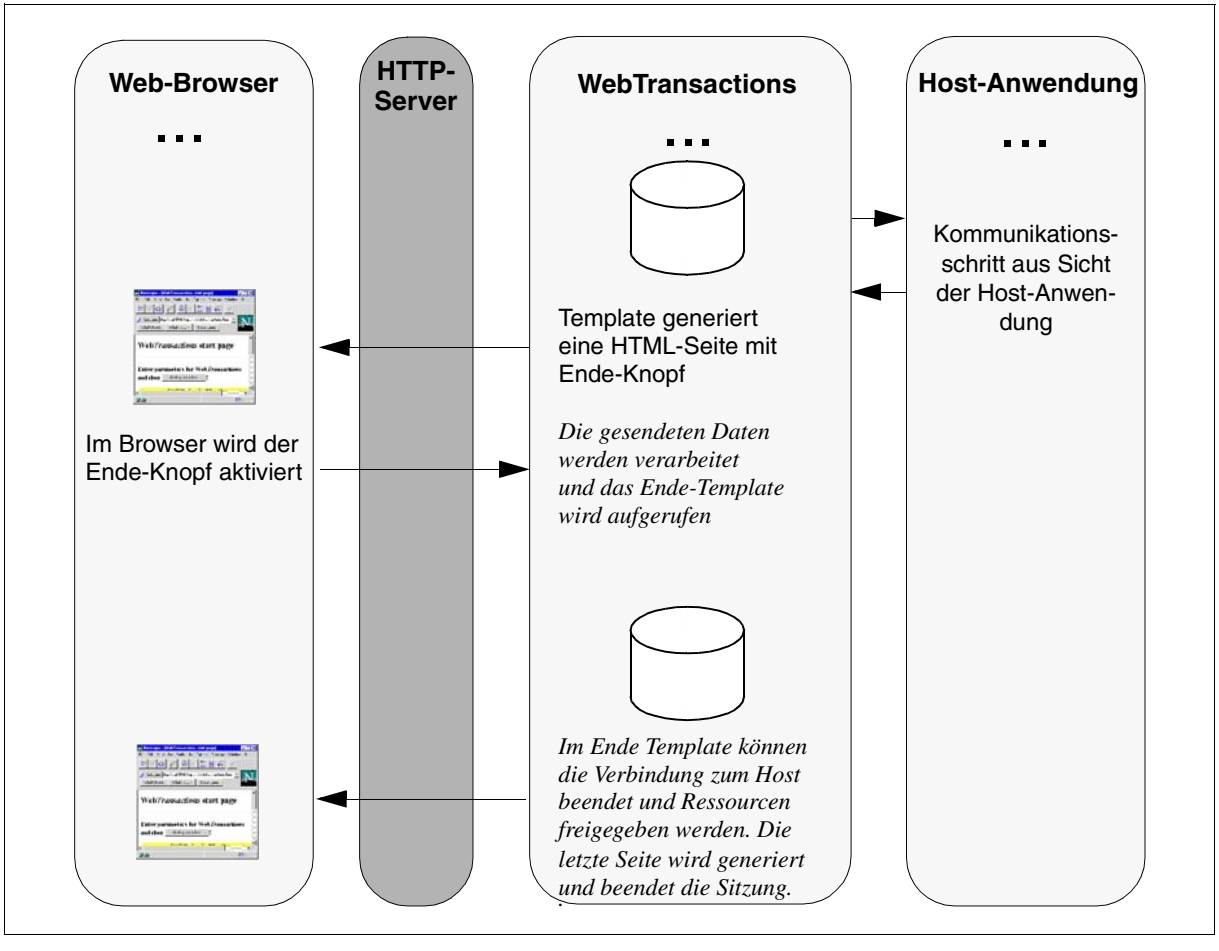


Bild 8: Explizites Ende einer Sitzung

Die letzte generierte HTML-Seite kann keine Verbindung mehr zur WebTransactions-Sitzung aufnehmen, da diese bereits beendet ist, wenn die Seite am Browser erscheint. Daher sollten im entsprechenden Template keine `wtDataForm`-Tags oder Links mit der URL `WT_SYSTEM.HREF[_ASYNC]` verwendet werden. `onReceive`-Scripts des letzten Templates werden nicht mehr berücksichtigt und sind daher sinnlos.

Vor dem Beenden schließt WebTransactions alle noch offenen Verbindungen zu Host-Anwendungen. Sind für diese Verbindungen über das reine Schließen hinaus weitere Aktionen gewünscht, so ist dieses explizit im letzten Template zu programmieren.

Eine Sitzung wird durch den Aufruf der Funktion `exitSession()` explizit beendet:

Diese Anweisung legt das System-Attribut `EXIT_SESSION` an.

Existiert das System-Attribut `EXIT_SESSION` und nicht das System-Attribut `PREVENT_EXIT_SESSION`, so beendet WebTransactions die Sitzung nach der Ausgabe der nächsten Seite, d.h.:

- wird die Aktion zum onCreate-Zeitpunkt durchgeführt, so wird mit dem aktuellen Template eine Seite aufgebaut, an den Browser geschickt und die Sitzung beendet.
- wird die Aktion zum onReceive-Zeitpunkt durchgeführt, so wird nach dem Abarbeiten der Anweisung noch ein weiteres Template gelesen, eine Seite aufgebaut und an den Browser geschickt. Erst dann wird die Sitzung beendet.

Beispiel

```
HTML-Template
    ....
    <Input Type="SUBMIT" Name="Exit" Value="Ende">
    <wtRem Ende oder Kommunikation mit Host>
    <wtonReceiveScript>
    if ( WT_POSTED.Exit == "Ende")
    {
    WT_HOST.std.close();
    exitSession();
    setNextPage("final");
    }
    else
    {
    WT_HOST.std.send();
    WT_HOST.std.receive();
    }
    </wtonReceiveScript>
```

In dem Beispiel soll in den generierten Templates eine Möglichkeit zum sofortigen Beenden der WebTransactions-Sitzung geschaffen werden.

Dazu ist ein entsprechendes Eingabe-Element in die Seite zu integrieren (hier ein Knopf mit dem Namen `Exit`). In einem OnReceive-Script wird nun abgefragt, ob diese Schaltfläche betätigt wurde. Ist das nicht der Fall (`else`-Zweig), so wird der Dialog mit der Host-Anwendung ganz normal fortgesetzt (Methodenaufrufe `send()/receive()`). Ist dagegen das Ende der Sitzung verlangt worden, so wird

– die Verbindung zur Host-Anwendung geschlossen (Methodenaufruf `close()`),

– das Ende der WebTransactions-Sitzung angefordert (Aufruf der Funktion `exitSession()`)

– und auf ein abschließendes Template `final.htm` umgeschaltet.

Das Template kann einen abschließenden Text und ggf. Links oder Formulare zum Starten neuer WebTransactions-Sitzungen enthalten (siehe Abschnitt „WebTransactions-Dialoganwendung starten“ auf Seite 100). Das Template darf keine `wtDataForm`-Tags oder Links mit der URL `WT_SYSTEM.HREF` enthalten, da diese auf eine bereits beendete Sitzung verweisen und damit zu einer Fehlermeldung führen würden.

Vordefiniertes Ende-Template

Mit WebTransactions wird das allgemeine Ende-Template `wtend.htm` ausgeliefert. `wtend.htm` kann für einfache 1:1-Anbindungen z.B. für die `disconnect`-Bearbeitung eingetragen werden, um eine abgeschlossene Anbindung zu erhalten.

Beim Erstellen eines Basisverzeichnisses (siehe Abschnitt „Basisverzeichnis erstellen“ auf Seite 172) wird ein Link `wtend.htm` eingerichtet. Wenn Sie `wtend.htm` in WebLab editieren und speichern, wird der Link aufgelöst und die geänderte Datei im Basisverzeichnis gespeichert.

4.6.2 Beenden durch Timeout

Die HTTP-Verbindung zwischen Web-Server und Browser wird nach der Ausgabe jeweils einer HTML-Seite abgebaut. Somit besteht in der Phase, in der eine Seite am Browser angezeigt wird, nur eine logische, aber keine physikalische Verbindung zwischen Browser und WebTransactions. Verlässt der Endbenutzer die Host-Anwendung, indem er z.B. den Browser beendet, so wird der stellvertretende Prozess davon nicht benachrichtigt. Damit solche verlassenen Sitzungen nicht beliebig lange sinnlos warten, unterstützt WebTransactions einen Timeout-Mechanismus.

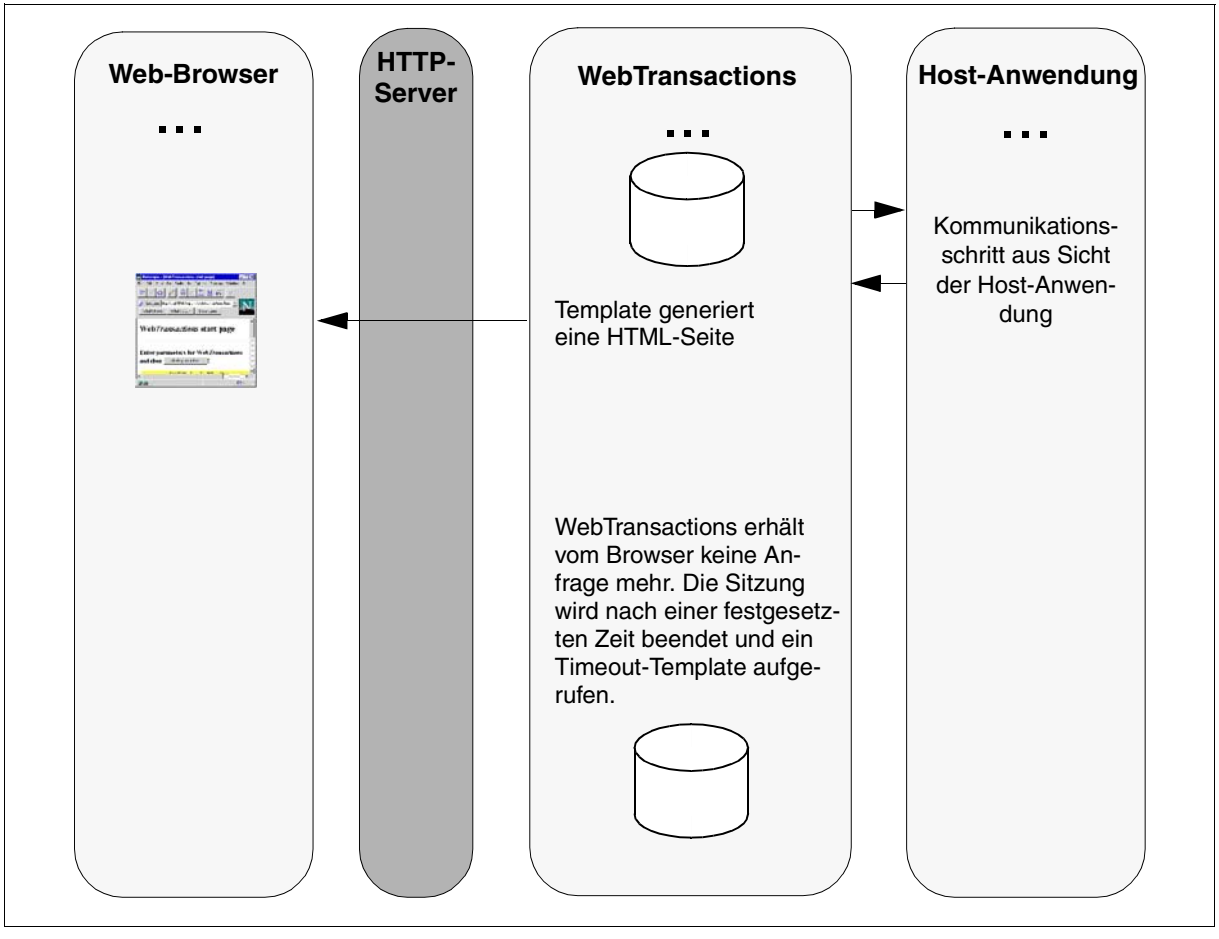


Bild 9: Ende einer Sitzung durch Timeout

Wird innerhalb der Timeout-Spanne keine Anfrage an WebTransactions gestellt, so wird die Sitzung beendet. Vor dem Beenden schließt WebTransactions alle noch offenen Verbindungen zu Host-Anwendungen. Sind für diese Verbindungen über das reine Schließen hinaus weitere Aktionen gewünscht, so sind diese explizit in einem speziellen Timeout-Template zu programmieren. Der Name dieses Templates wird im Attribut `TIMEOUT_FORMAT` am Systemobjekt hinterlegt. WebTransactions führt dieses Template als letzte Aktion vor dem Ende der Sitzung aus.

Bitte beachten Sie, dass ein Timeout-Template keine Ausgaben an den Browser enthalten darf, da es erst aufgerufen wird, wenn die Verbindung zum Browser nicht mehr besteht

Im Attribut `TIMEOUT_USER` des Systemobjekts wird die Zeit in Sekunden abgelegt, die WebTransactions auf die nächste Anfrage vom Browser warten soll. Folgende Anweisung stellt z.B. eine Timeout-Spanne von 5 Minuten ein:

```
<wtOn..Script>
  WT_SYSTEM.TIMEOUT_USER = "300";
</wtOn..Script>
```

Wird das Attribut `TIMEOUT_USER` nicht explizit in einem Template gesetzt, so nimmt WebTransactions eine Zeit von 10 Minuten an.

Beispiel

In diesem TIMEOUT-Template wird nicht nur eine bestehende Verbindung beendet, sondern der Benutzer wird vorher noch mit einem speziellen Kommando abgemeldet.

```
<wtOnCreateScript>
<!--
   WT_HOST.osd.send();
   WT_HOST.osd.receive();
   WT_HOST.osd.E_001_001.VALUE="LOGOFF SYSTEM_OUTPUT=*TAPE_OUTPUT";
   WT_HOST.osd.send();
   WT_HOST.osd.receive();
   WT_HOST.osd.close();
//-->
</wtOnCreateScript>
```

4.6.3 Beenden über WT_REMOTE

Ein Client-Programm beendet eine Sitzung über die Schnittstelle `WT_REMOTE` mit der Methode `EXIT_SESSION`.

4.7 Diagnose in einer WebTransactions-Anwendung

Als Diagnosemöglichkeiten stehen in einer WebTransactions-Anwendung verschiedene Trace-Funktionen zur Verfügung. Außerdem können Sie Sitzungen aufzeichnen.

Eine weitere Diagnosemöglichkeit bietet die Einzelschrittverfolgung in WebLab, siehe auch Abschnitt „Ablauf über ein Template oder über einen Bereich im Template verfolgen“ auf Seite 209.

4.7.1 Trace-Funktionen

Zu Diagnosezwecken können Sie unterschiedliche Traces einstellen:

– Kommunikations-Traces für WebLab und WTEdit

– einen Sitzungs-Trace für WebTransactions

4.7.1.1 Kommunikations-Traces

Damit Kommunikations-Probleme zwischen WebLab und WTEdit schnell erkannt werden, können Sie in WebLab mit **Optionen/Einstellungen/Diagnose** zwei Traces aktivieren:

– den WebLab-Trace

– den WTEdit-Trace

Beide Traces protokollieren die Kommunikation zwischen WebLab und WTEdit:

– der WebLab-Trace protokolliert die WebLab-Kommunikation lokal in die angegeben Datei

– der WTEdit-Trace protokolliert die WTEdit-Kommunikation auf dem WebTransactions-Server in die angegeben Datei

Zur schnellen Diagnose werden beide Dateien benötigt. Sie geben einen vollständigen Überblick über die Kommunikation.

4.7.1.2 WebTransactions-Trace

Der WebTransactions-Trace protokolliert die gesamte Sitzung. In der Trace-Datei finden Sie Informationen aus folgenden Quellen:

– interne Funktionsaufrufe

– Aufrufe von Funktionen des Host-Adapters

– Aufrufe von Userexits (sofern diese Trace-Einträge schreiben)

– globale WTScript-Funktion `writeToTrace()` (für beliebige eigene Trace-Einträge)

Trace in WebLab einschalten

In WebLab haben Sie zwei Möglichkeiten, den WebTransactions-Trace ein- oder auszuschalten:

– die Befehle **Optionen/Trace einschalten** bzw. **ausschalten**

– Sie klicken jeweils doppelt auf das Symbol  der Statusleiste

Diese Einstellung bleibt über das Sitzungsende hinaus aktiv. Sie müssen den Trace explizit wieder ausschalten.

Wenn der Trace-Modus eingeschaltet ist, erstellt WebTransactions im Basisverzeichnis unter `tmp/`*`session_id`*`/Trace` eine Datei. Die SessionID können Sie, solange die Sitzung läuft, mit **Optionen/Informationen anzeigen** ermitteln.

Sie können die Trace-Datei ansehen mit den Befehlen

– **Datei/Öffnen**

– **Administrieren/Anwendung**

Trace für eine Anwendung einschalten

Für eine WebTransactions-Anwendung schalten Sie die Trace-Funktion ein, indem Sie WebTransactions über eine HTML-Aufrufseite starten, die ein Form-Tag der Methode POST enthält. Das Formular muss ein Input-Element mit Namen `WT_TRACE` enthalten, für das der Value `Trace` gesetzt ist. Dabei kann es sich um ein verstecktes Input-Feld handeln, wie in folgender Darstellung. Sie können aber auch Input-Felder eines anderen Typs verwenden und z.B. über eine Checkbox eine Wahlmöglichkeit bieten, ob die Tracefunktion eingeschaltet werden soll oder nicht.

```
<form method="post" action= "/cgi-prefix/WTPublish.exe/basedir?startTemplate">
<input type="hidden" name="WT_TRACE" value="Trace">
```

Das Feld `WT_TRACE` muss den Wert `Trace` haben, um den Trace einzuschalten. Alle anderen Werte (einschließlich nicht definiert) schalten den Trace aus.

Trace in einem Template einschalten

Alternativ zu `WT_TRACE` können Sie in einem Template auch die globale Funktion `setTraceLevel()` nutzen. Mit `setTraceLevel("FULL")` lässt sich der Trace während einer laufenden Sitzung einschalten und mit `setTraceLevel("NONE")` wieder ausschalten.

Dadurch kann der Trace auf bestimmte Teile der Templates beschränkt werden, um die Trace-Datei klein zu halten und auf festgelegte Abschnitte der Sitzung zu beschränken. Die Funktion ist im WebTransactions-Handbuch „Template-Sprache“ beschrieben.

Trace Level für Host-Adapter festlegen

Für die Host-Adapter OSD und MVS können Sie nicht nur den Trace ein- oder ausschalten, sondern auch die Ebene festlegen, auf der protokolliert wird. Über das verbindungsspezifische Systemobjekt-Attribut `TRACE_LEVEL` können Sie festlegen, was protokolliert wird:

– unterschiedliche Trace Levels von 0 bis 3

– `E`: Ausgabe der Aufrufe der Emulationsfunktionen

– `M`: Ausgabe aller Host-Matrizen, d.h. der "rohen" Maskendaten

Detaillierte Informationen dazu finden Sie in den WebTransactions-Handbüchern zu den jeweiligen Host-Adaptern.

4.7.2 Sitzung aufzeichnen

Zur Laufzeit wird die Kommunikation zwischen dem Kern von WebTransactions und der Host-Anwendung über die in den Host-Adapter integrierte Terminal-Emulation abgewickelt. Wenn Sie eine Sitzung aufzeichnen, werden alle Nachrichten zur Host-Anwendung und von der Host-Anwendung protokolliert. Sie können danach die Aufzeichnung „abspielen“. D.h., die aufgezeichnete Sitzung läuft nochmals ab, ohne eine Verbindung zum Host zu haben („offline“). Damit können Sie eine WebTransactions-Anwendung auch weiterentwickeln, ohne dass Sie eine Verbindung zum Host haben.

Sie zeichnen eine Sitzung auf, indem Sie im Start-Template das Systemobjekt-Attribut `RECORD_HOST_COMMUNICATION` auf den Wert „Yes“ setzen. Standardmäßig wird die Datei, die die Aufzeichnung enthält, `WEBTADUMP.LOG` genannt. Sie können aber über das Systemobjekt-Attribut `OFFLINE_LOGFILE` einen anderen Namen festlegen.

```
RECORD_HOST_COMMUNICATION="Yes"
OFFLINE_LOGFILE=<filename>
```

Aufzeichnung „abspielen“

Sie spielen eine Aufzeichnung ohne Verbindung zur Host-Anwendung ab, indem Sie die folgenden Systemobjekt-Attribute mit Werten versorgen:

```
OFFLINE_COMMUNICATION="Yes"
OFFLINE_TRACEFILE="<trace filename>"
```

Da es unterschiedliche Systemobjekt-Attribute für die Dateien gibt, in denen eine Sitzung aufgezeichnet und abgespielt wird, können Sie in einer Sitzung eine Aufzeichnung anlegen und gleichzeitig eine ältere abspielen.

4.8 WebTransactions-Anwendung transferieren

In der Regel werden WebTransactions-Anwendungen auf einem Entwicklungsrechner erstellt und getestet. Für den produktiven Einsatz werden sie auf einen Produktionsrechner transferiert. Die Wartung und Weiterentwicklung verbleibt auf dem Entwicklungsrechner und muss von Fall zu Fall auf dem Produktionsrechner ergänzt werden.

Beim Transfer einer WebTransactions-Anwendung von einem Rechner zum anderen unterstützt Sie ein Assistent. Dabei werden alle Dateien und Verzeichnisse unter einem Basisverzeichnis mit WebLab in ein Archiv gepackt. Dieses Archiv können Sie auf einem anderen Rechner einspielen und dort mit WebLab wieder entpacken. Damit wird auf dem Produktivrechner das Basisverzeichnis wieder genau so hergestellt, wie Sie es auf dem Entwicklungsrechner angelegt hatten.

WebTransactions-Cluster-Anwendungen können Sie mit dem Befehl **Administration**/**Anwendung verteilen** auf die Cluster-Member verteilen.

4.8.1 Anwendung mit Kommando entpacken

Wenn Sie auf einen WebTransactions-Server mit WebLab nicht zugreifen können, können Sie eine WebTransactions-Anwendung auch mit dem Kommando `install` von WTEdit.exe entpacken. Dieses Kommando erzeugt auf dem Server, auf dem es ausgeführt wird, ein Basisverzeichnis und entpackt die angegebene WebTransactions-Anwendung in dieses Basisverzeichnis. Das Kommando rufen Sie wie folgt auf:

`wtedit.exe install` *name basedir kennung passwort wtpublish-pfad isapi-pfad cluster-pfad*

Sie müssen alle Parameter angeben.

name	Name des Archivs, das entpackt werden soll, mit absolutem Pfad
basedir	Basisverzeichnis, das erzeugt werden soll, mit absolutem Pfad
kennung	WebTransactions-Benutzerkennung
passwort	Passwort der Kennung
wtpublish-pfad	Aliasname für den Pfad von WTPublish.exe
isapi-pfad	Aliasname für den Pfad vonWTPublishISAPI.exe
cluster-pfad	Aliasname für den Pfad von WTCluster.exe

Beispiel

Das Beispiel entpackt die Datei *packtest.zip* in das Basisverzeichnis *osd_test*. Als Benutzerkennung wird *ac13* verwendet, diese Benutzerkennung hat kein Passwort. Der Befehl muss in einer Zeile angegeben werden.

```
wtedit.exe install "d:\projekte\packtest.zip" "d:\basedirs\osd_test" ac13 ""
cgi-bin cgi-bin cgi-bin
```

4.9 Client-Schnittstelle WT_REMOTE

Mit WT_REMOTE steht eine Schnittstelle für beliebige Client-Programme zur Verfügung, mit der WebTransactions-Anwendungen genutzt werden können. Dabei kann über diese Schnittstelle nicht nur auf die Daten der zu Grunde liegenden Dialoganwendungen zugegriffen werden, sondern alle Ressourcen der WebTransactions-Sitzung können genutzt werden.

Über WT_REMOTE kann auf eine laufende WebTransactions-Sitzung zugegriffen werden, indem die Sitzungs-ID beim Client-Zugriff angegeben wird. Eine WebTransactions-Sitzung kann auch eigens von einem Client-Programm gestartet werden. Dann sind sowohl Einschritt- als auch Mehrschritt-Transaktionen möglich:

– Bei einer Einschritt-Transaktion wird eine WebTransactions-Sitzung für die Ausführung eines einzelnen Kommandos gestartet, das Kommando ausgeführt und die Sitzung danach wieder beendet. Dies alles geschieht über einen einzigen Client-Zugriff auf WT_REMOTE.

– Bei einer Mehrschritt-Transaktion wird eine WebTransactions-Sitzung durch den WT_REMOTE-Zugriff START_SESSION explizit gestartet. Danach werden mehrere Client-Zugriffe mit PROCESS_COMMANDS durchgeführt und die Sitzung wird mit EXIT_SESSION schließlich beendet.

WT_REMOTE bietet für die Kommunikation mit der WebTransactions-Anwendung die Methoden START_SESSION, PROCESS_COMMANDS und EXIT_SESSION an, mit deren Hilfe Client-Zugriffe möglich werden. START_SESSION und EXIT_SESSION stellen die Methoden für den expliziten Sitzungsstart und die Beendigung einer Sitzung im Rahmen einer Mehrschritt-Transaktion dar. Mit PROCESS_COMMANDS werden die eigentlichen Client-Zugriffe ausgeführt (sowohl bei Einschritt- als auch bei Mehrschritt-Transaktionen).

Über die Methode PROCESS_COMMANDS können Daten vom Client-Programm zur WebTransactions-Anwendung und umgekehrt übertragen, Objekte in der WebTransactions-Sitzung erzeugt und Methoden innerhalb der WebTransactions-Sitzung aufgerufen werden.

Die Anwendungsbereiche für die WT_REMOTE-Schnittstelle sind vielfältig:

– WebTransactions-Sitzungen können über mitgelieferte Bibliotheken die WT_REMOTE-Schnittstelle nutzen, um miteinander zu kommunizieren, wodurch verteilte WebTransactions-Sitzungen möglich werden (WT.RPC).

– Mit Hilfe vordefinierter Klassen können JAVA-Applets auf WebTransactions-Sitzungen zugreifen und deren Leistungen für eigene Zwecke nutzen.

– Sie können auch Dialoganwendungen und Client-Programme miteinander kombinieren. Beispielsweise kann in einer WebTransactions-Dialoganwendung eine HTML-Seite mit einem Applet generiert werden, das seinerseits über WT_REMOTE mit der WebTransactions-Anwendung kommuniziert.

– Auf der Basis der offen gelegten Schnittstelle WT_REMOTE können Sie eigene Client-Programme für WebTransactions realisieren.

Eine ausführliche Beschreibung der Client-Schnittstelle finden Sie im WebTransactions-Handbuch „Client-APIs für WebTransactions“.

4.10 WebTransactions-Anwendung administrieren

Sie können Ihre WebTransactions-Anwendung vom Browser aus vollständig administrieren. Hierfür stellt Ihnen WebTransactions ein Administrationsprogramm bereit, das auf einem eigenen Sicherheits- und Benutzerkonzept aufsetzt, siehe hierzu auch Abschnitt „Benutzerkonzept“ auf Seite 142.

Das Administrationsprogramm umfasst folgende **Funktionen**:

- Anzeige der Sitzungen
- Sperren und Entsperren von WebTransactions. Sperren bedeutet, dass keine neuen Sitzungen mehr gestartet werden können.
- Beenden von laufenden Sitzungen
- Anzeige der temporären Dateien laufender Sitzungen
- Bereinigung des `tmp`-Verzeichnisses

Administrationsprogramm aufrufen

Die Administrationsoberfläche starten Sie entweder über WebLab oder direkt im Browser über eine eigene URL.

