Benutzerhandbuch - Deutsch

FUJITSU

WebTransactions V7.5

Web-Frontend für Web-Services

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 Architektur des Web-Frontends für Web-Services

Folgende Abbildung zeigt die Architektur des Web-Frontends für Web-Services:

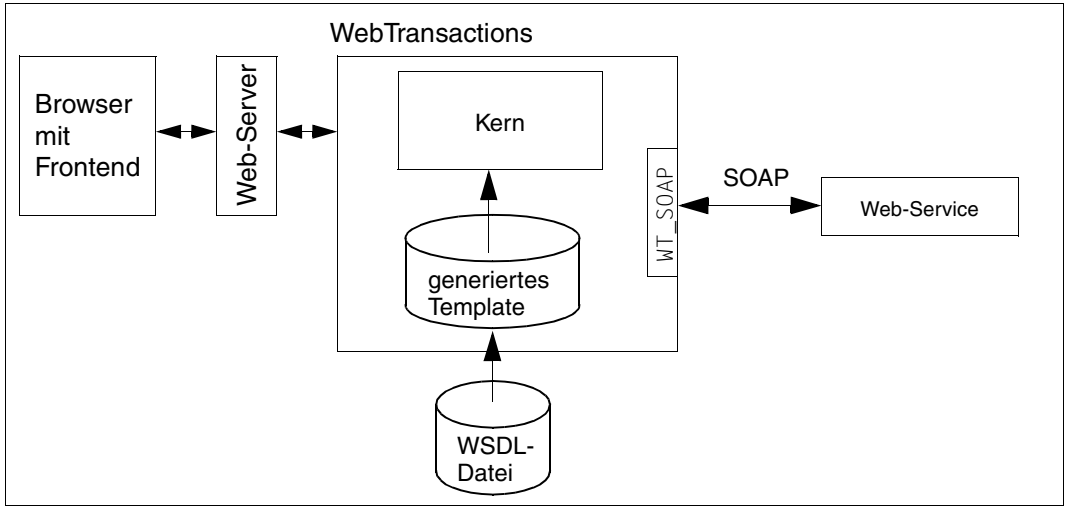


Bild 1: Architektur des Web-Frontends für Web-Services

WebTransactions stellt mit dem Web-Frontend für Web-Services eine Oberfläche zur Verfügung, um allgemeine Web-Services aufzurufen.

1.3 Dokumentation zu WebTransactions

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

- Ein einführendes Handbuch, das für alle Liefereinheiten gilt:

  Konzepte und Funktionen

  Das Handbuch beschreibt alle zentralen Konzepte von WebTransactions:

  – die unterschiedlichen Einsatzmöglichkeiten von WebTransactions.

  – das Konzept von WebTransactions und die Bedeutung der Objekte in WebTransactions, ihre wesentlichen Eigenschaften und Methoden, ihr Zusammenspiel und ihre Lebensdauer.

  – den dynamischen Ablauf einer WebTransactions-Anwendung.

  – die Administration von WebTransactions.

  – die Entwicklungsumgebung WebLab.

- 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.

1.4 Konzept und Zielgruppe dieses Handbuchs

Diese Dokumentation wendet sich an alle, die allgemeine Web-Services an WebTransactions anschließen wollen. Die einzelnen Kapitel beschreiben die hierfür notwendigen Schritte.

Das Handbuch ergänzt das einführende WebTransactions-Handbuch „Konzepte und Funktionen“ und das WebTransactions-Referenzhandbuch „Template-Sprache“ um die Informationen, die Sie für den Anschluss von allgemeinen Web-Services an WebTransactions benötigen.

Gültigkeit der Beschreibung

Web-Frontend für Web-Services ist auf den Systemplattformen Solaris, Linux, Windows und BS2000/OSD ablauffähig. Diese Dokumentation gilt für alle Plattformen. Falls sich eine Information speziell auf eine bestimmte Plattform bezieht, wird jeweils ausdrücklich darauf hingewiesen.

1.5 Neue Funktionen

Einen Überblick über alle Neuerungen in WebTransactions V7.5 finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

Die Unterstützung von Business Objekten entfällt.

1.6 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 Konzept des Web-Frontends

Mit WebTransactions können Sie beliebige Web-Services über die Schnittstelle WT_SOAP in Ihre WebTransactions-Anwendung integrieren. Die Einbettung des SOAP-Protokolls in WebTransactions ist im WebTransactions-Handbuch „Zugriff auf dynamische Web-Inhalte“ beschrieben.

Über die Funktionen der Schnittstelle WT_SOAP hinaus bietet WebTransactions eine Basis-Oberfläche – ein sogenanntes Web-Frontend – für die Integration von Web-Services, die im Netzwerk zur Verfügung stehen.

Das Web-Frontend ist in einem Template realisiert, das Sie mit WebLab generieren können. Es übernimmt die Darstellung am Browser und steuert die Kommunikation mit dem Web-Service. WebLab benötigt für die Generierung dieses Templates eine Beschreibungsdatei im WSDL-Format (**W**eb **S**ervice **D**escription **L**anguage), die die Operationen des Web-Service beschreibt.

Über das Web-Frontend können Sie Benutzern die Funktionalität des Web-Service zur Verfügung stellen. Oder Sie verwenden es, um sich einen Überblick über die Funktionalität zu verschaffen, und passen dann das generierte Template an Ihre Bedürfnisse an.