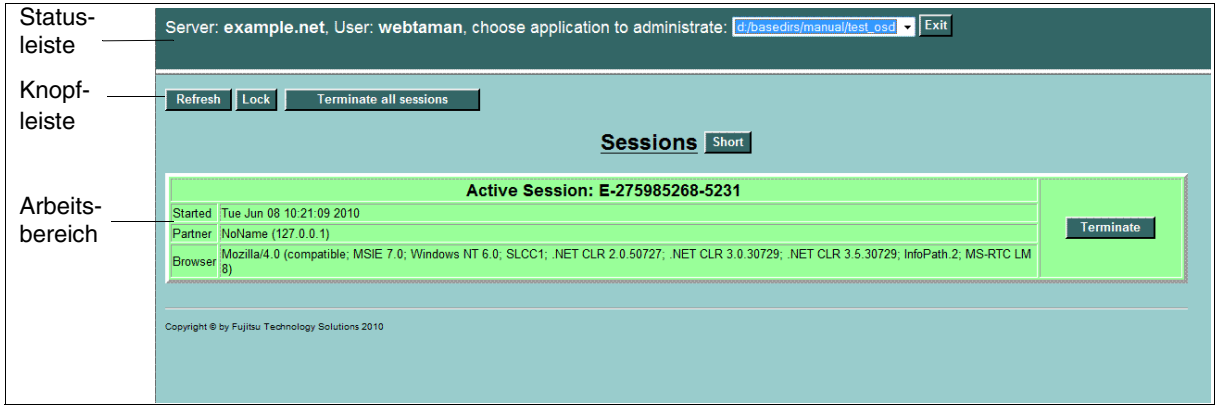
Aufruf in WebLab*:*
Wählen Sie den Befehl **Administrieren/Anwendung**.

Aufruf über URL:
Starten Sie einen beliebigen Browser und starten Sie das Administrationsprogramm mit folgender URL:
`http://`*webta-server/cgi-präfix*`/WTPublish.exe/server-admin`.
webta-server ist dabei der Name des Rechners, auf dem WebTransactions läuft, *cgi-präfix* ist der Pfad auf diesem Rechner für CGI-Programme.

Das Administrationsprogramm wird gestartet und das Anmelde-Fenster im Browser angezeigt. Melden Sie sich mit der Kennung an, die Sie vom WebTransactions-Administrator erhalten haben, um Ihre eigenen WebTransactions-Anwendungen zu administrieren. Sie erhalten das Fenster **WebTransactions Application Administration**.

Aufbau der Administrationsoberfläche

Die folgende Abbildung zeigt das Administrationsfenster:



Die Oberfläche des Administrationsprogramms zur Anwendungsverwaltung besteht aus drei Bereichen:

- Der Statusleiste, die den Namen des aktuellen WebTransactions-Servers und den Benutzer anzeigt. Wenn Sie mehr als eine WebTransactions-Anwendung auf dem WebTransactions-Server bearbeiten, dürfen Sie aus einer Liste die zu administrierende Anwendung auswählen.
- Einer Knopfleiste, mit der Sie die Administration steuern können. Abhängig von der Situation auf dem WebTransactions-Server werden folgende Knöpfe angezeigt:

  Refresh

  Aktualisiert den Inhalt der Seite.
  (Die Seite wird auch in regelmäßigen Abständen automatisch aktualisiert. Der Balken am oberen Rand der Statusleiste zeigt die Zeit an bis zum nächsten automatischen Refresh: je kürzer dieser Balken ist, desto kürzer die Zeit.)

Lock
Zeigt an, dass WebTransactions neue Sitzungen zulässt. Wenn Sie auf diesen Knopf klicken, wird der Knopf mit der Beschriftung **UnLock** angezeigt und die WebTransactions-Anwendung wechselt in den Zustand gesperrt. Das bedeutet, dass zwar die laufenden Sitzungen erhalten bleiben, aber keine neuen Sitzungen mehr gestartet werden können. Um die WebTransactions-Anwendung wieder zu entsperren, klicken Sie auf den Knopf **Unlock**.

Terminate all Sessions
Beendet alle laufenden Sitzungen.

Delete all files of dead sessions
Löscht alle Dateien, die zu beendeten Sitzungen gehören. Laufende Sitzungen werden davon nicht berührt.

Remove all files not related to any session
Löscht alle Dateien im temporären Verzeichnis, die keiner Sitzung zugeordnet werden können. Diese Dateien werden in einer Liste angezeigt, falls vorhanden.

- Dem Arbeitsbereich, in dem alle Sitzungen mit ihren Betriebsmitteln und ihrer Sitzungs-ID angezeigt werden. Folgende Einträge können im Arbeitsbereich angezeigt werden:

Dead Session

Sind beim Beenden einer Sitzung noch Dateien im Sitzungsverzeichnis vorhanden (z.B. Trace-Dateien) wird die Sitzung als „Dead Session“ noch angezeigt und ein Zugriff auf die Sitzungsdateien ist weiterhin möglich (über den entsprechenden Link in der Zeile `Session Files`). Mit dem Knopf **Delete** können diese Dateien gelöscht werden.

Active Session

Für laufende Sitzungen zeigt die Administrationsoberfläche folgende Informationen an:

– die Betriebssystem-Mittel für die Prozesskommunikation zwischen WebTransactions und Browser bzw. Host-Anwendung („named pipes“)

– die Startzeit der Sitzung

– die Internet-Adresse des Rechners, auf dem der Browser abläuft (bzw. ggf. die Adresse eines Proxy-Rechners)

– Informationen über den Browser

– temporäre Dateien der Sitzung (z.B. Print-Files) und die Trace-Datei.

Mit dem Knopf **Terminate** können Sie jede dieser Sitzungen beenden. D.h. die Holder-Task wird beendet, und alle dazugehörigen temporären Dateien werden gelöscht.

Schickt der Benutzer dieser gelöschten Sitzung seine aktuelle HTML-Seite vom Browser ab, gibt WebTransactions eine Fehlermeldung aus. Der Benutzer muss dann eine neue Sitzung eröffnen.

Wenn es zu der angezeigten Sitzung eine Trace-Datei gibt, wird unter **Session Files** der Name dieser Datei ausgegeben. Mit dem entsprechenden Link können Sie temporäre Dateien anschauen.

5 WebTransactions-Server

Sie können den WebTransactions-Server vom Browser aus vollständig administrieren. Hierfür stellt Ihnen WebTransactions ein Administrationsprogramm bereit, das auf einem eigenen Sicherheits- und Benutzerkonzept aufsetzt, siehe hierzu auch Abschnitt „Benutzerkonzept“ auf Seite 142. Das Administrationsprogramm ist eine eigenständige WebTransactions-Anwendung, die Sie mit WebLab oder direkt aus dem Browser aufrufen können. Das Programm selbst läuft immer auf dem WebTransactions-Server ab.

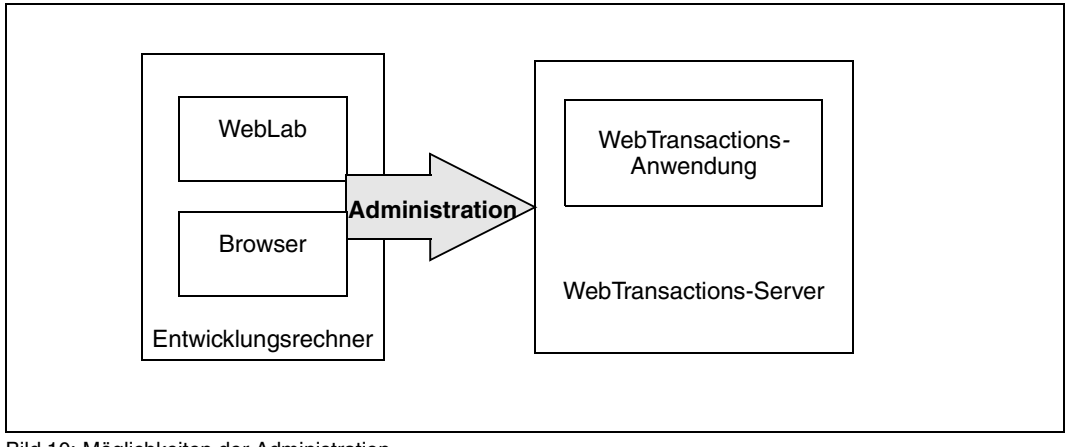


Bild 10: Möglichkeiten der Administration

Das Administrationsprogramm umfasst folgende **Funktionen**:

– Lizenzen eingeben oder erhöhen (siehe Seite 146)

– WebTransactions-Server verwalten, d.h. die grundsätzliche Handhabung der Verwaltung von Benutzern, Pools, Anwendungen, Sitzungen, Werkzeugen, Clustern (siehe Seite 150)

– Cluster einrichten und verwalten (siehe Seite 157)

5.1 Benutzerkonzept

Die Verwaltung von WebTransactions baut auf einem eigenen Benutzerkonzept auf, über das die Zugriffskontrolle auf WebTransactions-Server und auf WebTransactions-Anwendungen realisiert wird.

WebTransactions kennt zwei unterschiedliche Benutzertypen: `admin` und andere `user`.

`admin` entspricht dem Administrator unter Windows oder dem Superuser in Unix-Systemen, der im WebTransactions-Umfeld auf dem WebTransactions-Server alles darf. Der Benutzer `admin` ist der Server-Administrator von WebTransactions und wird ohne Passwort schon während der Installation von WebTransactions angelegt. Er hat alle Rechte und kann nicht gelöscht werden.

Denken Sie daran, dem Benutzer `admin` direkt nach der Installation ein sicheres Passwort zu geben. Sie können das Passwort jederzeit ändern.

Die anderen `user` werden vom Benutzer `admin` angelegt und haben die Rechte und Zugriffsmöglichkeiten, die ihnen `admin` gewährt. Diese `user` können beispielsweise entsprechend ihren Aufgaben bestimmte Rollen annehmen:

– Die Anwendungsadministratoren dürfen WebTransactions-Anwendungen administrieren. Sie können Sitzungen überwachen, Trace-Dateien auswerten, WebTransactions-Anwendungen sperren oder freigeben, Sitzungen beenden.

– Die Anwendungsentwickler dürfen mit WebLab neue WebTransactions-Anwendungen erstellen und bestehende weiterentwickeln.

Der einzelne Anwendungsprogrammierer braucht sich nicht mehr um Einstellungen des Betriebssystems oder des Web-Servers zu kümmern und kann davon ausgehen, dass seine WebTransactions-Anwendungen auf dem Server vor unerwünschten Zugriffen geschützt sind.

Die Verwaltungsfunktionen hängen von der Rolle ab, die der jeweilige Benutzer ausübt:

– Der WebTransactions-Administrator kann den WebTransactions-Server als einzelnen Server und als Teil eines Clusters verwalten. Die Administration eines WebTransactions-Servers ist in den folgenden Abschnitten beschrieben.

– Jeder Anwendungsprogrammierer kann seine WebTransactions-Anwendungen verwalten. Der Aufruf des Administrationsprogramms erfolgt mit WebLab mit dem Befehl **Administrieren/Anwendung**. Diese Administration ist im Abschnitt „WebTransactions-Anwendung administrieren“ auf Seite 137 beschrieben.

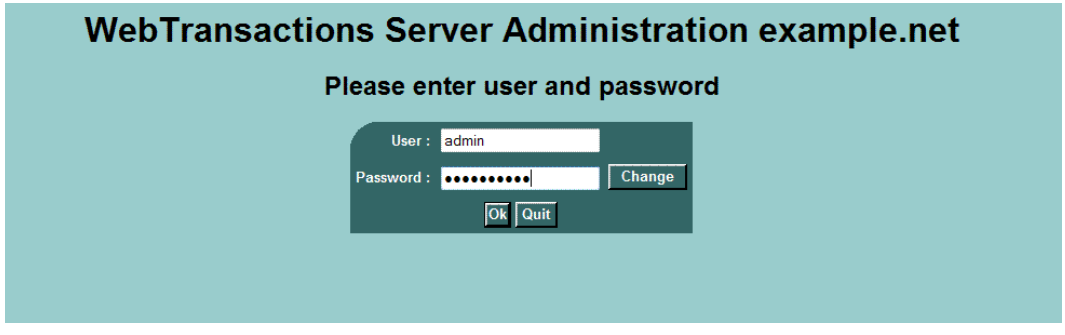
5.2 Administration starten

Das Administrationsprogramm starten Sie entweder über WebLab oder direkt im Browser über eine eigene URL.

Aufruf in WebLab*:*
Wählen Sie den Befehl **Administration/Server**.

Aufruf über URL:
Starten Sie einen beliebigen Browser und starten Sie das Administrationsprogramm mit folgender URL:
`http://`*webta-server/cgi-präfix*`/WTPublish.exe/server-admin`. *webta-server* ist dabei der Name des Rechners, auf dem WebTransactions läuft, *cgi-präfix* ist der Pfad auf diesem Rechner für CGI-Programme.

Das Administrationsprogramm wird gestartet und das Anmelde-Fenster im Browser angezeigt.



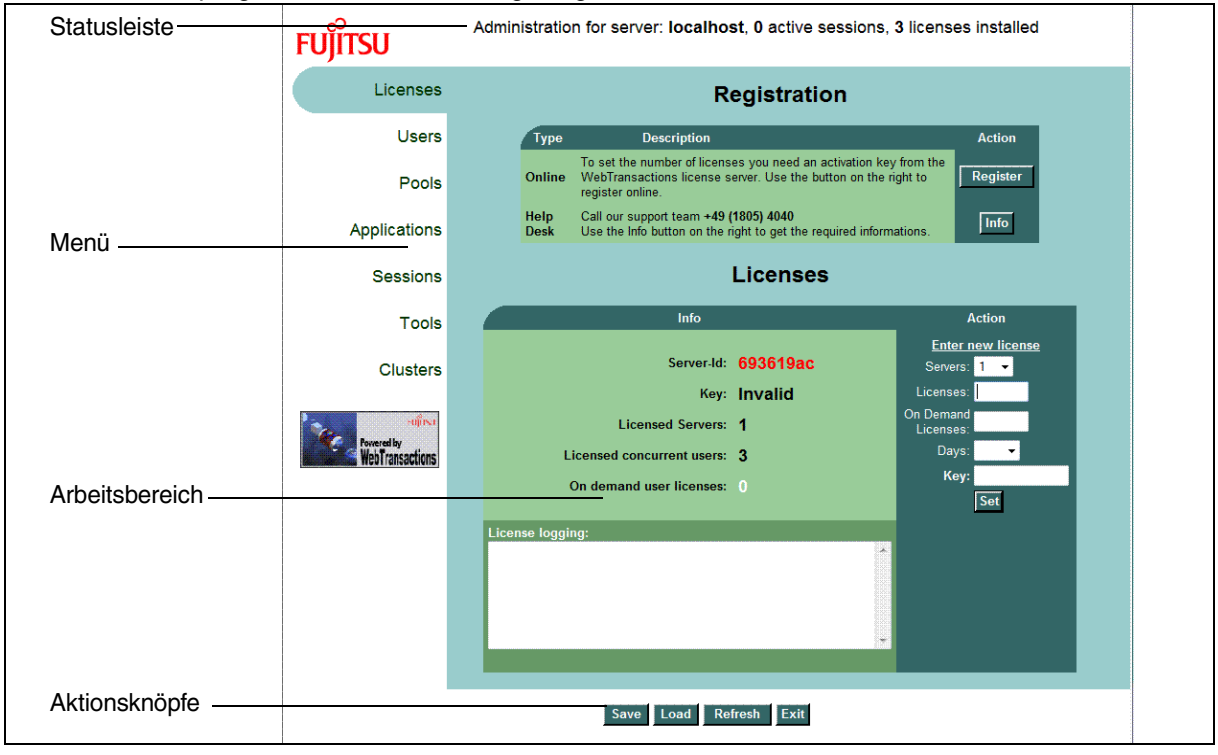
Wenn Sie das Administrationsprogramm von WebTransactions erstmals starten, sollten Sie für den Benutzer `admin` erst ein Passwort setzen. Mit einem Klick auf **Change** setzen oder ändern Sie das Passwort.

Bei jedem weiteren Anmelden ist die Sicht auf das Administrationsprogramm abhängig davon, unter welcher Kennung Sie sich anmelden:

`admin`
Sie sind der Server-Administrator von WebTransactions

anderer `user`
Sie haben die Rechte, die `admin` Ihnen gewährt, siehe auch Abschnitt „WebTransactions-Anwendung administrieren“ auf Seite 137.

Nachdem Sie sich als `admin` angemeldet haben, wird das erste Fenster des Administrationsprogramms im Browser angezeigt:



Die Oberfläche des Administrationsprogramms besteht aus

– einer Statusleiste mit Informationen zum verwalteten Server, Anzahl Sitzungen und Lizenzen

– einem vertikalen Menü links im Fenster, mit den Menüpunkten **Licenses**, **Users**, **Pools**, **Applications, Sessions, Tools** und **Clusters**. Die Darstellung des Menüs ist abhängig vom verwendeten Browser und der installierten Liefereinheit.

– dem Arbeitsbereich rechts neben dem Menü, in dem Sie die eigentliche Arbeit ausführen

– den Aktionsknöpfen, mit denen Sie Ihre Einstellungen sichern oder laden, den Inhalt des Browser-Fensters neu aufbauen oder aber das Administrationsprogramm beenden können. Alle Änderungen während der Administration werden erst wirksam, wenn Sie auf den Knopf **Save** klicken.

Alle Einstellungen, die Sie zur Administration der WebTransactions-Server und -Anwendungen eingeben, werden in der Datei `config/wtaccess` im Installationsverzeichnis von WebTransactions abgelegt.

5.3 Lizenzen eingeben oder erhöhen

WebTransactions bietet folgende drei Lizenzmodelle:

– Standalone-Lizenzen

– onDemand-Lizenzen

– Cluster-Lizenzen

Wie Sie jeweils Lizenzen registrieren, wird in den nachfolgenden Abschnitten detailliert beschrieben.

Nachdem Sie sich beim Administrationsprogramm angemeldet haben, wird automatisch die Lizenzierungsseite angezeigt.

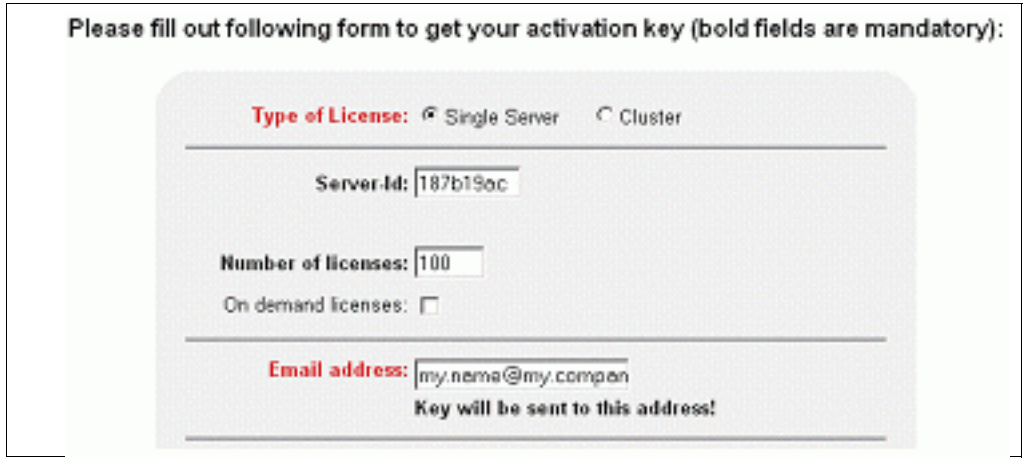
5.3.1 Standalone-Lizenzen

Ein Server mit einer WebTransactions-Installation wird für eine Maximalanzahl gleichzeitiger Benutzer lizenziert.

Um Standalone-Lizenzen neu einzugeben oder zu erhöhen, gehen Sie folgendermaßen vor:

► Drücken Sie auf der Lizenzierungsseite die Schaltfläche **Register** bzw. **Change Registration**.

Die Registrierungsseite wird geöffnet:



- Um Lizenzen für einen Standalone-Server zu registrieren, aktivieren Sie unter **Type of license** die Option **Single Server**.
- Geben Sie die Anzahl der erworbenen Lizenzen in das Feld **Number of licences** ein.
- Geben Sie Ihre eMail-Adresse und ggf. weitere Parameter an.
- Schicken Sie das Formular mit **Request Key** ab.

  Der Lizenz-Schlüssel wird Ihnen nach kurzer Zeit an die angegebene Mail-Adresse gesendet.
- Tragen Sie auf der Lizenzierungsseite in die Felder **Licenses** und **Key** die Anzahl der erworbenen Lizenzen bzw. den gültigen Lizenz-Schlüssel ein, der Ihnen per eMail zugesendet wurde, und bestätigen Sie mit **Set** und anschließend mit **Save**.

  Die Lizenzen sind aktiviert. Die neue Anzahl Lizenzen wird in der Statusleiste angezeigt.

5.3.2 onDemand-Lizenzen

Die Anzahl gleichzeitiger Benutzer, die benötigt wird, weist oft zeitlich begrenzte Spitzen auf. Mit zusätzlichen, kostengünstigen onDemand-Lizenzen können Sie solche Lastspitzen abdecken.

onDemand-Lizenzen können Sie sowohl für Standalone-Server als auch für Cluster-Lizenzen einsetzen.

Beispiel

"Wir benötigen 2000 gleichzeitige Benutzer, aber nur am Montag, sonst reichen auch 500."

Für dieses Lastprofil mussten bisher 2000 gleichzeitige Benutzer lizenziert werden, um einen zuverlässigen Betrieb gewährleisten zu können.

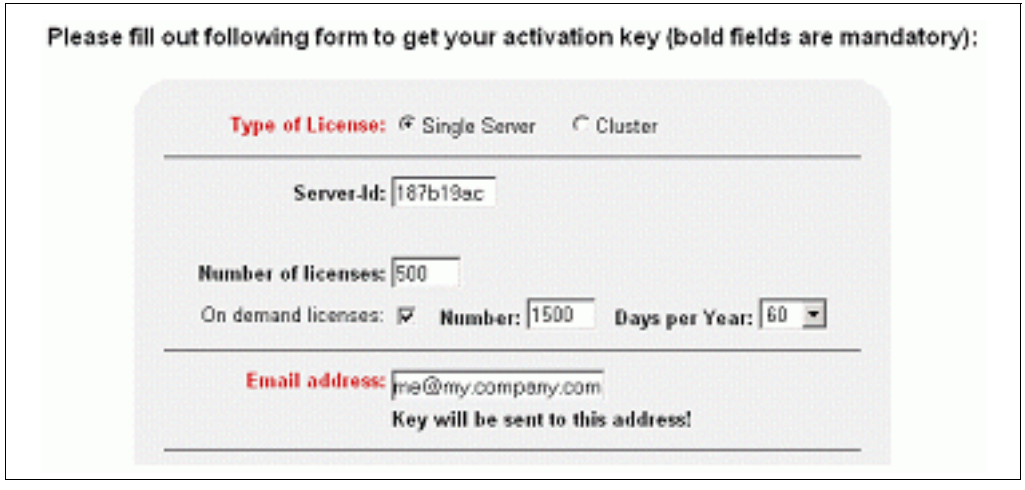
Nach dem neuen, erweiterten Lizenzmodell sind nur Lizenzen für 500 gleichzeitige Benutzer notwendig, plus1500 kostengünstigere onDemand-Lizenzen für 60 Tage im Jahr. 500 Benutzer können nun jederzeit gleichzeitig arbeiten. Sobald eine 501. Sitzung gestartet wird, wird ein Tag von den "onDemand-Tagen" abgezogen und für 24 Stunden können 2000 Sitzungen gleichzeitig laufen.

Nach Ablauf eines Jahres steht wieder die lizenzierte Anzahl von Tagen für das nächste Jahr zur Verfügung, ohne dass neu lizenziert werden muss. Das Jahr beginnt mit dem Zeitpunkt der Registrierung.

Um onDemand-Lizenzen neu einzugeben oder zu erhöhen, gehen Sie folgendermaßen vor:

- Drücken Sie auf der Lizenzierungsseite die Schaltfläche **Register** bzw. **Change Registration**.

  Die Registrierungsseite wird geöffnet:



- Geben Sie auf der Registrierungsseite an, ob Sie die onDemand-Lizenzen für einen Standalone-Server (Option **Single Server**) oder für einen Cluster (Option **Cluster**) benötigen.
- Aktivieren Sie die Option **On demand licences**.

  Das Eingabefeld **Number** für die Anzahl der onDemand-Lizenzen und eine Auswahlliste **Days per Year** für die Anzahl der onDemand-Tage (30, 60, 90 oder 120) werden eingeblendet.
- Geben Sie die Anzahl der onDemand-Lizenzen und die Anzahl der onDemand-Tage an.
- Geben Sie Ihre eMail-Adresse und ggf. weitere Parameter an.
- Schicken Sie das Formular mit **Request Key** ab.

  Der Lizenz-Schlüssel wird Ihnen nach kurzer Zeit an die angegebene eMail-Adresse gesendet.

► Tragen Sie auf der Lizenzierungsseite Folgendes ein:

Feld	**Angabe**
Servers	Anzahl der Server, für die Sie die Lizenzen benötigen
Licences	Anzahl der erworbenen Lizenzen
On Demand Licences	Anzahl der onDemand-Lizenzen
Days	Anzahl der onDemand-Tage
Key	gültiger Lizenz-Schlüssel, der Ihnen per eMail zugesendet wurde

► Bestätigen Sie die Angaben mit **Set** und anschließend mit **Save**.

Die Lizenzen sind aktiviert. Die neue Anzahl Lizenzen wird in der Statusleiste angezeigt.

Auf der Lizenzierungsseite

– im Bereich **Info** können Sie jederzeit den aktuellen Stand der verbrauchten Tage überprüfen,
– im Feld **License Logging** wird das Datum aller verbrauchten onDemand-Tage protokolliert.

5.3.3 Cluster-Lizenzen

Um ein WebTransactions-Cluster auf moderner Hardware, wie z.B. Blade-Servern, bequem und flexibel einrichten zu können, ist es notwendig, die Lizenzverwaltung unabhängig von den gerade eingesetzten Servern durchführen zu können. Hierfür gibt es Cluster-Lizenzen, die nur auf dem Cluster Controller eingetragen werden müssen und für alle am Cluster beteiligten Server gelten.

Detaillierte Informationen, wie Sie Cluster-Lizenzen eingeben und erhöhen, finden Sie im Abschnitt „Cluster-Lizenz registrieren“ auf Seite 155.

5.4 WebTransactions-Server verwalten

Wenn Sie das Administrationsprogramm zum ersten Mal aufrufen, sollten Sie zuerst die Anzahl der Lizenzen eintragen, die Sie erworben haben. Dies werden Sie wohl in der Regel nur einmal tun. Wenn Sie feststellen, dass die Lizenzen nicht ausreichen, können Sie jederzeit Lizenzen nachkaufen und auch im Administrationsprogramm die Anzahl der Lizenzen erhöhen.

Wie Sie Ihre Lizenzen registrieren, entnehmen Sie dem Abschnitt „Lizenzen eingeben oder erhöhen“ auf Seite 146.

Die Oberfläche des Administrationsprogramms ist intuitiv bedienbar. Während der Entwicklung werden Sie sich als Administrator überwiegend um Benutzer und Verzeichnisse kümmern. Jeder Anwendungsentwickler kann entsprechend der Voreinstellung des Administrationsprogramms seine eigenen WebTransactions-Anwendungen selbst verwalten. Dieses Recht muss ihm der Administrator aktiv entziehen.

Grundsätzliche Handhabung des Administrationsprogramms

Zu jedem Menüpunkt des Administrationsprogramms – z.B. **Users**, **Pools** – existieren folgende grundsätzliche Bearbeitungsmöglichkeiten:

Details, die über diese grundsätzliche Handhabung hinausgehen, entnehmen Sie den Beispielsitzungen zu den verschiedenen Host-Adapter-Varianten und der ausführlichen WebTransactions Online-Hilfe.

Anlegen, z.B. Benutzer anlegen

- Klicken Sie auf den gewünschten Menüpunkt im vertikalen Menü.

  Das zum jeweiligen Menüpunkt gehörige Fenster mit einer Liste der vorhandenen sowie Eingabefeldern für neue Komponenten, z.B. Benutzer, Pools, wird geöffnet.

- Tragen Sie in die dafür vorgesehenen Eingabefelder die gewünschte neue Komponente ein.

- Klicken Sie auf **Add**, um die neu angelegte Komponente in die Liste aufzunehmen.

Eigenschaften bearbeiten, z.B. Eigenschaften eines Benutzers bearbeiten

► Klicken Sie auf den gewünschten Menüpunkt im vertikalen Menü.

Das zum jeweiligen Menüpunkt gehörige Fenster mit einer Liste der vorhandenen Komponenten wird geöffnet.

► Klicken Sie auf die vorhandene Komponente, die bearbeitet werden soll.

Die Eigenschaften dieser Komponente werden angezeigt.

► Markieren Sie eine gewünschte oder überflüssige Eigenschaft und klicken Sie anschließend auf **Add** bzw. **Delete**.

oder

► Aktivieren bzw. deaktivieren Sie die entsprechenden Optionen, um Eigenschaften zuzuordnen bzw. zu entziehen.

Jeder Klick auf eine Option (Checkbox) schickt das Formular sofort ab.

Vorhandene Komponente löschen, z.B. Benutzer löschen

► Klicken Sie auf **Remove**, um eine vorhandene Komponente zu löschen.

Geltungsbereich

Die grundsätzliche Handhabung des Administrationsprogramms (siehe oben) gilt insbesondere für folgende Funktionen:

- Benutzer anlegen oder bearbeiten

  Nur die Benutzer, die Sie mit dem Administrationsprogramm angelegt haben, dürfen WebTransactions-Anwendungen administrieren oder mit WebLab arbeiten.

- Pool einrichten und Eigenschaften bearbeiten

  Ein WebTransactions-Pool ist ein Verzeichnis, in dem Sie mit WebLab Basisverzeichnisse erzeugen dürfen. Nur in Pools, die Sie mit dem Administrationsprogramm eingerichtet und bekannt gegeben haben, können Sie mit WebLab Basisverzeichnisse erzeugen.

- Eigenschaften von WebTransactions-Anwendungen bearbeiten

  WebTransactions-Anwendungen werden in der Regel mit WebLab mit dem Befehl **Projekt/Neu...** auf dem Server eingerichtet. Dabei erhält der erzeugende Benutzer das Administrations- und Editierrecht für diese Anwendung.

  Über die Schaltfläche **Administrate** rufen Sie die Anwendungsadministration direkt auf.

- Sitzungen

  Es wird angezeigt, wie die laufenden Sitzungen auf die lokalen Basisverzeichnisse verteilt sind. Nur die Basisverzeichnisse, in denen es laufende Sitzungen gibt, werden angezeigt.

  Über die Schaltfläche **Administrate** rufen Sie die Anwendungsadministration direkt auf. Bei Klick auf das Basisverzeichnis können die Eigenschaften der Anwendung editiert werden. Wenn die Schaltfläche bzw. der Link fehlt, ist das Basisverzeichnis nicht für die Administration über die Oberfläche eingerichtet wie z.B. die Administrations-Anwendung selbst.

- Werkzeuge definieren und deren Eigenschaften bearbeiten
- Cluster einrichten und dessen Eigenschaften bearbeiten

Siehe auch Abschnitt „Cluster einrichten“ auf Seite 157 und Abschnitt „Eigenschaften des Clusters bearbeiten“ auf Seite 157.

5.5 Cluster-Konzept

Jeder WebTransactions-Server hat, abhängig von seiner Prozessor-Leistung und dem Hauptspeicher-Ausbau, eine maximale Anzahl paralleler WebTransactions-Sitzungen, über die hinaus eine akzeptable Performance nicht mehr erreicht werden kann. Sollen weitere parallele Sitzungen auf einer WebTransactions-Anwendung ermöglicht werden, können Sie diese Sitzungen auf mehrere Server verteilen. Mit Hilfe eines WebTransactions-Clusters ist dies sehr einfach einzurichten.

Mehrere Integrations-Server können zu einem so genannten Cluster zusammengefasst werden. Diese Integrations-Server müssen nicht notwendigerweise von demselben Typ sein. Alle von WebTransactions unterstützten Integrationsplattformen, Windows, OSD und Unix-Plattformen, können beliebig miteinander kombiniert werden. Dazu muss WebTransactions auf allen Rechnern des zukünftigen Clusters installiert sein und es müssen die erforderlichen Host-Adapter auf den jeweiligen WebTransactions-Plattformen zur Verfügung stehen. Der Cluster wird über das Administrationsprogramm definiert und eingerichtet.

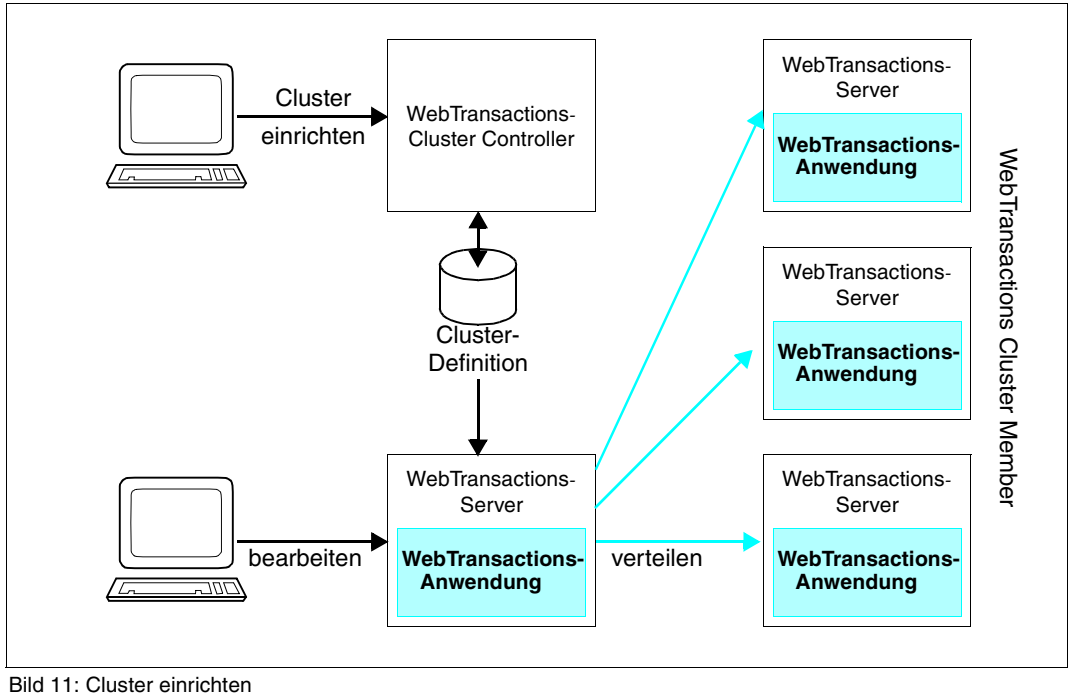


Bild 11: Cluster einrichten

Ein oder mehrere am Cluster beteiligte Server übernehmen die Rolle des Cluster Controllers, auf denen die Cluster-Definition hinterlegt wird. Die anderen Integrations-Server, auch Cluster Member genannt, erhalten je eine Kopie der Master-Anwendung. Die Master-Anwendung ist diejenige, die auf die Cluster Member verteilt wird. Sie unterscheidet sich ansonsten nicht von einer „normalen“ WebTransactions-Anwendung.

Für die Verteilung der Master-Anwendung auf die Cluster Member steht ein Assistent (Befehl **Administrieren/Anwendung verteilen**) in der Entwicklungsumgebung WebLab zur Verfügung. Dabei wird die Cluster-Definition gelesen und dann das Basisverzeichnis der Master-Anwendung auf die Cluster Member kopiert. Für eine gemeinsame Lizenzverwaltung können Sie Cluster-Lizenzen verwenden, siehe auch Abschnitt „Cluster-Lizenz registrieren“ auf Seite 155.

Vorgehen

- Zuerst erstellen Sie mit WebLab auf einem WebTransactions-Server eine Master-Anwendung.
- Danach richten Sie mit dem Administrationsprogramm einen WebTransactions-Cluster auf dem Cluster Controller ein. Der Cluster Controller ist der Rechner, auf dem die Cluster-Definition hinterlegt ist. Auf ihn weist auch der URL zum Start einer Sitzung auf dem Cluster. Der Cluster Controller entscheidet dann über das Cluster Member, auf dem die WebTransactions-Sitzung gestartet wird.
- Diese Master-Anwendung verteilen Sie mit WebLab mit dem Befehl **Administration/Anwendung verteilen** auf die Cluster Member.

5.5.1 Cluster-Lizenz registrieren

Cluster-Lizenzen gelten für alle an WebTransactions-Clustern beteiligten Server, wenn diese über den lizenzierten Server als Cluster Controller adressiert werden. Diese Rechner benötigen zwar noch die WebTransactions-Installation und das Einrichten von Applikationen, aber keine Einzelregistrierung mehr.

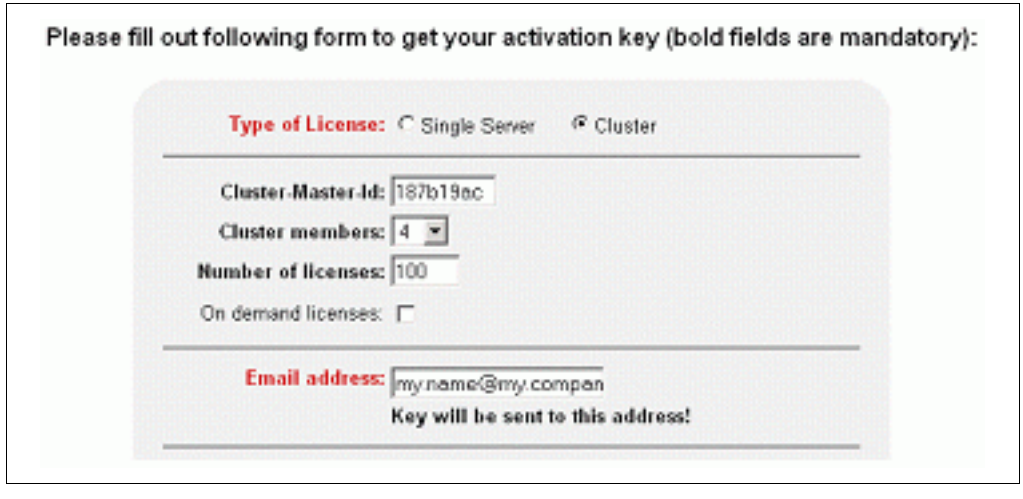
Die Anzahl der registrierten Server ist die maximale Anzahl von Rechnern, die am WebTransactions-Cluster beteiligt sein dürfen. Einzelne Cluster können auch weniger Einzelrechner enthalten und Cluster Member können in mehreren Cluster-Definitionen auftauchen.

Ein Cluster kann auch Rechner enthalten, die nicht über das Cluster lizenziert werden, sondern über individuell eingetragene Server-Lizenzen verfügen. WebTransactions bestimmt beim Verteilen der Sitzungen automatisch den zu verwendenden Lizenztyp.

Um Cluster-Lizenzen neu einzugeben oder zu erhöhen, gehen Sie folgendermaßen vor:

- Drücken Sie auf der Lizenzierungsseite die Schaltfläche **Register** bzw. **Change Registration**.

  Die Registrierungsseite wird geöffnet



- Aktivieren Sie auf der Registrierungsseite unter **Type of license** die Option **Cluster**.

  Die Auswahlliste **Cluster members** wird angezeigt.

- Wählen Sie in der Auswahlliste **Cluster members** die Anzahl der zu lizenzierenden Server aus.
- Geben Sie die Anzahl der erworbenen Lizenzen in das Feld **Number of licences** ein.
- Geben Sie Ihre eMail-Adresse und ggf. weitere Parameter an.

► Schicken Sie das Formular mit **Request Key** ab.

Der Lizenz-Schlüssel wird Ihnen nach kurzer Zeit an die angegebene eMail-Adresse gesendet.

► Tragen Sie auf der Lizenzierungsseite Folgendes ein:

Feld	**Angabe**
Servers	Anzahl der Cluster Member
Licences	Anzahl der erworbenen Lizenzen
Key	gültiger Lizenz-Schlüssel, der Ihnen per eMail zugesendet wurde

Tabelle 2:

► Bestätigen Sie die Angaben mit **Set** und anschließend mit **Save**.

Das Cluster ist lizenziert.

Der Cluster Controller wird automatisch eingetragen und kann nicht gelöscht werden. Um die Server einzutragen, die diese Cluster-Lizenz benutzen werden, gehen Sie wie folgt vor:

► Klicken Sie dazu im vertikalen Menü die Option **Clusters** an.

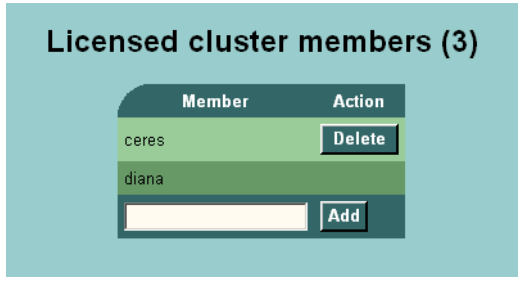
Die Liste **Licensed cluster members** wird angezeigt.

► Um einen neuen Server einzutragen, geben Sie den Servernamen in das Textfeld ein und drücken Sie **Add**.

Diese Funktion steht solange zur Verfügung, bis die Maximalanzahl von Cluster Members ausgeschöpft ist.

► Um einen einzelnen Server zu löschen, drücken Sie **Delete.**

► Speichern Sie die geänderte Konfiguration mit **Save**, um sie zu aktivieren.



5.5.2 Cluster einrichten

- Wie Sie einen Cluster einrichten, entnehmen Sie dem Abschnitt „Grundsätzliche Handhabung des Administrationsprogramms“ auf Seite 150.
- Details, die über diese grundsätzliche Handhabung hinausgehen, entnehmen Sie der ausführlichen WebTransactions Online-Hilfe.
- Die Administration, die Sie mit der Schaltfläche **Administrate** anzeigen, entspricht der Anwendungs-Administration und ist im Abschnitt „WebTransactions-Anwendung administrieren“ auf Seite 137 beschrieben.

5.5.3 Eigenschaften des Clusters bearbeiten

Sie können folgende Eigenschaften des Clusters bearbeiten:

- Verteilmethode festlegen

  Wenn mindestens ein Cluster Member über den Cluster lizenziert wird, wird als Verteilmethode automatisch Load Balancing verwendet.

  Für Session Roaming (siehe auch Abschnitt „Roaming Sessions“ auf Seite 43) wird ebenfalls immer die Verteilmethode Load Balancing verwendet, da jeder am Cluster beteiligte Rechner nach der angeforderten Sitzung durchsucht werden muss.

  Load Balancing funktioniert aus Performance-Gründen nicht mit einem HTTPS-Server. Es ist zwingend ein HTTP-Server auf jedem Rechner erforderlich, der den Zugriff auf `WTCluster.exe` und `WTPublish.exe` ermöglicht. Diesen HTTP-Server können Sie sehr einfach parallel zu einem HTTPS-Server betreiben und bei Bedarf für die Benutzung von `WTCluster.exe` und `WTPublish.exe` durch den Cluster-Master einschränken.

  Der Zugriff vom Browser auf den Cluster ist davon unabhängig und auch mit HTTPS uneingeschränkt möglich.
- Cluster Member testen oder löschen
- Neue Cluster Member zum Cluster hinzufügen

- Wie Sie die Eigenschaften eines Clusters bearbeiten, entnehmen Sie dem Abschnitt „Grundsätzliche Handhabung des Administrationsprogramms“ auf Seite 150.
- Details zur Bearbeitung der Eigenschaften des Clusters, entnehmen Sie der ausführlichen WebTransactions Online-Hilfe.

5.5.4 Cluster-Sitzung starten

Der Start einer WebTransactions-Anwendung auf einem WebTransactions-Cluster soll für den Endnutzer transparent sein. Er erhält einen festen URL und kann mit diesem die Sitzung wie auf einem einzelnen Server starten.

Eine Sitzung auf einem WebTransactions-Cluster wird über folgenden Link gestartet:

`<a href="http://`*webta-server*`/`*cgi-prefix*`/WTCluster.exe/`*cluster-id*`?[WT_SYSTEM_FORMAT=`*startTemplate*`&`*param2*`....]`

webta-server
: Cluster Controller, auf dem WebTransactions läuft

cgi-prefix
: Pfad, in dem auf dem Cluster Controller die CGI-Programme gespeichert sind

`WTCluster.exe`
: CGI-Programm für die Steuerung des Clusters, wird bei der Installation von WebTransactions im Script-Verzeichnis des Web-Servers abgelegt

cluster-id
: Name des Clusters, wie er bei der Definition festgelegt wurde

`WT_SYSTEM_FORMAT =`*startTemplate*
: Start-Template, mit der die WebTransactions-Anwendung auf dem Cluster Member gestartet wird. Ist im URL kein Start-Template angegeben, verwendet WebTransactions das in der Cluster-Definition angegebene Template.

`[&`*param2*`....]`
: Weitere optionale Parameter, die der WebTransactions-Anwendung mitgegeben werden können

Beispiel

Im folgenden Beispiel wird der Cluster `osdtest` über seinen URL aufgerufen. Ein Start-Template wird nicht angegeben. Es wird das Start-Template verwendet, das in der Cluster-Definition hinterlegt ist.

```
http://id1windhund/cgi-bin/WTCluster.exe/osdtest
```

Alternativ können Sie das Start-Template, hier `wtstart`, auch explizit angeben:

```
http://id1windhund/cgi-bin/WTCluster.exe/osdtest?WT_SYSTEM_FORMAT=wtstart
```

Für den Start einer Cluster-Sitzung aus einem Formular müssen Sie die Methode GET verwenden:

```
<FORM METHOD="GET"
      ACTION="http://webta-server/cgi-prefix/WTCluster.exe/cluster-id">
      [<INPUT TYPE=HIDDEN NAME=WT_SYSTEM_FORMAT VALUE="startTemplate">]
   weitere optionale Parameter
</FORM>
```

Für die Lastverteilung auf die verschiedenen Server ist das CGI-Programm `WTCluster.exe` auf dem Cluster Controller zuständig. Als Parameter wird der Name des Clusters angegeben.

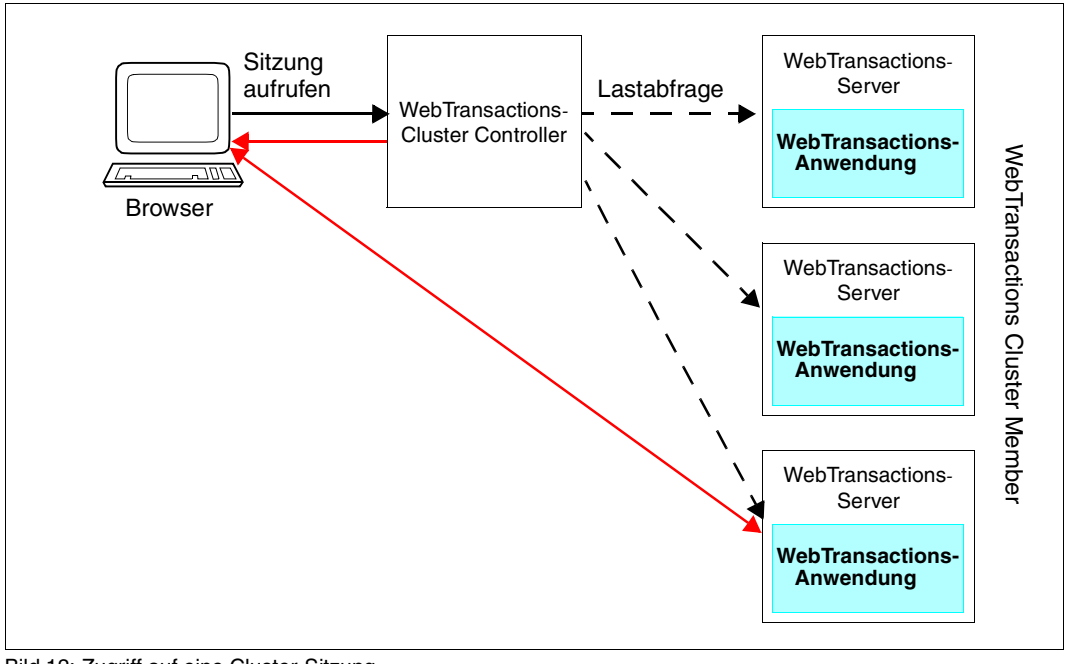


Bild 12: Zugriff auf eine Cluster-Sitzung

`WTCluster.exe` prüft entsprechend der eingestellten Methode die Auslastung der Cluster Member und gibt anschließend dem Browser den URL für den Start der „echten“ Sitzung zurück. Die weitere Kommunikation zwischen WebTransactions und dem Browser findet direkt statt, also nicht mehr über den Cluster Controller.

5.6 WebTransactions auf einem Blade Server

Ein Blade Server ist ein Komplex voneinander unabhängiger Rechner-Systeme (Server Blades). Kennzeichnend für einen Blade Server ist die Optimierung und Verkleinerung der Hardware-Komponenten, sowie die Reduktion der Ressourcen und der Kosten für die IT-Infrastruktur. Optimiert ist z.B. die Bereitstellung von Rechenleistung für das Netzwerk oder die Anzahl von CPU's pro Volumen.

Aus Sicht der WebTransactions-Anwendungen verhalten sich Blade Server wie unabhängige Rechner. Anders als in einem Multiprozessor-Rechner, in dem das Betriebssystem die Verwaltung der Prozessoren übernimmt, ist bei einem Blade-Server eine Cluster-Software oberhalb der Betriebssysteme notwendig, wenn eine logische WebTransactions-Anwendung auf verschiedene Blades verteilt werden soll.

5.6.1 Merkmale eines Blade Servers

Folgende Merkmale kennzeichnen einen Blade Server:

- Die Prozess-Module (CPU oder Server Blades) sind von den IO-Modulen (Switch oder LAN Blades) getrennt.
- Die CPU-Blades sind in Bezug auf die Prozessorleistung (z.B. Anzahl der Benutzer in einer Terminal-Server-Farm, Anzahl der Client-Requests in einer Web-Server-Farm) unabhängig skalierbar.
- Die Netzwerkverbindungen sind in Bezug auf den benötigten Datendurchsatz unabhängig skalierbar.
- Auf den CPU-Blades werden energiesparende Prozessoren und Festplatten aus der Notebook-Technologie verwendet.
- Durch die Kompaktheit und die geringe Größe der CPU-Blades und deren Komponenten wird eine hohe Verdichtung erreicht.
- Auch Komponenten wie die Verwaltungs-Einheit (Management Blade) und die Stromversorgungs-Einheit (power supply unit) sind von den CPU-Modulen getrennt.
- Die Administration, die Verwaltung und die Konfiguration des einzelnen CPU-Blades wird remote über das Netzwerk durchgeführt. Es kann jedoch auch eine lokale Konsole verwendet werden.

5.6.2 WebTransactions auf einem Blade bereitstellen

Im folgenden Einsatz-Szenario wird beschrieben, wie Sie WebTransactions auf einem Blade eines Blade Servers installieren, administrieren und Lizenzen eintragen.

Dieser Einsatzfall liegt vor, wenn auf einem heterogen genutzten Blade Server auf einem Blade dauerhaft WebTransactions zur Verfügung stehen soll. Erstellung und Nutzung von WebTransactions-Anwendungen unterscheiden sich hier nicht gegenüber einem normalen Server.

5.6.2.1 Auf Linux installieren

Betriebssystem installieren und konfigurieren

- Installieren Sie das Linux-Betriebssystem.
- Verwenden Sie bei der Installation feste TCP/IP-Adressen.
- Starten Sie das Betriebssystem.
- Loggen Sie sich nach dem Hochfahren des Core-Betriebssystems auf dem Blade ein.
- Nehmen Sie eventuell benötigte Konfigurationen des Betriebssystem vor (wie z.B. das Einrichten von Benutzerkennungen, das Zuweisen von Benutzerrechten, usw.).

Web-Server installieren und konfigurieren

- Installieren Sie, falls noch nicht geschehen, die Apache Software. Der Apache HTTP-Server ist als Softwarevoraussetzung für WebTransactions erforderlich.
- Konfigurieren Sie ggf. die Apache-Software, wenn die Default-Einstellungen in der Datei `httpd.conf` geändert werden müssen.

  Für den Betrieb mit WebTransactions sollten Sie die `Keepalive`-Funktionalität mit dem Internet Explorer deaktivieren, da der Internet Explorer diese Funktion mit Apache nicht vollständig unterstützt. Sie ergänzen dazu in der Datei `httpd.conf` die Zeile im entsprechenden Abschnitt:

  ```
  BrowserMatch "MSIE" nokeepalive
  ```

- Starten Sie den Web-Server.

Java installieren (optional)

Wenn Sie über WebTransactions auch Java-Objekte ansprechen wollen,

- Installieren Sie, falls noch nicht geschehen, eine Java-Ablaufumgebung.

WebTransactions installieren

- Transferieren Sie das komprimierte Archiv mit der WebTransactions-Software auf den Linux-Rechner.
- Melden Sie sich auf dem Rechner an.
- Installieren Sie WebTransactions gemäß der Beschreibung in den Handbüchern der Host-Adapter.

5.6.2.2 Auf Windows installieren

Betriebssystem und Web-Server installieren und konfigurieren

- Installieren Sie das Windows-Betriebssystems.
- Installieren Sie dabei, falls noch nicht geschehen, auch den Microsoft Internet Information Server oder Apache (siehe Seite 161).
- Verwenden Sie bei der Installation feste TCP/IP-Adressen.
- Nehmen Sie eventuell benötigte Konfigurationen des Betriebssystem vor (wie z.B. das Einrichten von Benutzerkennungen, das Zuweisen von Benutzerrechten, usw.).

Java installieren (optional)

Wenn Sie über WebTransactions auch Java-Objekte ansprechen wollen,

- Installieren Sie, falls noch nicht geschehen, eine Java-Ablaufumgebung.

WebTransactions installieren

- Erstellen Sie eine Konfigurationsdatei für die bedienerlose Installation von WebTransactions. Die Beschreibung der Parameter zur Steuerung der Installation finden Sie in den Handbüchern der Host-Adapter.
- Installieren Sie WebTransactions gemäß der Beschreibung in den Handbüchern der Host-Adapter.

5.6.2.3 WebTransactions konfigurieren

Für die Administration und Konfiguration von WebTransactions verwenden Sie einen beliebigen Browser.

- Starten Sie das Administrationsprogramm mit folgender URL:
  `http://`*webta-server/cgi-präfix*`/WTPublish.exe/server-admin`. *webta-server* ist dabei der Name des Rechners, auf dem WebTransactions läuft, eventuell mit Portangabe, *cgi-präfix* ist der Pfad auf diesem Rechner für CGI-Programme.
- Gehen Sie wie gewohnt vor, um den Lizenzschlüssel einzutragen, Benutzer und Pools einzurichten und den Benutzern Pools zuzuweisen
  (siehe Abschnitt „WebTransactions-Server verwalten“ auf Seite 150).

5.6.3 WebTransactions auf mehreren Blades eines Blade Servers bereitstellen

In diesem Einsatz-Szenario wird beschrieben, wie Sie WebTransactions mit Hilfe eines Images schnell auf vielen Blades verfügbar machen können. Das Szenario beschreibt die automatische Vergabe von IP-Adressen, Server-Namen und die Eintragung von Lizenzen auf geklonten Blades.

Dieses Szenario liegt vor, wenn WebTransactions für eine Hochlast-Host-Anwendung oder für viele unabhängige Host-Anwendungen mit insgesamt großer Last auf einem Blade-Server angeboten werden soll.

Referenz-Blade installieren

- Installieren Sie WebTransactions als Referenz-Installation auf einem Blade.
  Sie gehen dabei vor wie im vorigen Abschnitt „WebTransactions auf einem Blade bereitstellen“ auf Seite 161 beschrieben.

Image erstellen

- Erstellen Sie ein Image der Installation.
  Dieses Image wird im Folgenden als WebTransactions-Core-Image bezeichnet.

Klonen

- Verteilen Sie das WebTransactions-Core-Image auf mehrere Blades.
- Verwenden Sie dabei feste TCP/IP-Adressen auf den Ziel-Blades (Produktiv-Blades).

WebTransactions konfigurieren

Die Produktiv-Blades haben andere Namen und IP-Adressen als der Referenz-Blade. Durch den Klonvorgang ist der Lizenzschlüssel vom Referenz-Blade auf die Produktiv-Blades kopiert worden. Da dieser Schlüssel aber nicht zur Identifikation der Produktiv-Blades passt, müssen Sie abschließend mit der WebTransactions-Administration passende Lizenz-Keys besorgen und eintragen (siehe Abschnitt „Standalone-Lizenzen“ auf Seite 146).

5.6.4 Einen WebTransactions-Cluster auf mehreren Blades eines Blade Servers bereitstellen

In diesem Szenario wird zusätzlich zum Umfang des zweiten Szenarios eine ablauffähige WebTransactions-Anwendung und eine Cluster-Definition mit dem Image-Verfahren auf mehrere Blades verteilt.

Dieses Szenario liegt vor, wenn eine Anwendung mit sehr vielen Clients betrieben wird, so dass eine Lastverteilung auf mehrere Blades notwendig ist. Eine besondere Arbeitserleichterung liefert das Image-Verfahren, wenn die Anzahl der installierten Blades dynamisch der Last angepasst wird.

Anwendung erstellen, Cluster einrichten und Anwendung verteilen

WebTransactions ist auf die durchgängige Entwicklung, Konfiguration und Administration über http-Verbindungen ausgelegt. Daher gibt es keinerlei Einschränkungen oder Besonderheiten bei der Verwendung von Blade-Servern. Wie auf anderen Plattformen und heterogenen Clustern erstellen Sie Ihre Anwendung, richten Cluster ein und verteilen die Master-Anwendung auf die Cluster Member (siehe Abschnitt „Cluster-Konzept“ auf Seite 153).

Dynamische Auslastung im Cluster sichern

In vielen Web-Applikationen gibt es schwankende Auslastungen. Um zu Spitzenzeiten gute Performance zu garantieren, muss ein entsprechender Cluster auf hinreichend vielen Blades eingerichtet werden. Oft ist es aber sinnvoll, einige Blades in den Zeiten geringer Last mit anderen Aufgaben zu beschäftigen. Sie erreichen dies, indem Sie die jeweils benötigten Images auf den Blades einspielen.

Sie gehen dazu wie folgt vor:

- Installieren Sie WebTransactions auf den maximal benötigten Blades wie im Abschnitt „WebTransactions auf mehreren Blades eines Blade Servers bereitstellen“ auf Seite 163 beschrieben.

- Legen Sie auf einem oder mehreren Blades, die dauerhaft installiert bleiben, eine Definition für einen WebTransactions-Cluster an, der die Verteilung der Requests nach der Methode `Take first with load lower x%` übernimmt (siehe Abschnitt „Eigenschaften des Clusters bearbeiten“ auf Seite 157).

  Die Methode `Take first with load lower x%` ist hier sinnvoll, weil sie nicht aktive Cluster Member unberücksichtigt lässt. Sie können bei dieser Vorgehensweise Blades abschalten. Der Cluster arbeitet ohne Probleme weiter, solange noch wenigstens ein CPU-Blade aktiv ist.

- Tragen Sie als Cluster Member alle Blades (mit Rechnernamen oder IP-Adresse) ein, die für einen Betrieb dieses Clusters in Frage kommen.
- Verteilen Sie die WebTransactions-Anwendung mit WebLab nun auf alle Blades.
- Sichern Sie von allen Blades, die zeitweise andere Aufgaben übernehmen sollen, ein Image (siehe Abschnitt „Image erstellen“ auf Seite 163).
- Stellen Sie später diese Blades jeweils mit dem gesicherten Image wieder her. Sie ersparen sich so die erneute Eingabe eines Lizenzschlüssels, weil der jeweils passende Schlüssel im gesicherten Image korrekt eingetragen ist.

6 Die Entwicklungsumgebung WebLab

WebLab ist die WebTransactions-Entwicklungsumgebung. Mit WebLab können Sie sowohl eine WebTransactions-Anwendung erstellen als auch Templates nachbearbeiten und das Ergebnis testen. WebLab wird ergänzt durch einen beliebigen Web-Browser, in dem Sie das Ergebnis Ihrer Nachbearbeitungen sofort überprüfen können (WYSIWYG).

Zusätzlich bietet WebLab eine Import-/Export-Schnittstelle zu einem HTML-Editor Ihrer Wahl. Alle HTML-Tags können unter der WebLab-Oberfläche eingefügt und komfortabel bearbeitet werden. Für das Einfügen und Bearbeiten der gängigsten HTML-Tags stellt WebLab Assistenten zur Verfügung.

WebLab, der HTML-Editor und der Browser laufen auf einem PC unter Windows.

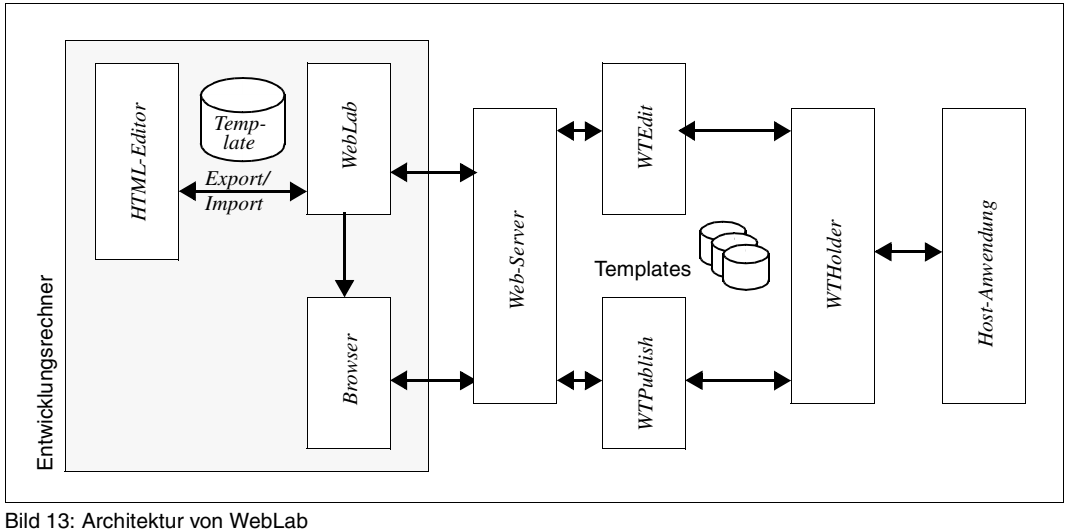


Bild 13: Architektur von WebLab

WebLab und Web-Browser werden über eine im Hintergrund laufende WebTransactions-Sitzung gekoppelt und koordiniert. Diese ermöglicht es, WebTransactions-Anwendungen „on-the-fly“ anzupassen und die aktuellen Werte und Einstellungen zu überprüfen oder auch explizit neu zu setzen.

WebLab

WebLab ist die zentrale Schaltstelle der Entwicklungsumgebung. Es startet und verwaltet eine WebTransactions-Sitzung, die im Web-Browser abläuft. WebLab kann die Dateien (Templates und Verwaltungsdateien) dieser WebTransactions-Sitzung laden und speichern. Mit WebLab kann der Anwender auf die Objekte der aktuellen Sitzung zugreifen, diese ansehen und ggf. ändern.

Web-Browser

Für die Navigation in der WebTransactions-Anwendung und die Darstellung der Templates wird der Browser verwendet, den Sie mit dem Befehl **Optionen/Einstellungen** festgelegt haben. Auf eine fest in WebLab eingebaute WYSIWYG-Komponente wurde bewusst verzichtet, da sich das Aussehen einer HTML-Seite je nach verwendetem Browser ändern kann. Sie können für die Entwicklung den Browser verwenden, der auch für die Einsatzphase vorgesehen ist, und können auch Browser-spezifische Erweiterungen nutzen. Für das Anzeigen des aktuellen Templates im Browser ist nur ein Klick auf einen WebLab-Menüpunkt notwendig. Änderungen im Layout einer Seite lassen sich so unmittelbar ansehen, ohne dass alle Dialogschritte wieder durchlaufen werden müssen, die ursprünglich zur Anzeige dieser Seite geführt haben.

HTML-Editor

WebLab kann optional durch einen HTML-Editor ihrer Wahl ergänzt werden.

Der von Ihnen gewählte HTML-Editor muss allerdings folgende Voraussetzungen erfüllen:

- HTML-Kommentare müssen unverändert bleiben
- Unbekannte Tags sollten nicht automatisch repariert werden oder diese Funktionalität muss ausschaltbar sein.

WebLab stellt Funktionen zur Verfügung, um Templates für die Arbeit in dem HTML-Editor aufzubereiten. Die Kopplung zwischen HTML-Editor und WebLab erfolgt mit einem Export-/Importmechanismus.

6.1 Funktionalität von WebLab

Um Ihnen die Integration der Host-Anwendung und die Modernisierung Ihrer Oberflächen zu erleichtern, bietet WebLab unter anderem folgende Funktionalität:

- Einfache und schnelle Integration der vorhandenen Host-Anwendungen durch:
  - Anlegen von Projekten
  - Erstellen eines Basisverzeichnisses
  - Automatische Generierung der erforderlichen Templates
  - Unterstützung beim Erstellen eines anwendungs-spezifischen Start-Templates
- Komfortable Nachbearbeitung der Templates durch:
  - Unterstützung der Eingabe und Änderung von WTML-Tags über Dialogfelder für die Parametereingabe
  - Die verschiedenen Sprachmittel, die Sie in einem Template verwenden können, werden zur schnellen Identifizierung durch unterschiedliche Farben markiert.
  - Unterstützung bei der Eingabe und Änderung von HTML-Tags über Dialogfelder für die Parametereingabe
  - Unterstützung bei der Eingabe und Änderung von WTBeans über Dialogfelder für die Parametereingabe
    Auch bereits im Template enthaltene WTBeans können bearbeitet werden: Es wird das gleiche Dialogfeld wie bei der Eingabe des WTBean geöffnet, die Felder sind aber mit den aktuellen Werten vorbelegt. Die verschiedenen Elemente der WT-Beans werden zur schnellen Identifizierung mit unterschiedlichen Farben hinterlegt.
  - Unterstützung beim Einbringen von Kommentaren und die Möglichkeit, aus den Kommentaren eine HTML-Dokumentation für die Templates zu generieren
  - Direkten Zugriff auf die Objekte der WebTransactions-Sitzung: die Objekte werden in WebLab in einem eigenen Fenster dargestellt. Objekt- und Attributnamen (z.B. für Host-Datenobjekte, die von der Host-Anwendung kommen) können mit der Maus in Templates eingefügt werden.
  - Variablenüberwachung in einem eigenen Fenster
  - Auswahl der Host-Datenobjekte in einer grafischen Darstellung des Originalschirms
  - Öffnen der referenzierten Includes und Klassen-Templates in Ihrem Template per Mausklick

  - Assistenten zum schnellen Bereitstellen typischer grafischer Dialogführungstechniken (z.B. Drop-Down-Listen oder Radiobuttons). Ein weiterer Assistent unterstützt Sie, wenn Sie aus einer HTML-Seite ein Template erstellen wollen. WebLab fügt die WTML-Sprachelemente, die zum Datenaustausch und der Kommunikation mit dem Host erforderlich sind, in die HTML-Seite ein.

- Unterstützung beim Testen einer WebTransactions-Anwendung durch
  - Testen der Gestaltung eines Templates (siehe Abschnitt „Gestaltung eines Templates testen“ auf Seite 206)
  - Einzelschrittverfolgung (siehe Abschnitt „Ablauf im Template testen“ auf Seite 209)
  - spezielle Start-Templates
- Erweiterte Sicherheit durch enge Verzahnung mit dem Administrationsprogramm
- Einfaches Transferieren von ganzen WebTransactions-Anwendungen

6.2 Erste Schritte

WebLab läuft auf einem Windows-System. WebTransactions und damit der Integrations-Server und die Host-Anwendung laufen auch auf anderen Plattformen ab. Die Integration der Host-Anwendung kann dezentral vorgenommen werden, ohne dass Sie sich auf dem Integrations-Server einloggen müssen. Die benötigten Daten werden über das HTTP- oder HTTPS-Protokoll über das Netz zur Verfügung gestellt.

In den folgenden Abschnitten ist beschrieben, wie Sie Ihre Host-Anwendung schnell und einfach an das WWW anschließen können.

Das generelle Konzept der Nachbearbeitung ist im Abschnitt „Templates bearbeiten“ auf Seite 183 beschrieben. Eine Beispielsitzung, die auch einen Nachbearbeitungszyklus umfasst, finden Sie in den Handbüchern zu den einzelnen Liefereinheiten von WebTransactions.

6.2.1 Projekt anlegen

Das Projekt ist der Haupteinstiegspunkt in WebLab. In der Projektdatei sind die wichtigsten Daten gespeichert, die WebLab beim Arbeiten mit einer WebTransactions-Anwendung benötigt, z.B. Daten des WebTransactions-Servers. Um die individuellen Sitzungseinstellungen für eine WebTransactions-Anwendung dauerhaft speichern zu können, verwenden Sie ein Projekt.

Alle Befehle zum Anlegen und Verwalten von Projekten finden Sie im Menü **Projekt**.

Sie können WebLab so konfigurieren, dass das zuletzt verwendete Projekt beim Start von WebLab sofort geladen wird.

Eine detaillierte Beschreibung aller projektbezogenen Funktionen von WebLab finden Sie in der ausführlichen Online-Hilfe von WebLab. Sie gelangen zu diesem Kapitel mit dem Befehl **?/Hilfethemen**.

- Um ein neues Projekt zu erstellen, wählen Sie den Befehl **Projekt/Neu...** und bestätigen Sie die Abfrage, ob ein Basisverzeichnis erstellt werden soll, mit **Ja**.

  Das Dialogfeld **Verbinden** wird geöffnet.
- Gehen Sie anschließend vor, wie in den nachfolgenden Abschnitten dargestellt:
  - Basisverzeichnis erstellen, siehe Seite 172
  - Automask-Template erstellen, siehe Seite 173
  - Individuelles Start-Template erzeugen, siehe Seite 173

Eine detaillierte Anleitung dazu finden Sie in den Handbüchern zu den einzelnen Liefereinheiten von WebTransactions sowie in der ausführlichen Online-Hilfe von WebLab.

6.2.1.1 Basisverzeichnis erstellen

Das Erstellen eines Basisverzeichnisses und das Eröffnen einer Sitzung sind die wichtigsten Schritte bei der Integration einer Host-Anwendung.

Ein Basisverzeichnis erstellen Sie auf dem Rechner, auf dem WebTransactions läuft. WebLab baut dazu eine Verbindung zum WebTransactions-Rechner auf und startet dort das CGI-Programm `WTEdit`. `WTEdit` liest die notwendigen Informationen aus dem Installationsverzeichnis von WebTransactions und erstellt daraus ein Basisverzeichnis.

Beachten Sie, dass Sie vorher mit dem Administrationsprogramm von WebTransactions-Benutzer und mindestens einen Pool für WebTransactions-Anwendungen eingerichtet und freigegeben haben müssen, siehe hierzu auch Kapitel „WebTransactions-Server“ auf Seite 141.

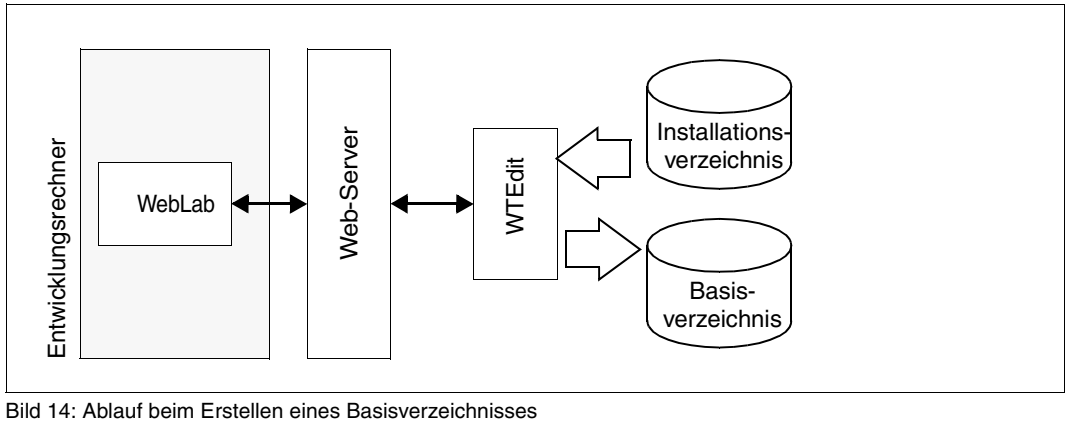


Bild 14: Ablauf beim Erstellen eines Basisverzeichnisses

- Bauen Sie die Verbindung zum Server auf (Dialogfeld **Verbinden**).

  Legen Sie dazu die für das Öffnen einer Verbindung notwendigen Parameter (für **URL von WTPublish** sowie **URL von WTEdit**) fest.

- Geben Sie anschließend die Eigenschaften Ihres Basisverzeichnisses an (Dialogfeld **Basisverzeichnis erstellen**).

  Ein neues Basisverzeichnis wird unter dem eingegebenen Namen im ausgewählten Pool erstellt.

6.2.1.2 Automask-Template erstellen (OSD, MVS)

► Wenn Sie das Dialogfeld **Basisverzeichnis erstellen** mit **OK** bestätigen, wird das Dialogfeld **Automask generieren** eingeblendet.

Das Automask-Template regelt bei Host-Anwendungen über die Protokolle 9750 (OSD) und 3270 (MVS) die automatische Umsetzung der Formate in die Browser-Darstellung, d.h. ohne Capture-Verfahren und Nachbearbeitung der Formate werden diese eins zu eins im Browser nachgebildet.

► Diese automatische Umsetzung steuern Sie mit den Parametern im Dialogfeld **Automask generieren**.

Anschließend besteht die Möglichkeit, für eines der beim Generieren ausgewählten Kommunikationsobjekte ein Start-Template zu erzeugen.

6.2.1.3 Individuelles Start-Template erstellen

Sie können sofort ein individuelles Start-Template für Ihre Host-Anwendung generieren. WebLab öffnet dann ein Dialogfeld, in dem Sie Angaben zur Verbindung zu Ihrer Host-Anwendung machen.

Auf der Registerkarte **Verbindungsparameter** finden Sie alle obligatiorischen Einstellungen, z.B. den Namen der Host-Anwendung oder den Namen des Rechners, auf dem die Host-Anwendung läuft.

6.2.2 Projekt speichern

Sie können ein Projekt jederzeit auf Ihrer Festplatte abspeichern (**Projekt/Speichern** bzw. **Speichern unter...**).

Ist das Projekt noch nicht gespeichert, wenn Sie das Projekt oder WebLab schließen, werden Sie gefragt, ob Sie das Projekt speichern wollen.

6.2.3 Sitzung starten

- Öffnen Sie mit dem Befehl **Datei**/**Sitzung starten** das Dialogfeld **Sitzung starten**.
- In diesem Dialogfeld können Sie die für das Öffnen einer Sitzung notwendigen Parameter eingeben:
  - Hier haben Sie die Möglichkeit, die Parameter für die Verbindung zum Server zu editieren über **URL von WTPublish**.
  - Darüber hinaus legen Sie weitere für die Sitzung relevante Parameter und Einstellungen fest, z.B. Basisverzeichnis, Start-Template...

Wie Sie dabei im Detail vorgehen ist in den Handbüchern zu den einzelnen Lifereinheiten von WebTransactions sowie in der ausführlichen Online-Hilfe von WebLab beschrieben.

6.3 Benutzeroberfläche von WebLab

Dieser Abschnitt gibt Ihnen einen kurzen Einstieg in die Benutzeroberfläche von WebLab. Genauere Informationen über WebLab finden Sie in der Online-Hilfe zu WebLab.

6.3.1 Das Hauptfenster

Wenn Sie WebLab mit dem Befehl **Start/Programme/WebTransactions 7.5/WebLab** aufrufen, wird das Hauptfenster eingeblendet:

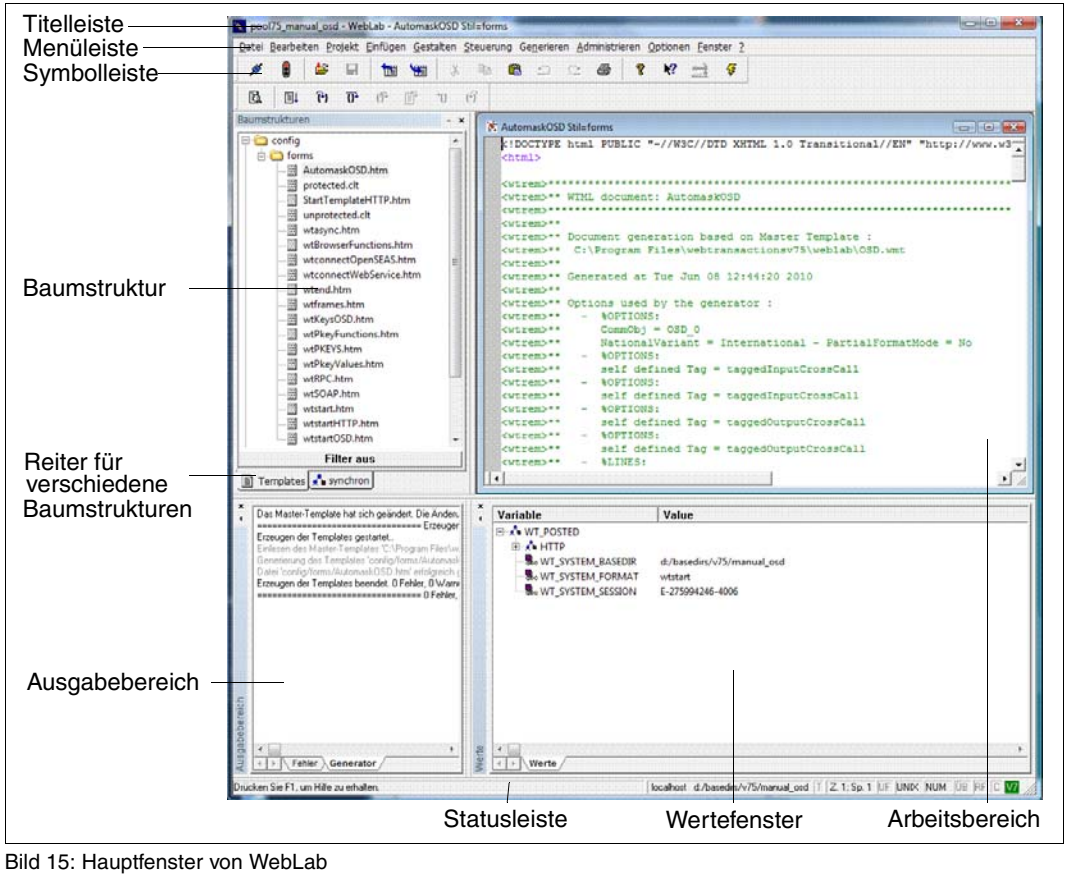


Bild 15: Hauptfenster von WebLab

Das Hauptfenster besteht von oben nach unten betrachtet aus einer Titelleiste mit dem Namen des aktuellen Templates, einer Menüleiste und einer Symbolleiste.

Der Inhalt des Hauptfensters ist in zwei Bereiche unterteilt:

- Die linke Seite enthält verschiedene **Baumstrukturen**, in der die Objekte des aktuellen Templates und die Dateien des Basisverzeichnisses nach Art des Explorers als Baum dargestellt sind.

  Sobald eine Verbindung besteht, wird der Template-Baum mit den im Basisverzeichnis der WebTransactions-Anwendung vorhandenen Dateien angezeigt. Andernfalls ist die Baumstruktur leer.

  Wenn eine Sitzung gestartet ist, wird in der Baumstruktur der Objektbaum des aktuell ausgeführten Templates angezeigt.

- Die rechte Seite dient als **Arbeitsbereich**. Hier werden die Templates zum Bearbeiten angezeigt. Auch die Ausgaben des Template-Baums werden im Arbeitsbereich angezeigt. Beim Start von WebLab ist der Arbeitsbereich zunächst leer.

- Unterhalb des Arbeitsbereichs werden während der Arbeit mit WebLab im **Ausgabebereich** Status- oder Fehlermeldungen in verschiedenen Registerkarten angezeigt:

  – für Ausgaben des Generators (z.B. beim Generieren des Basisverzeichnisses oder formatspezifischer Templates)

  – für Fehlermeldungen

  – für die Verteilung von WebTransactions-Anwendungen

  – für die Anzeige der Variablen und Werte während der Einzelschrittverfolgung

  – für die Ausgaben der Werkzeuge auf dem Server

  Zwischen diesen Fenstern können Sie mit den Reitern blättern.

- Ebenfalls im unteren Bereich befindet sich das **Wertefenster** zur Anzeige von Variablen und Objekten und deren Werten.

Am unteren Rand des Hauptfensters befindet sich die Statusleiste. In der Statusleiste wird kontextsensitiv ein kurzer Hilfetext, der verbundene Rechner und das verwendete Basisverzeichnis angezeigt. Wenn Sie ein Template bearbeiten, werden zusätzlich die Zeilen- und Spaltennummer angezeigt, in der sich der Cursor im Template befindet.

6.3.2 Die Baumstruktur

Am unteren Ende der Baumstruktur befinden sich Reiter der Registerkarten, mit denen Sie zwischen verschiedenen Baumstrukturen wechseln können. Wieviele Registerkarten dargestellt werden, ist abhängig von der aktuellen Sitzung.
Folgende Registerkarten sind möglich:

Templates
: Baumstruktur des Basisverzeichnisses

synchron
: Objektbaum des synchronisierten Dialogs

asynchron
: Objektbaum des nicht synchronisierten Dialogs

remote
: Objektbaum der Zugriffe über WT_REMOTE

6.3.2.1 Der Template-Baum

Sobald eine Verbindung aktiv ist, werden alle Dateien im Basisverzeichnis in Baumform angezeigt. Die Handhabung der Baumstruktur entspricht der im Windows-Explorer: Sie können Hierarchiestufen aufklappen und wieder schließen. Über einen Filter können Sie Dateien aus der Darstellung ausblenden. Einzelne Dateien markieren Sie per Mausklick.

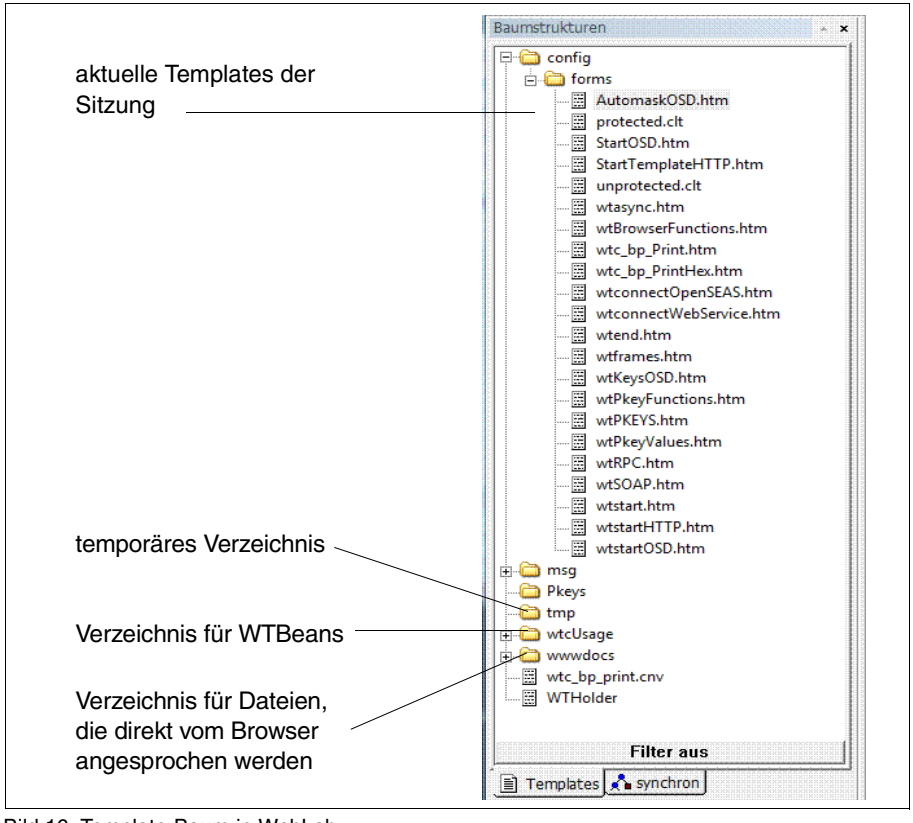


Bild 16: Template-Baum in WebLab

Kontextmenü

Wenn Sie eine Datei im Template-Baum markieren und die rechte Maustaste drücken, öffnet sich ein Kontextmenü mit Befehlen für die aktuelle Datei.

Über das Kontextmenü können Sie z.B. den Template-Browser aktivieren (**Kontext anzeigen**), der die verschiedenen Beziehungen zwischen den Templates grafisch darstellt.

Eine detaillierte Beschreibung der einzelnen Menüpunkte finden Sie in der ausführlichen Online-Hilfe von WebLab.

Dateiauswahl

Es gibt folgende zwei Möglichkeiten, die Anzeige im Template-Baum zu filtern:

- nach Dateiendung
  Es werden entweder Dateien angezeigt, die eine der angegebenen Endungen haben oder die keine der angegebenen Endungen haben.
- nach Erstellungsdatum
  Es werden entweder Dateien angezeigt, deren Erstellungsdatum vor oder nach dem angegebenen Datum liegt. Ebenso lässt sich ein Zeitraum (von – bis) einstellen.

Beide Filterarten lassen sich wahlweise als UND- oder ODER-Verknüpfung verbinden.

Damit ist es z.B. möglich, nur Dateien mit der Endung .bak, die älter als ein Monat sind, anzuzeigen.

Sie haben unterhalb des Template-Baumes die Möglichkeit, über die Schaltfläche **Filter aus/ein** die Filtereinstellungen zu ändern und aktive Filter anzuzeigen.

Wie Sie einen neuen Filter einrichten oder einen vorhandenen Filter editieren**,** entnehmen Sie der ausführlichen Online-Hilfe zu WebLab.

6.3.2.2 Die Objektbäume

Sobald eine Sitzung gestartet wurde, werden alle Objekte und Variablen der aktuellen Sitzung in Baumstruktur angezeigt.

Variablen, die global für eine WebTransactions-Sitzung zur Verfügung stehen, sind blau gekennzeichnet. Das sind diejenigen Attribute, die aus WebTransactions-Modulen importiert wurden. Auf der Registerkarte **Objektbaum** (Befehl **Optionen**/**Einstellungen**) können Sie die Sortierung des Objektbaums ändern und sitzungsglobale Attribute aus dem Objektbaum ausblenden.

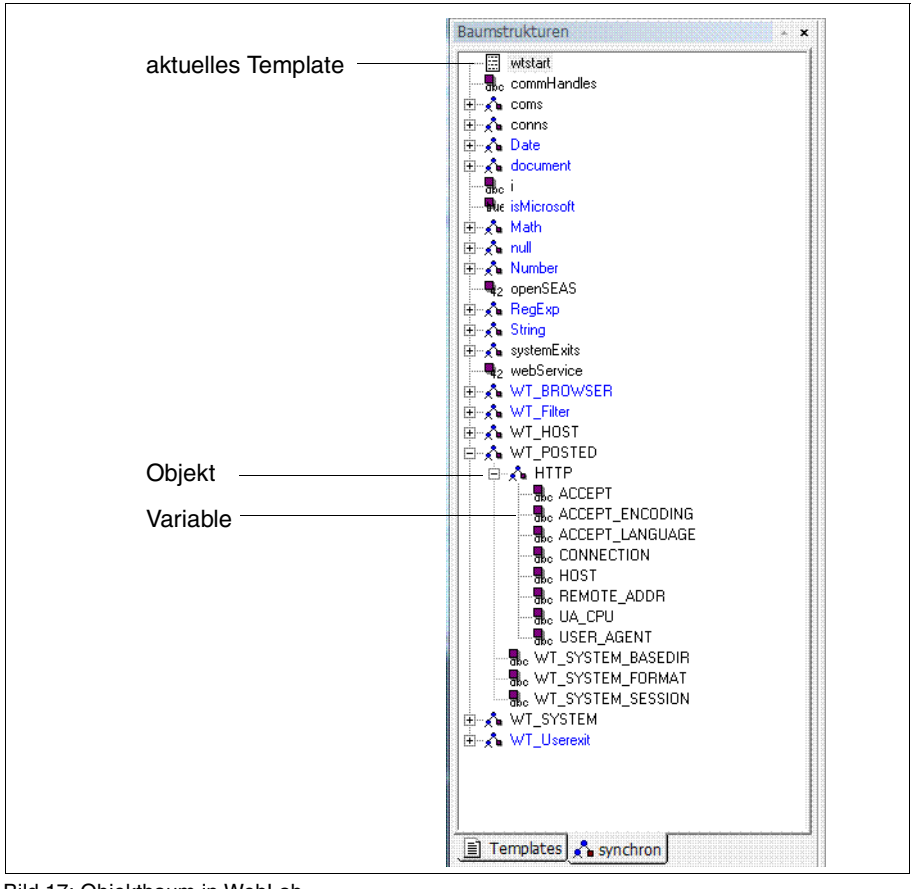


Bild 17: Objektbaum in WebLab

Die Werte der Objekte und Variablen können Sie über den Befehl **Eigenschaften** im Kontextmenü ansehen und ändern. Objekte und Variablen selbst können Sie durch Ziehen mit der Maus in das aktuelle Template einfügen. Was in das Template eingefügt wird, legen Sie mit folgenden Tastenkombinationen fest:

einfaches Verschieben
: fügt den vollständigen Variablennamen in das Template ein

Verschieben mit gedrückter Shift-Taste
: fügt den Attributnamen in das aktuelle Template ein

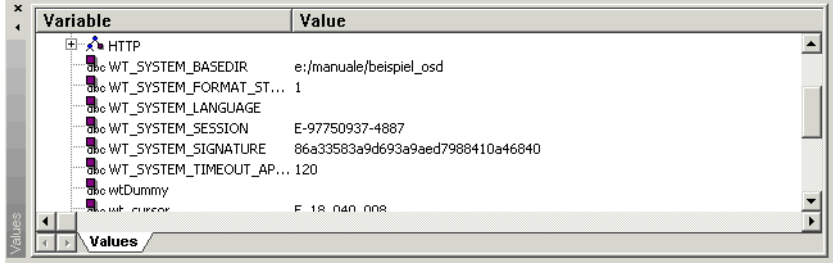
Verschieben mit gedrückter Alt-Taste
: fügt die Variablen mit Auswertungsoperator in das aktuelle Template ein

Verschieben mit gedrückter Crtl-Taste
: fügt den Wert der Variablen in das aktuelle Template ein

6.3.3 Das Wertefenster

Das Wertefenster stellt Variablen und ihre Werte dar und ist in einem eigenen, andockbaren Fenster realisiert. Sie können Variablen und Objekte mit der Maus aus dem Objektbaum und der grafischen Hostobjektauswahl hineinziehen.

Die Objekte werden in einer zweispaltigen Liste dargestellt, wie abgebildet:



Bei Objekten, die einen Wert haben, wird in der linken Spalte der Variablenname inklusive Pfad und in der rechten Spalte der Wert angezeigt. Durch Doppelklick auf einen Wert, können Sie diesen auch in einem eigenen Fenster anzeigen lassen; lange Werte werden dort an geeigneten Stellen umgebrochen.

Bei Objekten, die eine Strukur haben, wird nur der Variablenname in der linken Spalte angezeigt und zusätzlich ein +-Zeichen zum Öffnen der Struktur.

Wie Sie mit dem Wertefenster arbeiten, entnehmen Sie der ausführlichen Online-Hilfe zu WebLab.

6.4 Templates generieren

Mit WebLab können Sie folgende Templates automatisch generieren:

- Automask-Templates (OSD, MVS)
- Formatspezifische Templates durch das Capture-Verfahren (OSD, MVS)
- Formatspezifische Templates aus IFG-Beschreibungen (OSD, openUTM)
- Templates für einfachen Zugriff auf Web-Services

Wie Sie dabei im Detail vorgehen ist in den Beispielsitzungen der Handbücher zu den einzelnen Liefereinheiten von WebTransactions beschrieben.

6.5 Templates bearbeiten

Templates können Sie mit WebLab direkt bearbeiten. Sie müssen nicht explizit auf den Entwicklungs-PC übertragen werden. Dies erledigt die Entwicklungsumgebung automatisch für Sie.

6.5.1 Allgemeines Vorgehen

Um Templates auf dem Server zu bearbeiten, gehen Sie folgendermaßen vor:

- Öffnen Sie mit dem Befehl **Datei/Sitzung starten** das Dialogfeld **Sitzung starten**.
- Bestätigen Sie das Dialogfeld mit **OK**. WebLab startet den Web-Browser, in dem dann die WebTransactions-Sitzung läuft. Nach dem Start einer Sitzung steht Ihnen auch der Objektbaum zur Verfügung. Er wird mit jedem Laden des aktuellen Templates und jedem Befehl **Steuerung/Im Browser aktualisieren** erneut vom Server geladen.
- Navigieren Sie im Browser zu dem Bildschirmformat, das Sie verändern wollen.
- Laden Sie das dazugehörende Template in WebLab mit dem Befehl **Datei/aktuelles Template öffnen**.
- Bearbeiten Sie das Template.
- Wählen Sie **Steuerung/Im Browser aktualisieren,** um sich die Auswirkungen Ihrer Änderungen anzusehen. WebLab stellt dann das Template erneut im Browser dar.

Eine detaillierte Beschreibung aller WebLab-Funktionen finden Sie in der ausführlichen Online-Hilfe von WebLab.

6.5.2 Templates gestalten

Die formatspezifischen Templates sind dem Aussehen und der Funktionalität der Terminal-Darstellung nachempfunden. Wollen Sie das generierte Template aufbereiten und z.B. ein Auswahlmenü in eine Drop-Down-Liste umwandeln, müssen Sie die Templates nachbearbeiten. Für die Nachbearbeitung steht Ihnen neben den üblichen Mitteln der Web-Seiten-Gestaltung die Template-Sprache WTML zur Verfügung, die Sie im WebTransactions-Handbuch „Template-Sprache“ ausführlich beschrieben finden.

6.5.2.1 Templates editieren

Bei der individuellen Gestaltung der Templates sind Ihnen keine Grenzen gesetzt: Animierte und anklickbare Bilder, Java-Scripts und -Applets, ActiveX-Controls, Bild- und Tonsequenzen etc. können eingefügt werden. Sie können alle Sprachmittel verwenden, die der eingesetzte Web-Browser unterstützt.

Beim Einfügen und Bearbeiten der wichtigsten HTML-Tags werden Sie von WebLab unterstützt (Befehle **Einfügen/HTML** bzw. **Bearbeiten/Tag bearbeiten**). Sie können die Schreibweise der eingefügten Tags beeinflussen (Befehl **Optionen**/**Einstellungen**/Registerkarte **Format**). Wenn Sie hier die Option **HTML-Tags XHTML-konform erzeugen** auswählen, erzeugt WebLab alle HTML-Tags XHTML-konform.

Sie können die HTML-Oberfläche aber auch mit einem HTML-Editor unabhängig von den generierten Templates entwerfen und anschließend über einen Assistenten mit dem Befehl **Gestalten/Zusammenführen** von WebLab in Templates umwandeln.

Mit den WTBeans bietet WebTransactions eine Sammlung von wiederverwendbaren Komponenten, mit denen Sie sowohl die Darstellung ihrer Daten im Browser als auch die Kommunikation mit der Host-Anwendung steuern können. Weitere Informationen finden Sie im nächsten Abschnitt und im Abschnitt „WTBeans“ auf Seite 57.

Mit den Befehlen im Menü **Einfügen**/**WTScript** fügen Sie Prototypen für alle Funktionen und Klassen von WTScript in ein Template ein.

6.5.2.2 Snippets einfügen

Snippets sind Textbausteine, die Sie unverändert in Templates übernehmen können. Snippets werden zusammen mit den benutzerspezifischen Daten an Ihrem PC gespeichert.

Um die vorhandenen Snippets anzuzeigen, wählen Sie den Befehl **Optionen**/**Snippets anzeigen**. Mit einem Doppelklick auf das betreffenden Snippet fügen Sie es an der Cursor-Position in Ihr Template ein.

Das Kontextmenü im Snippet-Fenster bietet Ihnen außerdem Funktionen, um Snippets zu bearbeiten, umzubenennen, zu löschen, sowie neue Ordner und neue Snippets anzulegen.

6.5.2.3 WTBeans einfügen

Abhängig davon, ob Sie ein standalone oder inline WTBean einfügen wollen, unterscheiden sich die Befehle.

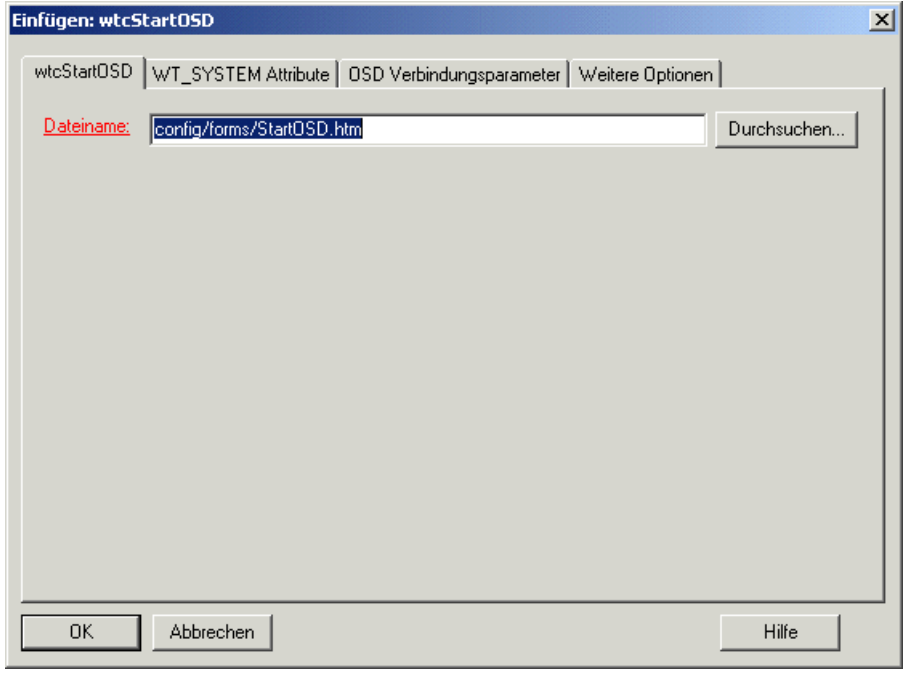
Um WTBeans einzufügen oder zu bearbeiten, muss eine Verbindung zum WebTransactions-Server bestehen.

Beim Download von WebTransactions können Sie weitere WTBeans als Goodies mitbeziehen. Diese zusätzlichen WTBeans müssen eigens installiert werden, siehe hierzu auch die zugehörige Dokumentation. Nach der Installation und dem Neustart von WebLab werden die zusätzlichen standalone WTBeans ebenfalls im Untermenü angezeigt.

Standalone WTBeans

- Um ein standalone WTBean einzufügen, verbinden Sie sich zuerst mit einer WebTransactions-Anwendung.
- Wählen Sie dann in WebLab den Befehl **Datei/Neu**. In einem Untermenü werden Ihnen die verfügbaren WTBeans angezeigt, aus denen Sie wählen können. Mit ausgeliefert werden die standalone WTBeans zur Erstellung eines Start-Templates.

  WebLab bietet zum Bearbeiten der Parameter aller WTBeans eine einheitliche Oberfläche, die hier am Beispiel des mitausgelieferten WTBeans `wtcstartOSD` dargestellt ist.



Die möglichen Parameter können Sie in einem Dialogfeld in einer oder mehreren Registerkarten bearbeiten. Die Namen der Parameter, für die Sie einen Wert angeben müssen, sind rot eingefärbt. Alle weiteren Parameter werden mit Vorbelegungen versorgt, wenn Sie nichts angeben.

► Bestätigen Sie Ihre Angaben mit **OK**. Aus Ihren Angaben und der Beschreibungsdatei wird das entsprechende Template erzeugt und im Arbeitsbereich von WebLab angezeigt, siehe hierzu auch Abschnitt „Darstellung der WTBeans im Template“ auf Seite 187.

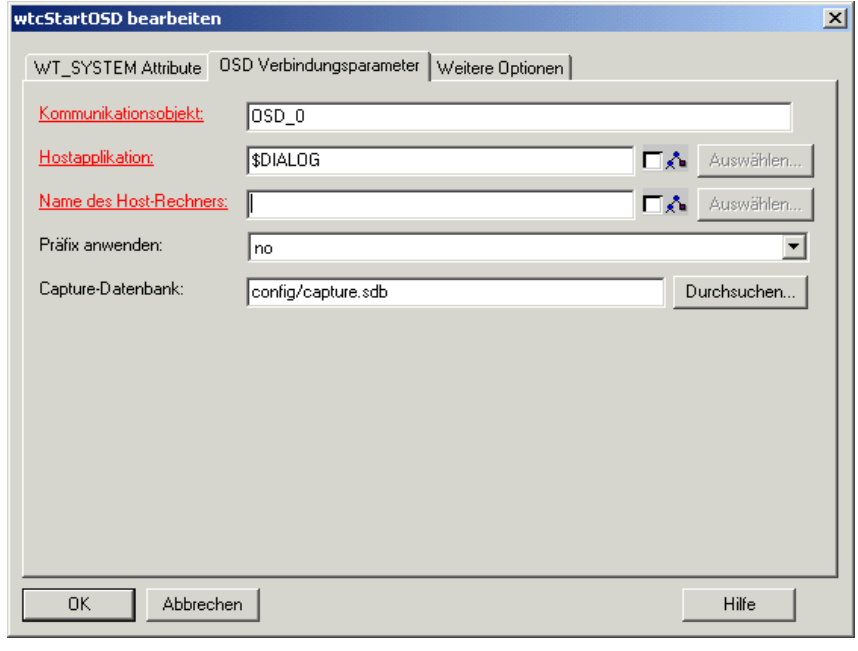
Das erzeugte Template wird unter dem Pfad gespeichert, den Sie in der ersten Registerkarte des WTBean eingeben. Die Beschreibungsdatei für das verwendete WTBean wird im Unterverzeichnis `wtcUsage` des Basisverzeichnisses gespeichert.

Inline WTBeans

► Um ein inline WTBean einzufügen, öffnen Sie in WebLab zuerst das Template, in das Sie das WTBean einfügen wollen.

► Positionieren Sie in diesem Template an die entsprechende Stelle und wählen Sie den Befehl **Einfügen/WTBean**. In einem Untermenü werden Ihnen die verfügbaren WTBeans angezeigt, aus denen Sie wählen können. Mit ausgeliefert werden die inline WTBeans zur Erstellung eines Kommunikationsobjekts für die verschiedenen Host-Adapter.

WebLab bietet zum Bearbeiten der Parameter aller WTBeans eine einheitliche Oberfläche, die hier am Beispiel des mitausgelieferten WTBeans `wtcOSD` dargestellt ist.



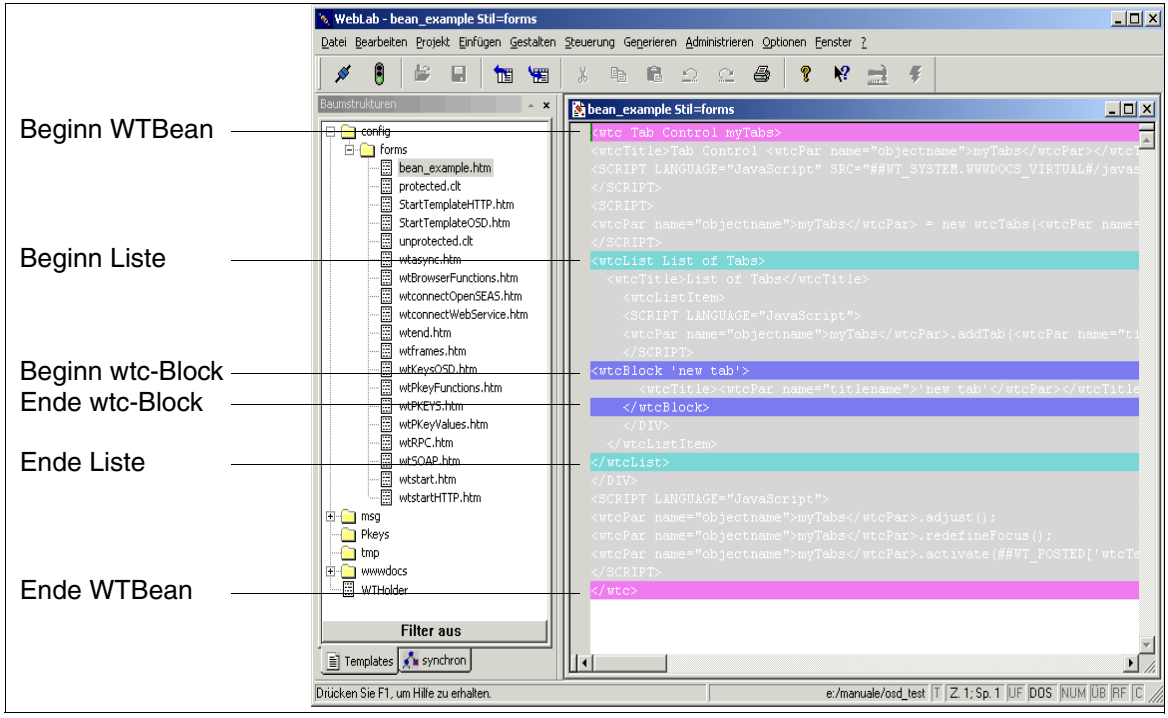
Die möglichen Parameter können Sie in einem Dialogfeld in einer oder mehreren Registerkarten bearbeiten. Die Namen der Parameter, für die Sie einen Wert angeben müssen, sind rot eingefärbt. Alle weiteren Parameter werden mit Vorbelegungen versorgt, wenn Sie nichts angeben.

- Bestätigen Sie Ihre Angaben mit **OK**. Aus Ihren Angaben und der Beschreibungsdatei wird der entsprechende WTML-Code erzeugt und im Template eingefügt. Die Beschreibungsdatei für das verwendete WTBean wird im Unterverzeichnis `wtcUsage` des Basisverzeichnisses gespeichert.

Darstellung der WTBeans im Template

WTBeans können Sie mit WebLab einfügen und bearbeiten. Wie Sie konkret vorgehen, lesen Sie in Abschnitt „WTBeans einfügen“ auf Seite 184. Ein WTBean besteht aus dem eigentlichen Code und Blöcken von freiem HTML-Text, in den Sie beliebige Texte, Scripts oder weitere WTBeans einfügen können.

Der Code eines WTBeans, unabhängig ob es sich um ein standalone oder ein inline WTBean handelt, kann nicht bearbeitet werden. Deswegen wird dieser Bereich im Template grau hinterlegt dargestellt. Start- und Endezeile eines WTBeans sind pink hinterlegt dargestellt. Beginn und Ende einer Liste sind in der Darstellung cyan hinterlegt. Die Eigenschaften eines WTBeans können Sie nachträglich ändern, siehe hierzu auch Abschnitt „WTBeans bearbeiten“ auf Seite 188.



Die Bereiche eines WTBeans, in die Sie Text und andere WTBeans eingeben dürfen, werden durch einen so genannten `wtc`-Block gekennzeichnet, dessen Beginn und Ende blau hinterlegt ist.

In der Beschreibungsdatei des WTBeans ist auch festgelegt, was in diesen Block eingefügt werden darf:

– Wenn Sie Text in den Block einfügen dürfen, können Sie nach dem Beginn des `wtc`-Blocks Leerzeilen eingeben.

– Wenn Sie weitere WTBeans einfügen dürfen, werden diese beim Befehl **Einfügen/ WTBean** im Untermenü oder im Kontextmenü des `wtc`-Blocks angezeigt.

WTBeans bearbeiten

Sie können zwar den Code eines WTBeans nicht ändern, aber Sie können nachträglich die Parameter bearbeiten.

► Klicken Sie dazu mit der rechten Maustaste auf die Startzeile des WTBeans, um das Kontextmenü zu öffnen.

- Wählen Sie den Befehl **WTBean bearbeiten**. Es wird das Dialogfeld mit den Registerkarten geöffnet, um die Parameter zu bearbeiten. Die Parameter sind mit den aktuellen Werten vorbegelegt.

Beim Bearbeiten von Templates, die WTBeans enthalten, beachten Sie folgende Veränderungen im Verhalten von WebLab:

– Sie können nur das gesamte WTBean markieren: Sobald Sie ein Zeichen innerhalb eines WTBeans markieren, ist das gesamte WTBean markiert. Ausgenommen hiervon ist nur der Text innerhalb eines `wtc`-Blocks.

– Beim Suchen und Ersetzen werden die nicht editierbaren Bereiche (grau hinterlegt) nicht berücksichtigt.

6.5.3 Templates für Hostformate gestalten

In diesem Abschnitt finden Sie vertiefende Informationen zu folgenden Themen:

– Globales Layout festlegen, das alle Templates gleichermaßen umfasst

– Hostformate gestalten

– Hostobjekte grafisch auswählen

6.5.3.1 Globales Layout festlegen

Für die Umsetzung eines globalen Layouts, das alle Templates gleichermaßen umfasst, haben Sie mehrere Möglichkeiten:

– Templates inkludieren

– Master-Templates

– Systemobjekte-Attribute `EPILOG`, `FORMTPL` und `PROLOG`

Templates inkludieren

Wollen Sie, dass sich die Gestaltungsmaßnahmen auf alle Templates auswirken, z.B. wenn Sie Firmenlogos einfügen oder generelle Informationen auf allen Ihren Seiten zur Verfügung stellen wollen, können Sie die entsprechenden Passagen in eine eigene Datei schreiben und diese mit Include-Tags in die Templates einfügen. Dadurch wird die Pflege der Templates einfacher. Wenn sich etwas ändert, brauchen Sie diese Änderung nur einmal im zentralen Include-Template vorzunehmen.

In WebLab wählen Sie zum Inkludieren einer Datei den Befehl
Einfügen/Include... oder
Einfügen/WTScript/Template Functions/include, falls Sie sich in einen Scriptbereich befinden.

Master-Templates

Sie können das globale Layout auch über die Master-Templates festlegen.

Beim Generieren von Templates geben Sie im Dialogfeld
Optionen für FLD und Template Generierung von WebLab an, welches Master-Template für die Generierung herangezogen werden soll. Einige Generierungsoptionen (z.B. die Generierungsmethode) können Sie sowohl im Master-Template als auch unmittelbar mit WebLab festlegen.

In diesem Fall werden die Einstellungen im Master-Template als Vorbelegung in das Dialogfeld übernommen. Diese Vorbelegung können Sie im Dialogfeld ändern, sodass die geänderten Werte die entsprechenden Festlegungen im Master-Template übersteuern. Das bedeutet, dass bei der Generierung eines Templates immer die Werte verwendet werden, die im Dialogfeld angezeigt werden.

Eine detaillierte Beschreibung der Master-Templates finden Sie im entsprechenden Kapitel im WebTransactions-Handbuch „Template-Sprache“.

Systemobjekte-Attribute `EPILOG`, `FORMTPL` und `PROLOG`

Auch mit den Attributen `EPILOG`, `FORMTPL` und `PROLOG` am verbindungsspezifischen Systemobjekt können Sie global die Gestaltung der generierten Templates beeinflussen.

Die Attribute enthalten jeweils den Namen eines Templates, das entsprechend dem Attribut zu unterschiedlichen Zeitpunkten ausgeführt wird:

`PROLOG`	zu Beginn des aktuellen Templates
`FORMTPL`	vor Ausführen des DataForm-Tags im aktuellen Template
`EPILOG`	am Ende des aktuellen Templates

Die mitgelieferten Master-Templates sorgen dafür, dass die in den Attributen angegebenen Templates in den generierten Templates inkludiert werden.

6.5.3.2 Hostformate gestalten

Standard-Verschönerungsschritte können Sie durch die Assistenten von WebLab automatisch ausführen lassen. Diese Assistenten ersetzen generierte Eingabefelder (INPUT-Tags vom Typ „Text“) durch grafische Elemente, wie Drop-Down-Listen, Radio-Knöpfe, Checkboxen oder Ausführungsknöpfe. (Eine Beispielsitzung hierzu finden Sie in den Handbüchern der jeweiligen Liefereinheit.)

Sie finden die Assistenten entweder im Menü **Gestalten** oder im Kontextmenü der grafischen Hostobjektauswahl, siehe folgender Abschnitt.

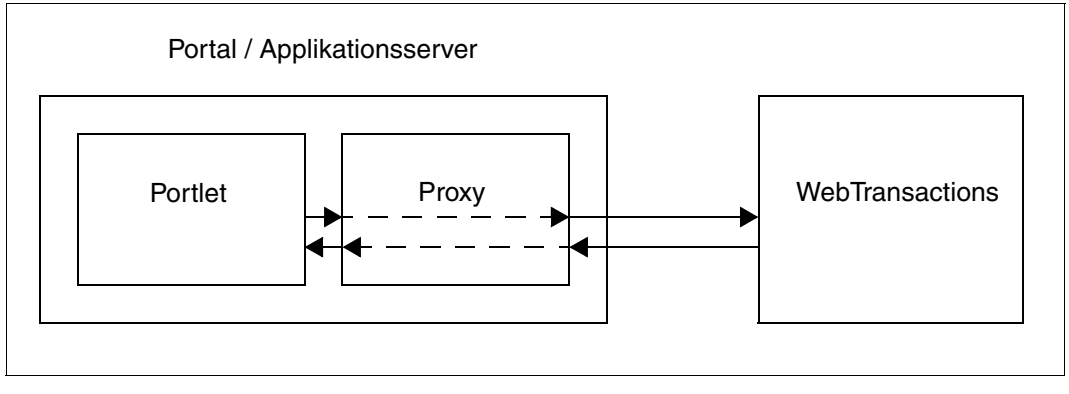
6.5.3.3 Hostobjekte grafisch auswählen

Da bei Host-Datenobjekten die Namen nicht immer einen Rückschluss auf den Zweck des Objekts zulassen, bietet Ihnen WebLab für diese Objekte eine zusätzliche grafische Auswahlmöglichkeit (Befehl **Gestalten/Hostobjekte grafisch auswählen**): In einer Darstellung des Ausgangsformats wählen Sie ein Host-Datenobjekt aus, indem Sie auf das entsprechende Oberflächenelement klicken. Per Doppelklick wird der entsprechende Name an der aktuellen Cursorposition im Editierbereich eingesetzt.

Sie können zusätzlich per Drag&Drop voll qualifizierte Variablennamen in den Arbeitsbereich übernehmen. Wenn Sie während des Drag&Drop die `Shift`-Taste drücken, wird nur der Attributname verwendet. Wenn Sie die `Alt`-Taste gedrückt halten, wird der voll qualifizierte Name jeweils innerhalb eines Auswertungsoperators eingefügt. Halten Sie die `Strg`-Taste gedrückt, wird der aktuelle Wert der Variablen eingefügt.

6.5.4 Templates für Portaleinsatz gestalten

Die Portlets bzw. iViews, die die Einbindung von Host-Dialoganwendungen in Portale unterstützen, verwenden intern einen Proxy, um mit der WebTransactions-Anwendung zu kommunizieren:



Dieser Proxy nimmt portalspezifische Modifikationen an den generierten Seiten vor, bevor sie im Browser angezeigt werden. Damit wird sichergestellt, dass Referenzen (z.B. auf Bilder) nicht am Portal vorbei aufgelöst werden. Wegen dieser Konstruktion und weil die Portlets nur auf einem Teil einer Seite im Portal ablaufen, ergeben sich folgende Einschränkungen für die Template-Gestaltung:

Einschränkung	gültig für	
	Oracle-/ JSR-168 Portlets	**SAP EP iViews**
Verwenden Sie nicht das gesamte Browserfenster für die Ausgabe: – Verwenden Sie in Scripts keine Konstrukte wie `top.location = …` – Setzen Sie in Links nicht `target="_top"` .	X	X
Der Proxy reicht nur den http-Header "User-Agent" weiter.	X	
Ersetzungen in den Antworten werden nur vorgenommen, wenn der Content-Type text/html oder text/javascript ist.	X	
Im Dokument enthaltene Adressen werden so angepasst, dass sie den Proxy ansprechen. Adressen werden erkannt im href-Attribut von a- und link-Tags, im src-Attribut der Tags frame, iframe, img und script, sowie im action-Attribut des form-Tags. Adressen in Javascript werden nur als solche erkannt (und modifiziert), wenn sie in Zuweisungen (nach einem =) stehen, und wenn die Variable, der sie zugewiesen werden, mit `document.location`, `document.location.href` oder `.src` endet.	X	
In jedem Template muss genau ein form-Tag enthalten sein; dieses muss die WebTransactions-Anwendung adressieren.		X

6.5.5 Dialogablauf gestalten

WebTransactions bietet Ihnen aber nicht nur die Möglichkeit zu einem „Face Lifting“ Ihrer Host-Anwendungen. Sie können auch die Dialogabläufe neu gestalten. Die starre 1:1 Zuordnung zwischen HTML-Seite und Host-Format wird aufgebrochen: Mit WebTransactions können Sie die von den Host-Anwendungen vorgesehenen Dialogstrategien aktiv steuernd verändern: Ein-/Ausgabe-Elemente können ausgefiltert oder hinzugefügt werden, Dialogschritte lassen sich zusammenfassen oder auffächern.

Dialoge in WTScript aufzeichnen

WebLab ermöglicht das Aufzeichnen von Dialogschritten mit Ihrer Host-Anwendung als WTScript. So können Sie beispielsweise komfortabel in einem Template mehrere Dialogschritte mit der Host-Anwendung zusammenfassen. Sie können die ganze Folge von Dialogschritten aufzeichnen, in das Template einfügen und ablaufen lassen.

Zum Aufzeichnen verwenden Sie die Befehle des Menüs **Steuerung**/**Dialog aufzeichnen**.

6.5.6 Templates formatieren

Sie haben folgende Möglichkeiten, ein geöffnetes Template zu formatieren (Befehle **Bearbeiten/Template formatieren** und **Optionen/Einstellungen/Format**):

- Schreibweise für HTML- und WTML-Tags ändern
- Script-Quelltext formatieren
- HTML-Tags einrücken
- WTML-Tags einrücken

Beim Einrücken bleiben Zeilenumbrüche des Originals unverändert.Sie können jedoch einstellen, dass geschweifte Klammern und Block-Tags nach der Formatierung auf einer eigenen Zeile stehen sollen.

6.5.6.1 Schreibweise für HTML- und WTML-Tags ändern

Die Schreibweise aller HTML- und WTML-Tags kann für ein Template global geändert werden.

Es gibt folgende zwei Varianten:

- Alle Tags werden groß geschrieben, z.B. WTONCREATESCRIPT.
- Alle Tags werden klein geschrieben, z.B. wtoncreatescript.

6.5.6.2 Script-Quelltext formatieren

Die Formatierung der Script-Bereiche unterliegt folgenden Regeln (Option **Zeilen ausrichten**):

- Bei jedem Blockbeginn wird die Einrückungstiefe inkrementiert, nach Blockende wird sie wieder dekrementiert.
- Fortgesetzte Anweisungen werden eine Stufe tiefer eingerückt als der Beginn der Anweisung.
- Kommentare werden wie der umgebende Text eingerückt.
- Leerzeilen werden auf das Zeilenumbruchzeichen reduziert.

Beispiel

```
if ( wtCurrentComm_system.EDIT_MODE )
{
if ( typeof wtCurrentComm_system.isOverwrite == 'undefined'
&& wtCurrentComm_system.EDIT_MODE.match(/OVERWRITE/) )
wtCurrentComm_system.isOverwrite = true;
} else
wtCurrentComm_system.isOverwrite = false;
```

wird zu:

```
if ( wtCurrentComm_system.EDIT_MODE )
{
   if ( typeof wtCurrentComm_system.isOverwrite == 'undefined'
      && wtCurrentComm_system.EDIT_MODE.match(/OVERWRITE/) )
      wtCurrentComm_system.isOverwrite = true;
} else
   wtCurrentComm_system.isOverwrite = false;
```

6.5.6.3 HTML-Tags einrücken

Bei einem öffnenden Block-Tag wird die Einrücktiefe für die Folgezeilen inkrementiert, beim Erkennen des Ende-Tags dekrementiert. Dabei wird das html-Tag nicht berücksichtigt.

Bei einem Tag mit optionalem Ende-Tag, wird die Einrücktiefe dekrementiert, sobald ein Tag erkannt wird, das dieses Tag implizit schließt.

Wenn ein Tag länger als eine Zeile ist, werden alle Zeilen bis zum schließenden > eine Stufe weiter eingerückt.

Inline-Tags und Tags, die kein Ende-Tag haben dürfen, werden wie Text behandelt.

Text innerhalb von pre-Tags wird nicht verändert.

Eine Einteilung der HTML-Tags in Klassen, die bei der Berechnung der Einrücktiefe eine Rolle spielen, finden Sie in der Online-Hilfe zu WebLab.

Nicht-paarige Tags

Durch die Mischung verschiedener Programmiertechniken, kann es vorkommen, dass HTML-Tags nicht-paarig auftreten, die laut Definition paarig sind.
Mögliche Ursachen dafür sind u.a.:

- Ein Tag wird in einem Script geöffnet und im HTML-Text wieder geschlossen oder umgekehrt.
- Quelltext wurde in Include-Dateien ausgelagert.

In diesem Fall kann es dazu kommen das zusammengehörende Tags nicht auf der gleichen Spalte beginnen.

Beispiel

```
<html>
  <body>
    <script>
      document.write("<table border=1>");
    </script>
  </table>
</body>
</html>
```

Style-Tags

Innerhalb eines Style-Tags sind die Stil-Formatierungen in geschweifte Klammern eingeschlossen. Deshalb wird innerhalb der geschweiften Klammern eingerückt.

Beispiel

```
<style>
    .h1
    {
        font-family:'Times New Roman', Times, serif;
        font-style:italic;
        font-weight:bold;
    }
</style>
```

6.5.6.4 WTML-Tags einrücken

WTML-Tags werden soweit als möglich wie HTML-Tags behandelt. Beachten Sie lediglich für die Tags für die Kontrollstrukturen Folgendes:

Diese Tags verändern die Einrückungstiefe für HTML-Tags nicht, lediglich für weitere WTML-Tags für Kontrollstrukturen wird die Einrückungstiefe erhöht. So kann garantiert werden, dass öffnendes und schließendes Tag in der gleichen Spalte beginnen, auch wenn HTML- und WTML-Tags verschränkt verwendet werden, z.B. ein HTML-Tag wird innerhalb eines If-Tags geöffnet und außerhalb geschlossen.

Beispiel

```
<html>
<body>
<wtIf (a>b)>
    <table>
<wtElse>
    <table bgcolor="#ff00ff">
</wtIf>
        sometext
        <tr>
            <td>
            </td>
        </tr>
    </table>
</body>
</html>
```

Wenn innerhalb des If- und des Else-Zweiges Tags geöffnet werden, werden für die Berechnung der Einrücktiefe nach dem schließenden If-Tag nur die Tags des If-Zweiges berücksichtigt.

6.5.7 Templates dokumentieren

WebTransactions-Anwendungen können sehr viele Templates mit komplexer Funktionalität enthalten und müssen daher ausreichend dokumentiert werden. WebLab bietet Ihnen Unterstützung beim Einbringen von Kommentaren in die Templates und die Möglichkeit, aus den Kommentaren eine Dokumentation in Form von HTML-Seiten zu erstellen. WebLab berücksichtigt alle Dateien im Basisverzeichnis mit einem Suffix `.htm`, `.html`, `.js`, `.clt`, und `.service`.

Bei der Generierung der Dokumentation werden nur die folgenden Kommentare berücksichtigt:

– Dateikommentare, die am Anfang der Datei stehen (siehe Abschnitt „Dateikommentare“ auf Seite 199).

– Funktionskommentare, die direkt in der Zeile vor der Funktionsdefinition enden (siehe Abschnitt „Funktionskommentare“ auf Seite 199).

Um Kommentare in Templates einzubringen oder HTML-Seiten zu generieren, muss eine Verbindung zum WebTransactions-Server bestehen.

6.5.7.1 Format der Kommentare

Alle Kommentare müssen entsprechend den folgenden Regeln erstellt werden. Um Sie bei der Erstellung von Kommentaren zu unterstützen, bietet WebLab Ihnen syntaktisch korrekte Gerüste an, die Sie dann nur noch vervollständigen müssen (siehe Abschnitt „Kommentare einfügen“ auf Seite 199).

– Jeder Kommentar beginnt mit der Zeichenkette `/**` .

– Jede Folgezeile in einem Kommentar beginnt mit einem Stern (`*`).

– Der erste Satz endet mit dem ersten Punkt, der im Kommentar auftritt.

– Der erste Satz des Kommentars sollte kurz die Funktion beschreiben. Dieser Satz wird als Kurzbeschreibung in der Funktionsübersicht verwendet (siehe Abschnitt „Beispiel“ auf Seite 203).

– Die Beschreibung der Funktion endet mit einer leeren Kommentarzeile.

– Die Zeilen, die Symbole enthalten, beginnen nach dieser leeren Kommentarzeile (zu den möglichen Symbolen siehe Seite 198).

– Zeilen, die ein Symbol enthalten, müssen mit dem Symbol beginnen.

– Jeder Kommentar endet mit der Zeichenkette `*/` .

Zur Formatierung von Kommentaren können alle HTML-Tags verwendet werden. Die Texte in den Kommentaren werden unverändert in die HTML-Seiten übernommen.

Folgende Punkte sind zu beachten:

– Zeichen, die in HTML eine Sonderbedeutung haben, müssen codiert werden (z.B < oder > als `<` bzw. `>`).

– Zeilenumbrüche in der Dokumentation müssen mit HTML-Mitteln erzeugt werden (z.B. mit dem Tag `<br>`).

– In Kommentaren können die folgenden Symbole verwendet werden (Beispiel siehe Seite 203). Diese Symbole müssen am Zeilenanfang stehen.

`@author`
: gibt den Autor der Datei an. Dieses Symbol wird nur in Dateikommentaren verwendet.

`@param`
: beschreibt einen Parameter einer Funktion. Folgendes Format ist dabei einzuhalten:

  `@param` *parameter description*

`@return`
: beschreibt den Rückgabewert der Funktion.

`@throws`
: beschreibt die Exception, die die Funktion wirft. Folgendes Format ist dabei einzuhalten:

  `@throws` *name_of_exception description*

`@method`
: ordnet in einem Konstruktor-Kommentar eine Funktion als Methode einer Klasse zu. Dabei muss folgendes Format eingehalten werden:

  `@method` *method-name function-name*

Dateikommentare

Diese Kommentare beschreiben den Inhalt einer Datei. Sie müssen deshalb am Anfang der Datei stehen.

Dateikommentare, die außerhalb von Script-Bereichen stehen, müssen zusätzlich in Rem-Tags eingeschlossen werden (Ausnahme: Dateikommentare in JavaScript-Dateien).

Beispiel

```
<wtrem>
/**
 *
 * @author
*/
</wtrem>
<wtrem>
************************************************************************
Funktionen zur Fehlerbehandlung
************************************************************************
</wtrem>
```

Funktionskommentare

Diese Kommentare sind genau einer Funktion zugeordnet. Sie müssen deshalb direkt in der Zeile über der Funktion enden.

Beispiele für Funktionskommentare finden Sie in den Abschnitten „Funktionskommentar“ auf Seite 200 und „Beispiel“ auf Seite 203).

6.5.7.2 Kommentare einfügen

WebLab bietet Ihnen verschiedene Funktionen, die Sie beim Einbringen von Kommentaren in Templates unterstützen. Sie gehen folgendermaßen vor:

- Öffnen Sie das Template, das Sie bearbeiten wollen.

Dateikommentar

- Um einen Dateikommentar einzufügen, wählen Sie den Befehl **Dateikommentar einfügen** im Kontextmenü oder im Menü **Einfügen**.

  WebLab fügt eine Vorlage für den Kommentar am Dateibeginn ein.

Funktionskommentar

- Um einen Funktionskommentar einzufügen, positionieren Sie den Cursor in die Zeile, in der die Funktion definiert ist.
- Wählen Sie den Befehl **Funktionskommentar einfügen** im Kontextmenü oder im Menü **Einfügen**.
  Dieser Befehl ist nur verfügbar, wenn der Cursor in der Zeile steht, in der die Funktion definiert ist.

  WebLab fügt eine Vorlage für den Funktionskommentar direkt über der Funktion ein. WebLab analysiert dabei den Funktionsaufruf und fügt für alle Parameter die passenden Zeilen mit dem Symbol `@param` ein.

Beispiele

WebLab fügt abhängig vom Funktionsaufruf die folgenden Kommentare ein:

Funktionsaufruf

```
function test (a,b,c)
```

Kommentar

```
/**
 *
 *
 * @param a
 * @param b
 * @param c
 * @return
 * @throws
*/
```

Funktionsaufruf

```
function WT_SOAP( /*string*/ wsdlSrc, /*string*/ proxyHost, /*number*/
proxyPort )
```

Kommentar

```
/**
 *
 *
 * @param wsdlSrc [string]
 * @param proxyHost [string]
 * @param proxyPort [number]
 * @return
 * @throws
*/
```

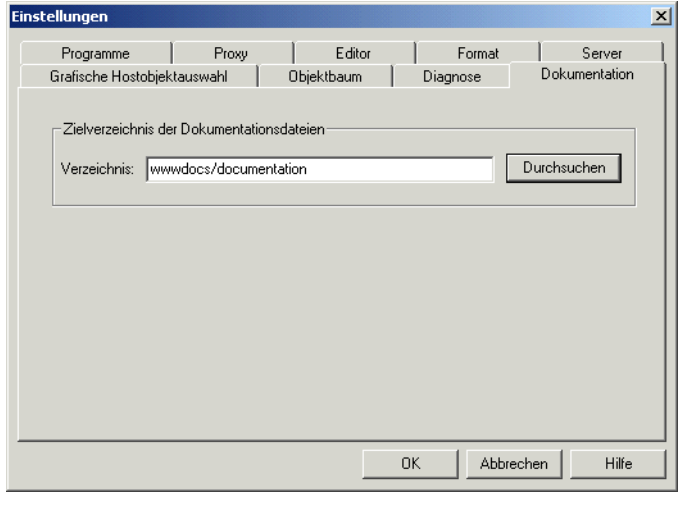
6.5.7.3 Dokumentation generieren

Um mit Hilfe von WebLab HTML-Seiten aus Ihren Kommentaren zu generieren, gehen Sie wie folgt vor:

Als ersten Schritt legen Sie fest, wo die HTML-Seiten abgelegt werden sollen.

- ► Wählen Sie den Befehl **Optionen/Einstellungen**.
- ► Wählen Sie im Dialogfeld **Einstellungen** die Registerkarte **Dokumentation**.
- ► Geben Sie im Feld **Verzeichnis** an, wo die HTML-Seiten abgelegt werden sollen. Die Voreinstellung für diesen Wert ist *wwwdocs/documentation*.

  Das Zielverzeichnis der Dokumentation muss sowohl von WebLab, als auch vom Web-Server aus erreichbar sein. Das Verzeichnis muss also:
  - unter dem Verzeichnis `wwwdocs` auf dem WebTransactions-Server liegen oder
  - auf dem lokalen Rechner unter dem Root-Verzeichnis für Web-Seiten (=Dokumentenverzeichnis) liegen.



Anschließend generieren Sie die HTML-Seiten:

► Wählen Sie den Befehl **Generieren**/**Dokumentation**.

WebLab erzeugt HTML-Seiten aus allen Dateien im Basisverzeichnis mit einem Suffix `.htm`, `.html`, `.js`, `.clt` und `.service`. Verweise auf das Installationsverzeichnis werden nicht berücksichtigt.

Die HTML-Seiten werden in dem Verzeichnis abgelegt, das Sie im Dialogfeld **Einstellungen** angegeben haben. Sie können mit einem beliebigen Browser geöffnet werden.

Sie können die Dokumentation von WebLab aus aufrufen:

► Wählen Sie den Befehl **?**/**Dokumentation anzeigen**.

WebLab öffnet dann ein Browser-Fenster mit der Startseite der Dokumentation

6.5.7.4 Format der Darstellung

Sie können die Dokumentation einer WebTransactions-Anwendung mit dem Befehl **?**/**Dokumentation anzeigen** aufrufen.

Für die HTML-Seiten gibt es die folgenden zwei Darstellungs-Möglichkeiten. Sie wählen die Darstellung über Knopf **Files** bzw. **Index** am Kopf der jeweiligen Seite.

Datei-Darstellung

Wenn Sie den Knopf **Files** auswählen, erhalten Sie eine Übersicht über die Funktionen, wie sie in der Datei aufeinanderfolgen. Die HTML-Seite enthält in dieser Reihenfolge:

– den Dateikommentar.

– eine Übersicht über die Funktionen in der Datei, aufgeteilt nach Server-seitig und Client-seitig definierten Funktionen. Die Übersicht enthält für jede Funktion die Definition und den ersten Satz aus dem Funktions-Kommentar. Die Detail-Ansicht der Funktion ist über einen Link erreichbar.

– Detail-Ansichten für alle Funktionen. Diese enthalten den kompletten Funktionskommentar.

Alphabetische Darstellung

Wenn Sie den Knopf **Index** auswählen, erhalten Sie eine alphabetische Liste aller Funktionen in der Datei. Die Liste enthält für jede Funktion die Definition und den ersten Satz aus dem Funktions-Kommentar. Die Detail-Ansicht der Funktion ist über einen Link erreichbar.

6.5.7.5 Beispiel

Die Datei `test.htm` enthält die Funktion `WT_SOAP` zusammen mit den folgenden Kommentaren:

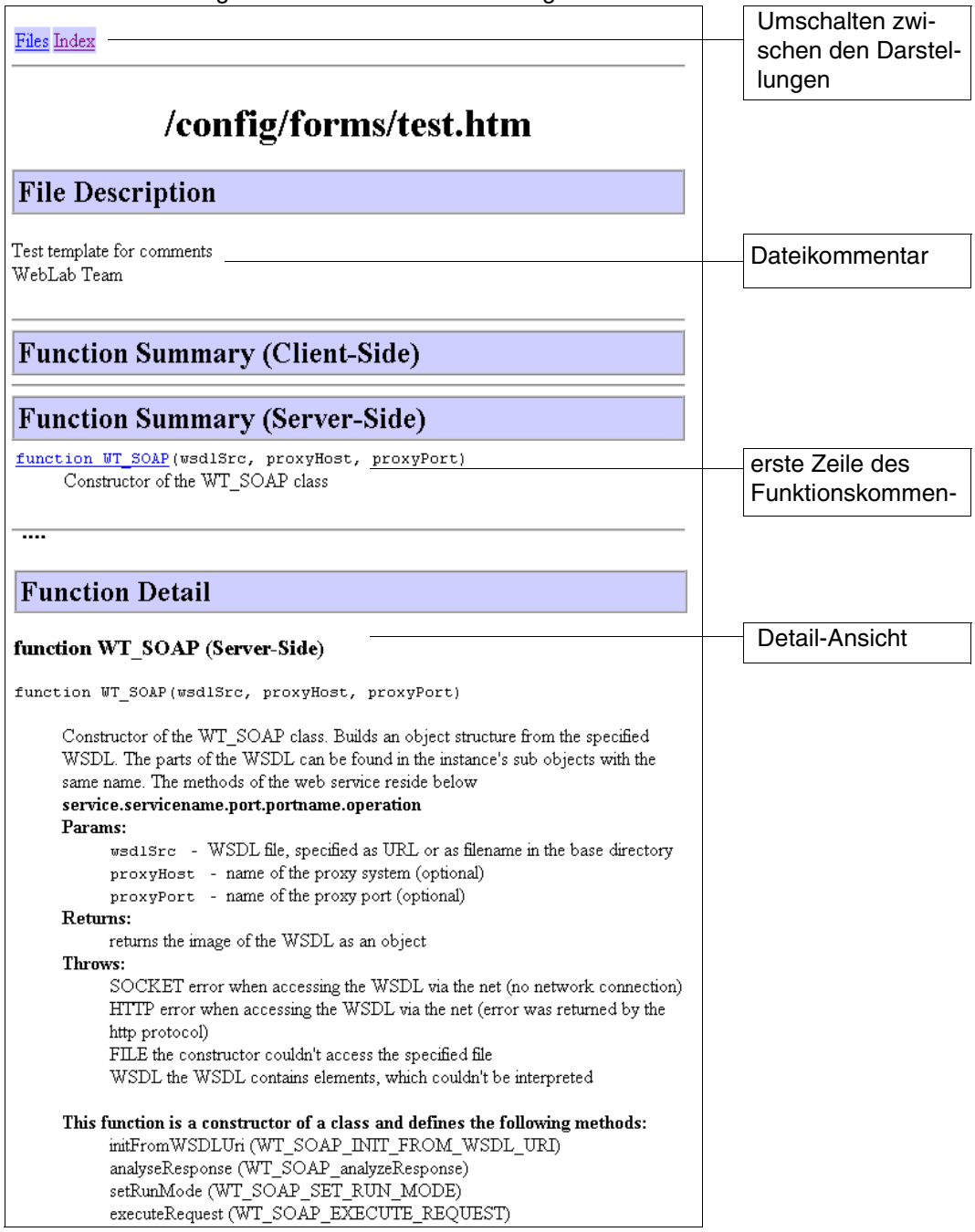
```
<wtrem>
/**
 *
 * Test template for comments <br>
 * WebLab Team
*/
</wtrem>
<wtoncreatescript>

/**
*  Constructor of the WT_SOAP class.
 * Builds an object structure from the specified WSDL.
 * The parts of the WSDL can be found in the instance's
 * sub objects with the same name.
 * The methods of the web service reside below
 *  <b>service.servicename.port.portname.operation</b>
 *
 * @param wsdlSrc WSDL file, specified as URL or as
 * filename in the base directory
 * @param proxyHost name of the proxy system (optional)
 * @param proxyPort name of the proxy port (optional)
 * @return returns the image of the WSDL as an object
 * @throws SOCKET:   error when accessing the WSDL via the net
                   (no network
connection)                                          (1)
 * @throws HTTP:     error when accessing the WSDL via the net
                   (error was returned by the http
protocol)                    (1)
 * @throws FILE:     the constructor couldn't access the specified
file             (1)
 * @throws WSDL:     the WSDL contains elements, which couldn't
                    be
interpreted                                                  (1)
 * @method initFromWSDLUri
WT_SOAP_INIT_FROM_WSDL_URI                                 (2)
 * @method analyseResponse
WT_SOAP_analyzeResponse                                    (2)
 * @method setRunMode
WT_SOAP_SET_RUN_MODE                                         (2)
 * @method executeRequest
WT_SOAP_EXECUTE_REQUEST                                    (2)
*/
function WT_SOAP(  wsdlSrc, proxyHost, proxyPort )
```

```
....
....
```

(1) Die Klasse `WT_SOAP` wirft mehrere Exceptions, die sich vom Namen her unterscheiden.

(2) In der Klasse `WT_SOAP` werden diese Funktionen als Methoden bereitgestellt.

Sie erhalten in der generierten Dokumentation folgende HTML-Seite:



6.6 Templates testen

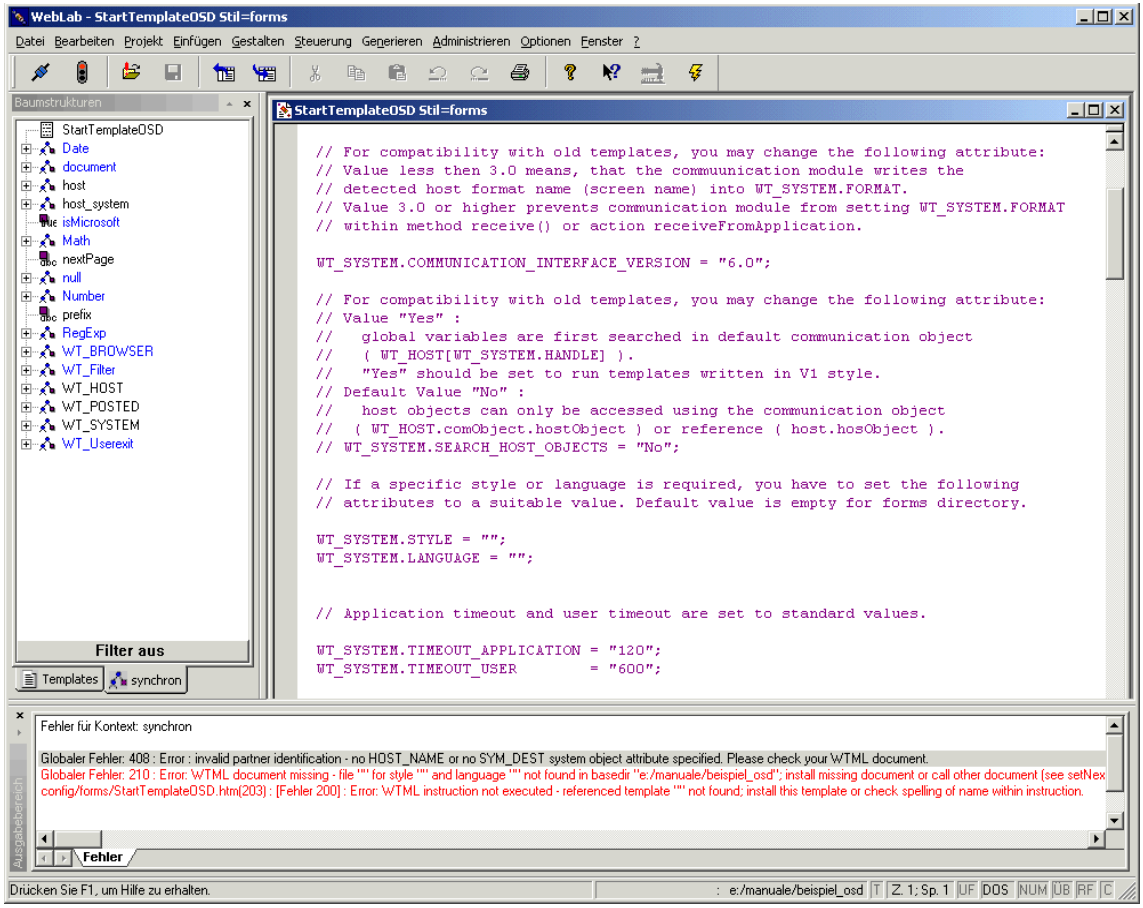
Für den Test Ihrer WebTransactions-Anwendung stehen Ihnen verschiedene Möglichkeiten zur Verfügung:

– Gestaltung eines Templates testen
– Ablauf im Template testen (Einzelschrittverfolgung)

Im Folgenden werden die Testfunktionen der Entwicklungsumgebung WebLab kurz vorgestellt. Eine detaillierte Beschreibung ist in der Online-Hilfe von WebLab enthalten.

6.6.1 Gestaltung eines Templates testen

WebLab bietet Ihnen die folgenden komfortablen Möglichkeiten, die Gestaltung einer HTML-Seite zu testen.



Fehler korrigieren

Falls Fehlermeldungen von WebTransactions ausgegeben werden, werden Ihnen diese im Ausgabebereich von WebLab angezeigt. Ein Doppelklick auf die Fehlermeldung positioniert im Editier-Bereich den Cursor automatisch auf die fehlerhafte Stelle. Diese kann dann sofort korrigiert werden.

Über den Befehl **Steuerung/Im Browser aktualisieren** wird die geänderte Seite unmittelbar neu erzeugt und im Browser dargestellt.

Werte neu setzen

In der Testphase ist es oft hilfreich, die Werte einzelner Attribute neu zu setzen und die Auswirkung zu überprüfen. Hierfür können Sie das gewünschte Objekt im Objektbaum auswählen.

Über den Befehl **Eigenschaften** im Kontextmenü wird der aktuelle Wert angezeigt und kann geändert werden. Wie bei Fehlerkorrekturen können Sie das Ergebnis unmittelbar überprüfen: Mit dem Befehl **Im Browser aktualisieren** wird die Seite mit den geänderten Werten erzeugt und angezeigt.

Hinweise zur Funktion „Im Browser aktualisieren“

Mit dem Befehl **Steuerung/Im Browser aktualisieren** können Sie die aktuelle Seite nach einer Korrektur im Template erneut ausgeben und Ihre Änderung überprüfen.

Bei dieser Funktion sind einige Einschränkungen zu beachten, die das Ergebnis der Neugenerierung gegenüber der normalen Ausgabe beeinflussen können:

– System- und Host-Objekte werden nicht auf den ursprünglichen Zustand zurückgesetzt: Werden Attribute dieser Objekte in OnCreateScript-Tags verändert, so liegen der Neugenerierung die geänderten Werte zu Grunde.

– Kommunikationsschritte mit der Host-Anwendung werden nicht zurückgesetzt: Werden die Methoden `open`, `close`, `send` und `receive` verwendet, so lässt sich deren Wirkung nicht rückgängig machen. Die Kommunikationsschritte werden bei der Neugenerierung unterdrückt, um die Host-Anwendung im aktuellen Zustand zu belassen. Änderungen solcher OnCreate-Scripts (hinzufügen, löschen) haben also keine direkte Wirkung. Die Neugenerierung kann also einen inkonsistenten Zustand der Host-Anwendung bewirken.

  OnReceive-Scripts sind hingegen vollkommen unkritisch, da deren Ausführung bei der Neugenerierung unterdrückt wird. D.h. die für den Receive-Zeitpunkt gespeicherten alten Anweisungen werden verworfen und die neuen OnReceive-Scripts werden gespeichert (siehe auch Abschnitt „Dialogzyklus“ auf Seite 67).

  In seltenen Fällen kann es Probleme mit der Neugenerierung geben. Hier müssen Sie erneut an den Testpunkt navigieren, um das betreffende Template zu testen. Im Normalfall, z.B. bei allen generierten Templates, ist die Neugenerierung problemlos und beschleunigt den Test der Oberfläche erheblich.

6.6.2 Ablauf im Template testen

Sie können den Ablauf eines Templates nachvollziehen, indem Sie zuerst alle Schritte protokollieren, die in einem Template während der Generierung der HTML-Seiten und der Abarbeitung der Receive-Scripts ablaufen. WebTransactions schreibt dabei eine Protokolldatei nach folgenden Regeln.

– Nach jeder ausgeführten Zeile werden die Variablen mit ihren Werten protokolliert.

– Für Hostobjekte wird ihr Klassen-Template mitausgeführt und das Ergebnis protokolliert.

– Die Generierung eines HTML-Bereichs ist ein Einzelschritt, stellvertretend wird die letzte Zeile des HTML-Bereichs angezeigt.

– Wenn innerhalb eines Dialogschrittes mehrere WebTransactions-Schritte ausgeführt werden (z.B. bei Framesets), werden nur die Einzelschritte der letzten Generierung angezeigt.

Diese Protokolldatei verwendet WebLab danach zur Einzelschrittverfolgung.

Bitte beachten Sie, dass WebTransactions in der Regel die Protokoll-Datei bei Beendigung der Sitzung löscht. Wollen Sie Einzelschritte protokollieren, die das Sitzungsende beinhalten, so müssen Sie zusätzlich den WebTransactions-Trace einschalten (siehe Abschnitt „Trace-Funktionen“ auf Seite 130).

Die Einzelschrittverfolgung können Sie nutzen, um

– den Ablauf über ein Template zu verfolgen

– den Ablauf eines bestimmten Bereichs in einem Template zu verfolgen

Die aktuellen Variablen und ihre Werte sehen Sie in der Registerkarte **Einzelschrittverfolgung** im Ausgabebereich.

6.6.2.1 Ablauf über ein Template oder über einen Bereich im Template verfolgen

► Als Einstieg zum Verfolgen des Ablaufs gibt es folgende zwei Möglichkeiten:

  ► Um die Einzelschritte eines kompletten Templates zu verfolgen, schalten Sie die Protokollierung ein mit dem Befehl **Steuerung/Einzelschrittverfolgung/Einzelschritte protokollieren**.

  oder:

  ► Um den Ablauf nur über einen bestimmten Bereich im Template zu testen, aktivieren Sie die Protokollierung für den Bereich direkt im Template mit der WTScript-Funktion setSingleStep(“on“|“off“), siehe hierzu auch das WebTransactions-Handbuch „Template-Sprache“ und das nachfolgende Beispiel.

  Die weiteren Schritte sind für beide Möglichkeiten gleich.

- Lassen Sie die Sitzung mit dem Template ablaufen, das Sie testen wollen.
- Starten Sie die Einzelschrittverfolgung mit dem Befehl **Steuerung/Einzelschrittverfolgung/Einzelschrittverfolgung beginnen**.
- Nutzen Sie die Befehle im Untermenü **Steuerung/Einzelschrittverfolgung** oder die Symbolleiste **Einzelschrittverfolgung**, um den Programmablauf nachzuvollziehen. WebLab markiert die gerade aktuelle Anweisung im Template und zeigt die von dieser Anweisung verwendeten Variablen und ihre Werte im Ausgabebereich an.

Eine detaillierte Beschreibung der Befehle des Untermenüs **Einzelschrittverfolgung** finden Sie in der ausführlichen Online-Hilfe zu WebLab.

Beispiel

In diesem Beispiel wird der Ablauf lediglich über einen Bereich im Template verfolgt:

```
...
<wtOnCreateScript>
<!--
  wtInputFields = new Object;
  currentLine   = 1;
  for (element = OSD_O.$FIRST.Name; OSD_O && element != '$END'; element =
OSD_O.$NEXT.Name)
  {
    currentHostObject = OSD_O[element];
    if ( currentHostObject.StartLine != currentLine )
    {
      document.writeln();
      currentLine++;
    }
   if ( currentHostObject.Type == 'Protected' && currentHostObject.Markable
   == 'No' )
       {
         setSingleStep ("on");
         taggedOutput( currentHostObject );
         setSingleStep ("off");
       }
       else
       {
         wtInputFields[ element ] = currentHostObject;
         taggedInput( currentHostObject );
       }
     }
   //-->
   </wtOnCreateScript>
   ...
```

6.6.2.2 Variablenwerte überwachen

Mit Hilfe des Wertefensters können Sie Variablenwerte überwachen, siehe auch Abschnitt „Das Wertefenster“ auf Seite 181.

Wie Sie im Detail mit dem Wertefenster arbeiten, entnehmen Sie der ausführlichen Online-Hilfe zu WebLab.

6.7 Werkzeuge eines Servers in WebLab einbinden

Als WebTransactions-Administrator (Benutzerkennung `admin`) können Sie Kommandos, die auf einem WebTransactions-Server ablaufen, in WebLab für Benutzer zur Verfügung stellen. Kommandos und Programme sind mit dem Begriff Werkzeug zusammengefasst.

Beachten Sie, dass die Kommandos batch-fähig sein müssen. Das heißt, die Kommandos dürfen keine Eingaben vom Anwender erwarten.

Werkzeuge definieren Sie mit dem Befehl
Optionen/Werkzeuge auf dem Server editieren.

Bei der Formulierung eines Kommandos können Sie bestimmte Schlüsselworte verwenden. Diese Schlüsselworte werden bei Ausführung eines Kommandos von WebLab mit den Daten der aktuellen Sitzung versorgt.

Schlüsselwort	**Ersetzung**
`%BASEDIR`	aktuelles Basisverzeichnis
`%CURRENT`	Name der Datei im aktiven Editierfenster inklusive Pfad z.B. basedirs\test\config/forms/trav0.htm
`%CURRENTNAME`	Name der aktiven Datei z.B. trav0.htm
`%CURRENTPATH`	Pfad der aktiven Datei (Windows: ohne Laufwerksangabe z.B. \basedirs\test\config\forms)
%DRIVE	Laufwerk der aktiven Datei (nur Windows)
`%SESSIONID`	ID der laufenden Sitzung
%SELTEXT	im aktuellen Fenster ausgewählter Text
`%USER`	aktiver WebTransactions-Benutzer
`%1,%2,...,%9`	Parameter, die jeder Anwender selbst mit **Optionen//Werkzeuge auf dem Server anpassen** angeben kann.

Zusätzlich können Sie festlegen, welche Benutzer die Werkzeuge benutzen dürfen. Wenn sich diese Benutzer mit einem Basisverzeichnis auf diesem Server verbunden haben, sehen sie die für sie freigegebenen Werkzeuge im Untermenü
Steuerung/Werkzeuge auf dem Server.

Mit einem Klick auf ein Werkzeug können Sie es am Server ausführen. Solange ein Werkzeug ausgeführt wird, sind alle anderen Werkzeuge gesperrt. Schließen Sie die Verbindung, wird die Ausführung des Werkzeugs abgebrochen.

Die Ausgaben des Werkzeugs werden in zwei Registerkarten des Ausgabebereichs angezeigt:

Werkzeuge auf dem Server (stdout)
enthält die Ausgaben, die vom Werkzeug auf der Console oder am Bildschirm ausgegeben werden.

Enthält eine Zeile der Ausgabe einen Dateinamen, können Sie durch einen Doppelklick auf diese Zeile die angegebene Datei öffnen. Enthält die Zeile zusätzlich eine Zeilennummer, wird zusätzlich der Cursor in die angegebene Zeile positioniert. Die Zeilen, für die dieses Vorgehen möglich ist, haben folgendes Format:

filename:*linenumber*:*text*
filename(*linenumber*) *text*
filename:*text*

Werkzeuge auf dem Server (stderr)
enthält mögliche Fehlermeldungen des Werkzeugs.

6.8 WebTransactions-Anwendung transferieren und verteilen

Eine WebTransactions-Anwendung transferieren Sie, indem Sie sie auf dem Entwicklungsrechner verpacken und auf dem Zielrechner wieder entpacken. Beim Verteilen werden Dateien oder Verzeichnisse jeweils in WebLab eingelesen und dann auf die Zielrechner eines Clusters übertragen.

Das genaue Vorgehen ist im Folgenden beschrieben.

6.8.1 Übertragungsumfang

Vor der Verpackung in ein Archiv können Sie auswählen, welche Verzeichnisse und Dateien übertragen werden sollen. Eine Übersicht, wie die Dateien der verschiedenen Unterverzeichnisse des Basisverzeichnisses übertragen werden, gibt folgende Tabelle.

Verzeichnis	**Inhalt**	**Übertragungsmodus**
`basedir`	Programmdateien, Bibliotheken	Binär
`config`	FLD-Dateien	Text
`forms`	Templates	Text
`msg`	Message-Dateien	Text
`tmp`	temporäre Dateien	werden nicht übertragen
`wwwdocs`	Dokumente für den Web-Server	Text
`applet`	Applets	Binär
`class`	Java-Klassen	Binär
`html`	HTML-Seiten	Text
`image`	Bilder	Binär
`javascript`	clientseitige Javascripts	Text
`style`	Stylesheet-Definitionen	Text
eigenes Verzeichnis		Text

Tabelle 3: Übertragungsmodi für die Daten eines Basisverzeichnisses

Beim Entpacken auf den Zielrechner können Sie entscheiden, ob Sie eine neue WebTransactions-Anwendung anlegen wollen, oder ob eine bereits vorhandene WebTransactions-Anwendung ergänzt werden soll. Nach der Übertragung einer WebTransactions-Anwendung werden bei Bedarf in allen Dateien im Unterverzeichnis `wwwdocs/html` die Pfade der WebTransactions CGI-Programme angepasst.

Wenn Sie beim Entpacken bereits mit einer WebTransactions-Anwendung verbunden sind, werden geöffnete entfernte Dateien nicht automatisch neu geladen. WebLab zeigt jedoch an, dass ein neuer Stand der Datei auf dem Server vorliegt und lädt auf Nachfrage diesen neuen Stand.

6.8.2 Anwendung verpacken

Sie verpacken eine Anwendung mit dem Befehl **Administrieren/Anwendung verpacken**.

Während des Verpackens haben Sie zwei Möglichkeiten festzulegen, welche Dateien verpackt werden sollen:

- Durch Einschalten eines Filters können Sie Dateien aus der Baumdarstellung ausblenden. Sie können nach Dateiendungen und/oder Änderungsdatum filtern, siehe auch Abschnitt „Dateiauswahl“ auf Seite 179.

  Dateien, die nicht dargestellt werden, werden auch nicht verpackt.

- Sie können in der Baumdarstellung auch Dateien und Verzeichnisse explizit ein- oder ausschließen. Klicken Sie dafür auf das Symbol das neben jedem Eintrag dargestellt wird.

Alle ausgewählten Dateien bzw. Verzeichnisse werden in das WebTransactions-Archiv gepackt. Im Ausgabebereich von WebLab erhalten Sie in der Registerkarte **Pack&Go** Meldungen über den Fortschritt des Verpackens.

6.8.3 Anwendung entpacken

Um ein Archiv, das eine WebTransactions-Anwendung enthält, wieder zu entpacken, verwenden Sie den Befehl **Administrieren/Anwendung entpacken**.

Wenn Sie bei Aufruf des Befehls mit einem Basisverzeichnis verbunden sind, wird in dieses Basisverzeichnis entpackt.

Wenn Sie nicht mit einem Basisverzeichnis verbunden sind, wird zunächst ein Basisverzeichnis generiert und dann in dieses Verzeichnis entpackt.

Wenn sich die virtuellen Pfade zu `WTPublish.exe`, `WTPublishISAPI.dll` oder `WTCluster.exe` zwischen Ausgangsrechner und Zielrechner unterscheiden, können Sie diese Pfade während des Entpackens anpassen.

Im Ausgabebereich von WebLab erhalten Sie in der Registerkarte **Pack&Go** Meldungen über den Fortschritt des Entpackens.

6.8.4 Anwendung verteilen

Einzelne Dateien oder Verzeichnisse werden jeweils in WebLab eingelesen und dann auf die Zielrechner des Clusters übertragen.

Wenn Sie mehrere Dateien oder Verzeichnisse verteilen wollen, wählen Sie den Befehl **Administrieren/Anwendung verteilen**.

Wie beim Transferieren einer Anwendung (siehe Abschnitt „Anwendung verpacken“ auf Seite 215 und Abschnitt „Anwendung entpacken“ auf Seite 215) werden diese Dateien auch beim Verteilen in eine Archiv-Datei verpackt und diese Datei wird auf allen Rechnern eines Clusters entpackt.

Sie haben die gleichen Möglichkeiten den Umfang des Archivs festzulegen, wie beim Verpacken (siehe Abschnitt „Anwendung verpacken“ auf Seite 215). So können Sie z.B. durch Einsatz eines Filters nur Dateien verteilen, die nach einem bestimmten Datum geändert wurden, siehe Seite 179.

7 Anhang: Demo-Anwendungen

Bei der Installation von WebTransactions können Sie verschiedene Demo-Anwendungen mit installieren. Sie werden im Dokumenten-Verzeichnis des Web-Servers abgelegt.

Die Demo-Anwendungen starten Sie über `http://`*system*`/webtav75/startdemos.htm`, damit wird eine Start-Seite eingeblendet, auf der Sie zwischen verschiedenen Demo-Anwendungen auswählen können. Wenn Sie eine Demo-Anwendung ausgewählt haben, wird auf einer weiteren Seite die Bedienung der Demo-Anwendung erläutert. Die Templates sind ausführlich kommentiert.

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit ->*kursiver* Schrift ausgezeichnet.

aktiver Dialog

Beim aktiven Dialog greift WebTransactions aktiv in die Steuerung des Dialogablaufs ein, d.h., das nächste zu verarbeitende ->*Template* wird von der Template-Programmierung bestimmt. Mit den ->*WTML*-Sprachmitteln können Sie z.B. mehrere ->*Host-Formate* in einer ->*HTML*-Seite zusammenfassen. Dabei wird am Ende eines Host- ->*Dialogschritts* keine Ausgabe an den ->*Browser* geschickt, sondern unmittelbar der Folgeschritt gestartet. Ebenso sind innerhalb **eines** Host-Dialogschritts mehrere Interaktionen zwischen Web- ->*Browser* und WebTransactions möglich.

Array

->*Datentyp*, der eine endliche Menge von Werten eines Datentyps enthalten kann. Der Datentyp kann sein

- ->*skalar*
- eine ->*Klasse*
- ein Array

Die Werte im Array werden durch einen numerischen Index angesprochen, der mit 0 beginnt.

Asynchrone Nachricht

Versteht WebTransactions als Nachricht, die ans Terminal geschickt wird, ohne dass sie vom Anwender ausdrücklich angefordert worden wäre - d.h. ohne dass der Anwender auf irgendeine Taste gedrückt oder auf ein Oberflächenelement geklickt hätte.

Attribut

Definiert eine Eigenschaft eines ->*Objekts*.
Ein Attribut kann z.B. Farbe, Größe oder Position eines Objekts oder selbst wieder ein Objekt sein. Attribute werden auch als ->*Variablen* verstanden und können abgefragt und verändert werden.

Aufrufseite

Eine ->*HTML*-Seite, die Sie benötigen, um eine ->*WebTransactions-Anwendung* zu starten. Auf dieser Seite steht der Aufruf, der WebTransactions mit dem ersten ->*Template* startet, dem Start-Template.

Ausdruck

Kombination aus ->*Literalen*, ->*Variablen*, Operatoren und Ausdrücken, deren Auswertung jeweils ein bestimmtes Ergebnis liefert.

Auswertungsoperator

WebTransactions versteht den Auswertungsoperator als Operator, der die angesprochenen ->*Ausdrücke* durch ihr Ergebnis ersetzt (Objekt-Attribut-Auswertung). Der Auswertungsoperator wird in der Form ##ausdruck# angegeben.

Automask-Template

Ein WebTransactions- ->*Template*, das von WebLab implizit beim Erzeugen eines Basisverzeichnisses oder explizit mit dem Befehl **Automask erzeugen** erstellt wird. Es wird verwendet, wenn kein formatspezifisches Template identifiziert werden kann. Ein Automask-Template enthält die Anweisungen, die für die dynamischen Formatabbildungen und zur Kommunikation notwendig sind. Es können verschiedene Varianten von Automask-Templates erstellt und über das System-Objekt-Attribut AUTOMASK ausgewählt werden.

Basisverzeichnis

Das Basisverzeichnis liegt auf dem WebTransactions-Server und ist die Grundlage einer ->*WebTransactions-Anwendung*. Im Basisverzeichnis liegen die ->*Templates* und alle Dateien oder Verweise auf Programme (Links), die für den Ablauf einer WebTransactions-Anwendung benötigt werden.

BCAM-Applikationsname

Entspricht dem openUTM-Generierungsparameter BCAMAPPL und ist der Name der ->*openUTM-Anwendung*, über den ->*UPIC* die Verbindung aufnehmen kann.

Benutzerkennung

Bezeichner für einen Benutzer. Einer Benutzerkennung können ein ->*Passwort* (zur ->*Zugangskontrolle*) und Zugriffsrechte (->*Zugriffskontrolle*) zugeordnet werden.

Berechtigungsprüfung

siehe ->*Zugangskontrolle*.

Browser

Programm, das zum Abrufen und Darstellen von ->*HTML*-Seiten erforderlich ist. Browser sind z.B. Microsoft Internet Explorer oder Mozilla Firefox.

Browser-Plattform

Betriebssystem des Rechners, auf dem ein ->*Browser* als Client für WebTransactions läuft.

Browserdarstellungs-Druck

Beim Browserdarstellungs-Druck von WebTransactions wird die im ->*Browser* dargestellte Information ausgedruckt.

Capture-Verfahren

Damit WebTransactions in der Ablaufphase die empfangenen ->*Formate* identifizieren kann, können Sie während einer ->*Sitzung* in WebLab für jedes Format einen bestimmten Bereich markieren und das Format benennen. Der Formatname und das ->*Erkennungskriterium* werden in der ->*Capture-Datenbank* gespeichert. Für das Format wird ein ->*Template* unter gleichem Namen generiert. Das Capture-Verfahren ist die Grundlage für die Bearbeitung formatspezifischer Templates für die Liefereinheiten WebTransactions for OSD und MVS.

Capture-Datenbank

Die Capture-Datenbank von WebTransactions enthält alle Formatnamen und die zugehörigen ->*Erkennungskriterien*, die mit dem ->*Capture-Verfahren* erzeugt wurden. Reihenfolge und Erkennungskriterien der Formate können mit WebLab bearbeitet werden.

CGI

(**C**ommon **G**ateway **I**nterface)
Normierte Schnittstelle für den Programmaufruf auf ->*Web-Servern*. Im Gegensatz zur statischen Ausgabe einer zuvor festgelegten ->*HTML*-Seite ermöglicht diese Schnittstelle den dynamischen Aufbau von HTML-Seiten.

Client

Anforderer und Nutzer von Diensten.

Cluster

Menge von identischen ->*WebTransactions-Anwendungen* auf verschiedenen Servern, die zu einem Lastverbund zusammengeschlossen sind.

Dämon

Bezeichnung für einen Prozesstyp in Unix-/POSIX-Systemen, der keine Ein-/Ausgaben auf Terminals durchführt und im Hintergrund abläuft.

Datentyp

Festlegung, wie der Inhalt eines Speicherplatzes zu interpretieren ist. Ein Datentyp hat einen Namen, eine Menge zulässiger Werte (Wertebereich) und eine bestimmte Anzahl von Operationen, die die Werte dieses Datentyps interpretieren und manipulieren.

Dialog

Beschreibt die gesamte Kommunikation zwischen Browser, WebTransactions und *->Host-Anwendung*. Er umfasst in der Regel mehrere *->Dialogzyklen*. Bei WebTransactions werden mehrere Dialogarten unterschieden:

- *->passiver Dialog*
- *->aktiver Dialog*
- *->synchronisierter Dialog*
- *->nicht-synchronisierter Dialog*

Dialogzyklus

Zyklus, der beim Ablauf einer *->WebTransactions-Anwendung* folgende Schritte umfasst:

- eine *->HTML*-Seite aufbauen und an den *->Browser* schicken
- auf Antwort vom Browser warten
- Antwortfelder auswerten und evtl. zur Weiterverarbeitung an die *->Host-Anwendung* schicken

Während des Ablaufs einer *->WebTransactions-Anwendung* werden mehrere Dialogzyklen durchlaufen.

Distinguished Name

Der Distinguished Name (DN) in *->LDAP* setzt sich hierarchisch aus mehreren Teilen zusammen (z.B. „Land, unterhalb von Land: Organisation, unterhalb von Organisation: Organisationseinheit, darunter: Gebräuchlicher Name“). Die Summe dieser Teile identifiziert ein Objekt innerhalb des Directory-Baums eindeutig.

Durch diese Hierarchie wird die eindeutige Benennung von Objekten selbst in einem weltweiten Directory-Baum sehr einfach:

- Der DN "Land=DE/Name=Emil Mustermann" reduziert das Eindeutigkeits-Problem auf das Land DE.
- Der DN "Organisation=FTS/Name=Emil Mustermann" reduziert es auf die Organisation FTS.
- Der DN "Land=DE/Organisation=FTS/Name=Emil Mustermann" reduziert es auf die Organisation FTS innerhalb des Landes DE.

Dokumentenverzeichnis

Verzeichnis des ->*Web-Servers*, in dem Dokumente liegen, auf die über das Netz zugegriffen werden kann. WebTransactions legt in diesem Verzeichnis Dateien zum Herunterladen ab, wie z.B. den WebLab-Client oder allgemeine Start-Seiten.

Domain Name Service (DNS)

Verfahren zur symbolischen Adressierung von Rechnern in Netzen. Bestimmte Rechner im Netz, die DNS- oder Name-Server, führen eine Datenbank mit allen bekannten Rechnernamen und IP-Nummern in ihrer Umgebung.

Dynamische Daten

Werden in WebTransactions durch das WebTransactions-Objektmodell abgebildet, z.B. als ->*Systemobjekt*, Host-Objekt oder Nutzereingaben am Browser.

Eigenschaft

Definiert die Beschaffenheit von ->*Objekten*, z.B. könnten Kundenname und Kundennummer Eigenschaften eines Objekts „Kunde“ sein. Diese Eigenschaften können innerhalb des Programms gesetzt, abgefragt und verändert werden.

EJB

(**E**nterprise **J**ava**B**ean)
Industriestandard auf Basis von Java, mit dem innerhalb einer verteilten, objektorientierten Umgebung selbstentwickelte oder auf dem Markt gekaufte Server-Komponenten zur Erstellung von verteilten Programmsystemen genutzt werden können.

EHLLAPI

(**E**nhanced **H**igh **L**evel **L**angugage **API**)
Programmschnittstelle z.B. von Terminal-Emulationen für die Kommunikation mit der SNA-Welt. Auf dieser Schnittstelle basiert die Kommunikation zwischen Transit-Client und dem SNA-Rechner, die über das Produkt TRANSIT abgewickelt wird.

Erkennungskriterium

Über Erkennungskriterien werden ->*Formate* einer ->*Terminal-Anwendung* identifiziert und Sie können auf die Daten des Formats zugreifen. Als Erkennungskriterium wählen Sie jeweils einen oder auch mehrere Bereiche des Formats, deren Inhalt das Format eindeutig identifiziert.

Felddatei (*.fld-Datei)

Enthält in WebTransactions die Struktur des Datensatzes eines ->*Formats* (Metadaten).

FHS

(**F**ormat **H**andling **S**ystem)
Formatierungssystem für BS2000/OSD-Anwendungen.

Field

Kleinster Teil eines ->*Service* und Element eines ->*Records* oder ->*Puffers*.

Filter

Programm oder Programmteil (z.B. eine Bibliothek) zur Umsetzung eines Formats in ein anderes (z.B. XML-Dokumente in ->*WTScript*-Datenstrukturen).

Format

Optische Darstellung auf alphanumerischen Bildschirmen, wird auch Maske oder Schirm genannt.

In WebTransactions wird ein Format jeweils durch eine ->*Felddatei* und ein Template repräsentiert.

Formatbeschreibungsquellen

Beschreibung mehrerer ->*Formate* in einer oder mehreren Dateien, die aus einer Format-Bibliothek (FHS/IFG) erzeugt wurden oder direkt am ->*Host* vorliegen für die Nutzung „sprechender“ Namen in Formaten.

Formattyp

(nur relevant bei FHS-Anwendungen und Kommunikation über UPIC)
Spezifiziert den Typ des Formats: #Format, +Format, -Format oder *Format.

Funktion

Benutzerdefinierte Code-Teile mit einem Namen und ->*Parametern*. Durch eine Beschreibung der Funktionsschnittstelle (oder Signatur) können Funktionen in Methoden aufgerufen werden.

Holder Task

Prozess, Task oder Thread in WebTransactions, je nach Betriebssystem-Plattform. Die Anzahl der Tasks entspricht der Anzahl der Benutzer. Die Task wird beendet, wenn sich der Benutzer abmeldet oder durch Timeout. Ein Holder Task entspricht genau einer ->*WebTransactions-Sitzung*.

Host

Rechner, auf dem die ->*Host-Anwendung* läuft.

Host-Adapter

Dienen dazu, bestehende *->Host-Anwendungen* an WebTransactions anzuschließen. Sie sorgen zur Laufzeit z.B. für den Auf- und Abbau von Verbindungen und für die Umsetzung der ausgetauschten Daten.

Host-Anwendung

Anwendung, die mit WebTransactions integriert ist.

Host-Plattform

Betriebssystem des Rechners, auf dem die *->Host-Anwendung* läuft.

Host-Daten-Druck

Beim Host-Daten-Druck von WebTransactions werden Informationen ausgedruckt, die von der *->Host-Anwendung* aufbereitet und gesendet wurden, z.B. Ausdruck von Host-Dateien.

Host-Datenobjekt

Bezeichnet in WebTransactions ein *->Objekt* der Datenschnittstelle zur *->Host-Anwendung*, das ein Feld mit all seinen Feldattributen repräsentiert. Es wird von WebTransactions nach dem Empfang von Daten der Host-Anwendung angelegt und existiert bis zum nächsten Datenempfang oder bis zum Beenden der *->Sitzung*.

Host-Steuerobjekt

In WebTransactions enthalten Host-Steuerobjekte Informationen, die nicht nur ein einzelnes Feld betreffen, sondern das gesamte *->Format*. Dazu gehören z.B. das Feld, in dem sich der Cursor befindet, die aktuelle Funktionstaste oder globale Formatattribute.

HTML

(**H**yper**t**ext **M**arkup **L**anguage)
Siehe *->Hypertext Markup Language*

HTTP

(**H**ypertext **T**ransfer **P**rotocol)
Protokoll zur Übertragung von *->HTML*-Seiten und Daten.

HTTPS

(**H**ypertext **T**ransfer **P**rotocol **S**ecure)
Protokoll zur gesicherten Übertragung von *->HTML*-Seiten und Daten.

Hypertext

Dokument mit Verweisen auf andere Stellen im gleichen oder in anderen Dokumenten, in die z.B. durch Anklicken mit der Maus gesprungen werden kann.

Hypertext Markup Language
Standardisierte Auszeichnungssprache für Dokumente im WWW.

JavaBean
Java-Programm (oder ->*Klasse*) mit genau festgelegten Konventionen für die Schnittstellen, die eine Wiederverwendung in mehreren Anwendungen ermöglichen.

KDCDEF
openUTM-Werkzeug für die Generierung von ->*openUTM-Anwendungen*.

Klasse
Enthält die Definition der ->*Eigenschaften* und ->*Methoden* eines ->*Objekts*. Sie ist das Modell für die Instanziierung von Objekten und definiert deren Schnittstellen.

Klassen-Template
Ein Klassen-Template in WebTransactions enthält für die gesamte Objektklasse (z. B. Eingabe- oder Ausgabefeld) gültige, immer wiederkehrende Anweisungen. Klassen-Templates werden durchlaufen, wenn auf ein ->*Host-Datenobjekt* der ->*Auswertungsoperator* oder die `toString`-Methode angewendet wird.

Kommunikationsobjekt
Steuert eine Verbindung zu einer ->*Host-Anwendung* und enthält Information über den aktuellen Zustand der Verbindung, über die zuletzt empfangenen Daten etc.

Konvertierungswerkzeuge
Dienstprogramme, die mit WebTransactions ausgeliefert werden. Mit den Konvertierungswerkzeugen werden die Datenstrukturen von ->*openUTM-Anwendungen* analysiert und in Dateien abgelegt. Diese Dateien können Sie dann in WebLab als ->*Formatbeschreibungsquellen* verwenden, um WTML-Templates und ->*FLD-Dateien* zu generieren.
Die Basis für die Konvertierung können Cobol-Datenstrukturen oder IFG-Formatbibliotheken sein. Für Drive-Programme wird das Konvertierungswerkzeug mit dem Produkt Drive ausgeliefert.

LDAP

(**L**ightweight **D**irectory **A**ccess **P**rotocol)
Der X.500-Standard definiert als Zugriffsprotokoll DAP (Directory Access Protocol). Speziell für den Zugang zu X.500-Verzeichnisdiensten vom PC aus hat sich jedoch der Internet-Standard LDAP durchgesetzt.
Bei LDAP handelt es sich um ein vereinfachtes DAP-Protokoll, das nicht alle Möglichkeiten von DAP zulässt und mit DAP nicht kompatibel ist. Praktisch alle X.500-Verzeichnisdienste unterstützen neben DAP auch LDAP. In der Praxis kann es zu Verständigungsproblemen kommen, da es diverse Dialekte von LDAP gibt. Die Unterschiede der Dialekte sind in der Regel gering.

Literal

Zeichenfolge, die einen festen Wert repräsentiert. Literale dienen dazu, in Source-Programmen konstante Werte unmittelbar anzugeben („wörtliche“ Wertangabe).

Master-Template

WebTransactions-Template, das als Schablone für die Generierung der Automask und der formatspezifischen-Templates verwendet wird.

Message Queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete Warteschlangen (Queues) ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen, die Übermittlung der Nachricht wird garantiert, unabhängig davon, ob gerade eine Netzverbindung besteht oder nicht.

Methode

Objektorientierter Begriff für ->*Funktion*. Eine Methode wirkt auf das ->*Objekt*, in dem sie definiert ist

Modul-Template

Dient in WebTransactions dazu, ->*Klassen*, ->*Funktionen* und Konstanten global für eine komplette ->*Sitzung* zu definieren. Ein Modul-Template wird mit Hilfe der Funktion `import()` geladen.

MT-Tag

(**M**aster-**T**emplate-Tag)
Spezielle Tags in ->*Master-Templates* für die dynamischen Teile eines Master-Templates.

Multi-Tier-Architektur

Allen Client-/Server-Architekturen liegt eine Gliederung in einzelne Software-Komponenten, auch Schichten oder Tiers genannt, zugrunde: Man spricht von 1-Tier, 2-Tier-, 3-Tier und auch von Multi-Tier-Modellen. Man kann die Aufgliederung auf der physischen oder der logischen Ebene betrachten:

– Logische Software-Tiers liegen vor, wenn die Software in modulare Komponenten mit klaren Schnittstellen gegliedert ist.
– Physische Tiers liegen dann vor, wenn die (logischen) Softwarekomponenten im Netz auf verschiedene Rechner verteilt sind.

Mit WebTransactions sind Multi-Tier-Modelle sowohl auf physischer als auch logischer Tiers-Ebene möglich.

Name/Value-Paar

In den vom ->*Browser* geschickten Daten die Kombination z.B. von einem ->*HTML*-Eingabefeldnamen mit seinem Wert.

nicht-synchronisierter Dialog

Der nicht-synchronisierte Dialog von WebTransactions erlaubt es, den Prüfmechanismus des ->*synchronisierten Dialogs* zeitweise auszuschalten. So lassen sich ->*Dialoge* zwischenschieben, die außerhalb des synchronisierten Dialogs liegen und keinen Einfluss auf den logischen Zustand der ->*Host-Anwendung* haben. Dadurch können Sie z.B. in einer ->*HTML*-Seite eine Schaltfläche anbieten, um Hilfeinformationen aus der laufenden Host-Anwendung anzufordern und in einem separaten Fenster anzuzeigen.

Objekt

Elementare Einheit innerhalb eines objektorientierten Softwaresystems. Jedes Objekt hat einen Namen, über den es angesprochen werden kann, ->*Attribute*, die seinen Zustand definieren und ->*Methoden*, die auf das Objekt angewandt werden können.

openUTM

(**U**niversal **T**ransaction **M**onitor)
Transaktionsmonitor von Fujitsu Technlogy Solutions, verfügbar für BS2000/OSD, verschiedenste Unix- und Windows-Plattformen.

openUTM-Anwendung

->*Host-Anwendung*, die Dienstleistungen zur Verfügung stellt, die Aufträge von Terminals, ->*Client-Programmen* oder anderen Host-Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine openUTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Einheit bildet.
Mit openUTM-Anwendungen kann sowohl über das Client/Server-Protokoll ->*UPIC* als auch über die Terminal-Schnittstelle (9750) kommuniziert werden.

openUTM-Client (UPIC)

Mit dem Produkt openUTM-Client (UPIC) können Sie Client-Programme für openUTM erstellen. openUTM-Client (UPIC) steht z.B. für Unix-, BS2000/OSD- und Windows-Plattformen zur Verfügung.

openUTM-Teilprogramm

Die Dienste einer ->*openUTM-Anwendung* werden durch ein oder mehrere openUTM-Teilprogramme realisiert. Sie sind über ->*Transaktionscodes* ansprechbar und enthalten spezielle openUTM-Funktionsaufrufe (z.B. KDCS-Aufrufe).

Parameter

Daten, die an eine ->*Funktion* oder ->*Methode* zur Verarbeitung übergeben werden (Eingabeparameter) oder Daten, die als Ergebnis von einer Funktion oder Methode zurückgeliefert werden (Ausgabeparameter).

passiver Dialog

Beim passiven Dialog von WebTransactions wird der Dialogablauf von der ->*Host-Anwendung* gesteuert, d.h., die Host-Anwendung bestimmt das nächste zu verarbeitende ->*Template*. Ein Anwender, der über WebTransactions auf die Host-Anwendung zugreift, durchläuft die gleichen Schritte wie beim Zugriff über ein Terminal. Passive Dialogsteuerung verwendet WebTransactions bei einer automatischen Umsetzung der Host-Anwendung oder wenn jedes Format der Host-Anwendung genau einem individuellen Template entspricht.

Passwort

In einer Anwendung für eine ->*Benutzerkennung* eingetragene Zeichenkette zur Authentisierung (->*Zugangsschutz*).

polling

Zyklische Abfrage auf Zustandsänderungen.

Pool

WebTransactions bezeichnet hiermit ein freigegebenes Verzeichnis, in dem WebLab ->*Basisverzeichnisse* anlegen und pflegen kann. Den Zugriff auf dieses Verzeichnis steuern Sie mit der Administration.

Posted-Objekt (`wt_Posted`**)**

Enthält in WebTransactions eine Liste der vom ->*Browser* zurückgeschickten Daten. Dieses ->*Objekt* wird von WebTransactions angelegt und lebt nur für die Dauer eines ->*Dialogzyklus*.

posten

Daten versenden

Projekt

Enthält in der WebTransactions-Entwicklungsumgebung verschiedene Einstellungen einer ->*WebTransactions-Anwendung*, die in einer Projektdatei (Endung `.wtp`) gespeichert werden. Sie sollten für jede WebTransactions-Anwendung, die Sie entwickeln, ein Projekt anlegen und zum Bearbeiten immer dieses Projekt öffnen.

Protokoll

Vereinbarungen über Verhaltensregeln und Formate bei der Kommunikation unter entfernten Partnern gleichen logischen Niveaus.

Protokolldatei

- openUTM-Client: Datei, in die bei abnormalem Beenden einer Conversation openUTM-Fehlermeldungen geschrieben werden.
- In WebTransactions werden Protokolldateien als Trace-Dateien bezeichnet.

Prozess

Der Begriff „Prozess“ wird als Oberbegriff für Prozess (Solaris, Linux und Windows) und Task (BS2000/OSD) verwendet.

Puffer

Definition eines Datensatzes, der von einem ->*Service* übertragen wird. Der Puffer dient zum Senden und zum Empfangen von Nachrichten. Zusätzlich gibt es einen speziellen Puffer für die Ablage der ->*Erkennungskriterien* und für Daten zur Darstellung am Bildschirm.

Roaming Session

->*WebTransactions-Sitzung*, die nacheinander oder gleichzeitig von verschiedenen ->*Clients* aus angesprochen werden kann.

Record

Definition eines Datensatzes, der in einem ->*Puffer* übertragen wird. Er beschreibt einen Teil des Puffers, der ein- oder mehrfach vorkommen kann.

Service-Anwendung

->*WebTransactions-Sitzung*, die abwechselnd von verschiedenen Benutzern aufgerufen werden kann.

Service-Knoten

Instanz eines ->*Service*. Beim Entwickeln und beim Ablauf einer ->*Methode* kann ein Service mehrfach instanziiert werden. Beim Modellieren und Code bearbeiten werden diese Instanzen als Service-Knoten bezeichnet.

Sichtbarkeit von Variablen

->*Objekte* und ->*Variablen* unterschiedlicher Dialogarten werden von WebTransactions in unterschiedlichen Adressräumen verwaltet. Das bedeutet, dass Variablen eines ->*synchronen Dialogs* im ->*asynchronen Dialog* oder im Dialog mit einer entfernten Anwendung nicht sichtbar und damit auch nicht zugreifbar sind.

Sitzung

Beginnt ein Endanwender mit einer ->*WebTransactions-Anwendung* zu arbeiten, so wird für ihn auf dem WebTransactions-Server eine WebTransactions-Sitzung eingerichtet. Diese Sitzung enthält alle für diesen Benutzer geöffneten Verbindungen zum ->*Browser*, zu speziellen ->*Clients* und ->*Hosts*.
Eine Sitzung kann gestartet werden

- durch Eingabe eines URL von WebTransactions im Browser.
- durch die Methode `START_SESSION` der Client/Server-Schnittstelle `WT_REMOTE`.

Eine Sitzung endet

- mit einer entsprechenden Eingabe des Benutzers im Ausgabebereich dieser ->*WebTransactions-Anwendung* (nicht über Standard-Buttons des Browsers).
- durch Überschreiten der konfigurierten Zeit, die WebTransactions auf eine Anwort von der ->*Host-Anwendung* oder vom ->*Browser* wartet.
- durch Terminierung mit HIlfe der WebTransactions-Administration.
- durch die Methode `EXIT_SESSION` der Client/Server-Schnittstelle `WT_REMOTE`.

Eine WebTransactions-Sitzung ist eindeutig durch eine ->*WebTransactions-Anwendung* und eine Session Id bestimmt. Während ihrer Lebensdauer existiert zu jeder WebTransactions-Sitzung auf dem WebTransactions-Server genau ein ->*Holder Task*.

Skalar

->*Variable*, die nur aus einem einzelnen Wert besteht - im Gegensatz zu einer ->*Klasse*, einem ->*Array* oder einer anderen komplexen Datenstruktur.

SOAP

(ursprünglich **S**imple **O**bject **A**ccess **P**rotocol)
Das ->*XML*-basierte SOAP-Protocol realisiert einen einfachen und transparenten Mechanismus, mit dem strukturierte und typisierte Informationen zwischen Rechnern in einer dezentralisierten, verteilten Umgebung ausgetauscht werden können.
SOAP stellt ein modulares Paketmodell sowie Mechanismen zum Verschlüsseln von Daten innerhalb von Modulen zur Verfügung. Dies ermöglicht die unkomplizierte Beschreibung der externen Schnittstellen eines ->*Web-Service*.

Stil

Realisiert in WebTransactions ein anderes Layout für ein ->*Template*, z.B. mit mehr oder weniger Grafikelementen für unterschiedliche ->*Browser*. Der Stil kann während einer ->*Sitzung* jederzeit geändert werden.

synchronisierter Dialog

Beim synchronisierten Dialog (Standardfall) überprüft WebTransactions automatisch, ob die Daten, die vom Web-Browser eingehen, auch wirklich die Antwort auf die letzte an den ->*Browser* geschickte ->*HTML*-Seite sind. Wenn z.B. der Anwender am Web-Browser über die Schaltfläche **Zurück** oder die History-Funktion zu einer „alten“ HTML-Seite der aktuellen ->*Sitzung* wechselt und diese zurückschickt, erkennt WebTransactions, dass die Daten nicht zum aktuellen ->*Dialogzyklus* passen und reagiert mit einer Fehlermeldung. Die zuletzt an den Browser gesendete Seite wird automatisch erneut an den Browser geschickt.

Systemobjekt (`wt_System`**)**

Das Systemobjekt von WebTransactions enthält ->*Variablen*, die während einer gesamten ->*Sitzung* existieren und erst am Ende einer Sitzung oder durch explizites Löschen wieder entfernt werden. Es ist immer sichtbar und identisch für alle Namensräume.

TAC

Siehe ->*Transaktionscode*

Tag

->*HTML*-, ->*XML*- und ->*WTML*-Dokumente bestehen aus Tags und dem eigentlichen Inhalt. Mit den Tags werden Auszeichnungen im Dokument durchgeführt z.B. Überschriften, Texthervorhebungen (fett, kursiv) oder Quellangaben für Grafikdateien.

TCP/IP

(**T**ransport **C**ontrol **P**rotocol/**I**nternet **P**rotocol)
Sammelname für eine Protokollfamilie in Rechnernetzen, die unter anderem im Internet verwendet wird.

Template

Vorlage für die Generierung von spezifischem Code. Ein Template enthält feste Teile, die bei der Generierung unverändert übernommen werden und variable Teile, die bei der Generierung durch die jeweils aktuellen Werte ersetzt werden. Ein Template ist eine *->WTML*-Datei mit speziellen Tags zur Steuerung der dynamischen Generierung einer *->HTML*-Seite und zur Verarbeitung der am *->Browser* eingegeben Werte. Es können mehrere Sätze von Templates parallel gehalten werden. Diese repräsentieren unterschiedliche Stile (z.B. viel/wenig Grafik, Java-Benutzung etc.).
WebTransactions nutzt verschiedene Arten von Templates:

- ->*Automask-Templates* für die automatische Umsetzung der ->*Formate* von MVS- und OSD-Anwendungen
- eigene Templates, die vom Programmierer selbst geschrieben werden, z.B. zur Steuerung eines *->aktiven Dialogs*
- formatspezifische Templates, die für eine spätere Nachbearbeitung generiert werden
- Include-Templates, die in andere Templates eingefügt werden
- *->Klassen-Templates*
- ->*Master-Templates* für ein einheitliches Layout fester Bereiche bei der Generierung der Automask und formatspezifischer Templates
- Start-Template, das als erstes Template einer WebTransactions-Anwendung durchlaufen wird

Template-Objekte

->Variablen zur Zwischenspeicherung von Werten für einen *->Dialogzyklus* in WebTransactions.

Terminal-Anwendung

Anwendung auf einem ->*Host*-Rechner, auf die über die 9750- oder 3270-Schnittstelle zugegriffen wird.

Terminal-Hardcopy-Druck

Beim Terminal-Hardcopy-Druck von WebTransactions wird die alphanumerische Darstellung des ->*Formats* gedruckt, wie es von einem Terminal oder einer Terminal-Emulation dargestellt würde.

Transaktion

Verarbeitungsschritt zwischen zwei Sicherungspunkten (innerhalb eines Vorgangs), der durch die ACID-Bedingungen gekennzeichnet ist (**A**tomicity, **C**onsistency, **I**solation und **D**urability). Die in einer Transaktion beabsichtigten Änderungen an der Anwenderinformation werden entweder alle oder gar nicht durchgeführt (Alles-oder-Nichts-Regel).

Transaktionscode/TAC

Name, über den ein openUTM-Vorgang oder ein *->openUTM-Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem openUTM-Teilprogramm bei der openUTM-Konfigurierung zugeordnet. Einem Teilprogramm können auch mehrere TACs zugeordnet sein.

UDDI

(**U**niversal **D**escription, **D**iscovery and **I**ntegration)
Umfasst Verzeichnisse, die Beschreibungen von *->Web-Services* enthalten. Diese Informationen stehen Web-Usern allgemein zur Verfügung.

Unicode

Von der International Standardisation Organisation (ISO) und dem Unicode-Konsortium genormter alphanumerischer Zeichensatz zur Codierung von Zeichen – Buchstaben, Ziffern, Satzzeichen, Silbenzeichen, Sonderzeichen sowie Ideogrammen. Unicode fasst alle weltweit bekannten Textzeichen in einem einzigen Zeichensatz zusammen.
Unicode ist hersteller- und systemunabhängig. Es verwendet Zeichensätze der Länge zwei oder vier Bytes für die Codierung jedes Textzeichens. Diese Zeichensätze werden bei ISO als UCS-2 (Universal Character Set 2) beziehungsweise UCS-4 bezeichnet. Statt der durch ISO definierten Bezeichnung UCS-2 wird häufig die Bezeichnung UTF-16 (Unicode Transformation Format 16 Bit) verwendet, ein vom Unicode-Konsortium definierter Standard.
Neben der Nutzung von UTF-16 ist auch der Einsatz von UTF-8 (Unicode Transformation Format 8 Bit) weit verbreitet. UTF-8 ist inzwischen die globale Zeichen-Codierung im Internet.

UPIC

(**U**niversal **P**rogramming **I**nterface for **C**ommunication)
Trägersystem für openUTM-Clients, das über die X/Open-Schnittstelle CPI-C die Client-Server-Kommunikation zwischen CPI-C-Client-Anwendung und der openUTM-Anwendung ermöglicht.

URI

(**U**niform **R**esource **I**dentifier)
Oberbegriff für alle Namen und Adressen die im Internet Objekte referenzieren. Die allgemein gebräuchlichen URIs sind *->URLs*.

URL

(**U**niform **R**esource **L**ocator)
Beschreibung von Ort und Zugriffsart einer Ressource im Internet.

Userexit

In C/C++ implementierte Funktion, die der Programmierer aus einem *->Template* aufruft.

Variable

Speicherplatz für variable Werte, der einen Namen und einen ->*Datentyp* benötigt.

Vorgang

In ->*openUTM* Bearbeitung eines Auftrags durch eine *->openUTM-Anwendung*. Es gibt Dialog-Vorgänge und Asynchronvorgänge. Dem Vorgang werden von openUTM eigene Speicherbereiche zugeordnet. Ein Vorgang setzt sich aus einer oder mehreren ->*Transaktionen* zusammen.

Web-Server

Rechner und Software zum Bereitstellen von *->HTML*-Seiten und dynamischen Daten über *->HTTP.*

Web-Service

Dienst, der im Internet bereitgestellt wird, z.B. ein Währungsumrechnungs-Programm, und über das SOAP-Protokoll angesprochen werden kann. Die Schnittstelle eines Web-Service ist in ->*WSDL* beschrieben.

WebTransactions-Anwendung

Anwendung, die *->Host-Anwendungen* für den Internet-/Intranet-Zugriff integriert. Eine WebTransactions-Anwendung besteht aus

- einem *->Basisverzeichnis*
- einem Start-Template
- den *->Templates*, die die Umsetzung zwischen ->*Host* und ->*Browser* steuern
- protokollspezifischen Konfigurationsdateien

WebTransactions-Plattform

Betriebssystem des Rechners, auf dem WebTransactions läuft.

WebTransactions-Server

Rechner, auf dem WebTransactions läuft.

WebTransactions-Sitzung

Siehe *->Sitzung*.

WSDL

(**W**eb **S**ervices **D**escription **L**anguage)
Bietet *->XML*-Sprachregeln für die Beschreibung von *->Web-Services*. Ein Web-Service wird dabei durch eine Auswahl von Ports definiert.

WTBean

In WebTransactions werden *->WTML*-Komponenten mit selbstbeschreibender Schnittstelle als WTBeans bezeichnet. Es wird zwischen inline und standalone WTBeans unterschieden:

– ein inline WTBean entspricht einem Teil eines WTML-Dokuments

– ein standalone WTBean ist ein eigenständiges WTML-Dokument

Verschiedene WTBeans gehören zum Produktumfang von WebTransactions, weitere WTBeans stehen Ihnen auf der WebTransactions-Homepage zum Download zur Verfügung:
ts.fujitsu.com/products/software/openseas/webtransactions.html

WTML

(**W**eb**T**ransactions **M**arkup **L**anguage)
Auszeichnungs- und Programmiersprache für WebTransactions *->Templates*. WTML besteht aus *->WTML-Tags*, die *->HTML* erweitern, und der server-seitigen Programmiersprache *->WTScript*, die z.B. den Datenaustauch mit *->Host-Anwendungen* ermöglicht. WTML wird von WebTransactions und nicht vom *->Browser* ausgeführt (serverside scripting).

WTML-Tag

(**W**eb**T**ransactions **M**arkup **L**anguage-Tag)
Spezielle Tags von WebTransactions zur Generierung der dynamischen Teile einer *->HTML*-Seite mit Daten aus der *->Host-Anwendung*.

WTScript

Server-seitige Programmiersprache von WebTransactions. WTScripts stehen ähnlich wie client-seitige JavaScripts in Bereichen, die mit speziellen Tags eingeleitet und beendet werden. Statt *->HTML*-`SCRIPT`-Tags verwenden Sie hierfür jedoch *->WTML-Tags*: `wtOnCreateScript` und `wtOnReceiveScript`. Damit zeigen Sie an, dass diese Scripts von WebTransactions und nicht vom *->Browser* ausgeführt werden sollen und signalisieren zusätzlich den gewünschten Ausführungszeitpunkt. OnCreate-Scripts werden ausgeführt, bevor die Seite an den Browser geschickt wird. OnReceive-Scripts werden erst ausgeführt, nachdem die Antwort vom Browser empfangen wurde.

XML

(e**X**tensible **M**arkup **L**anguage)
Definiert eine Sprache zur logischen Strukturierung von Dokumenten mit dem Ziel, diese einfach zwischen verschiedenen Anwendungen auszutauschen.

XML-Schema

Ein XML-Schema im allgemeinen Sinn definiert die zulässigen Elemente und Attribute einer XML-Beschreibung. XML-Schemata können verschiedene Formate haben, z.B. DTD (**D**ocument **T**ype **D**efinition), XML Schema (W3C-Standard) oder XDR (**X**ML **D**ata **R**educed).

Zugangskontrolle

Prüfung, ob ein Benutzer berechtigt ist, unter einer bestimmten Benutzerkennung mit der Anwendung zu arbeiten.

Zugriffskontrolle

Überwachung der Zugriffe auf die Daten und ->*Objekte* einer Anwendung.

Abkürzungen

BO	**B**usiness **O**bject
CGI	**C**ommon **G**ateway **I**nterface
DN	**D**istinguished **N**ame
DNS	**D**omain **N**ame **S**ervice
EJB	**E**nterprise **J**ava**B**ean
FHS	**F**ormat **H**andling **S**ystem
HTML	**H**yper**t**ext **M**arkup **L**anguage
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
HTTPS	**H**yper**t**ext **T**ransfer **P**rotocol **S**ecure
IFG	**I**nteraktiver **F**ormat **G**enerator
ISAPI	**I**nternet **S**erver **A**pplication **P**rogramming **I**nterface
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
LPD	**L**ine **P**rinter **D**aemon
MT-Tag	**M**aster-**T**emplate-Tag
MVS	**M**ultiple **V**irtual **S**torage
OSD	**O**pen **S**ystems **D**irection
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage
SOAP	**S**imple **O**bject **A**ccess **P**rotocol

SSL	**S**ecure **S**ocket **L**ayer
TCP/IP	**T**ransport **C**ontrol **P**rotocol/**I**nternet **P**rotocol
Upic	**U**niversal **P**rogramming **I**nterface for **C**ommunication
URL	**U**niform **R**esource **L**ocator
WSDL	**W**eb **S**ervices **D**escription **L**anguage
wtc	**W**eb**T**ransactions **C**omponent
WTML	**W**eb**T**ransactions **M**arkup **L**anguage
XML	e**X**tensible **M**arkup **L**anguage

Literatur

WebTransactions-Handbücher

Unter der Web-Adresse *http://manuals.ts.fujitsu.com* stehen Ihnen sämtliche Handbücher zum Download zur Verfügung.

WebTransactions
Template-Sprache
Referenzhandbuch

WebTransactions
Client-APIs für WebTransactions
Benutzerhandbuch

WebTransactions
Anschluss an openUTM-Anwendungen über UPIC
Benutzerhandbuch

WebTransactions
Anschluss an OSD-Anwendungen
Benutzerhandbuch

WebTransactions
Anschluss an MVS-Anwendungen
Benutzerhandbuch

WebTransactions
Zugriff auf dynamische Web-Inhalte
Benutzerhandbuch

WebTransactions
Web-Frontend für Web-Services
Benutzerhandbuch

Stichwörter

C

T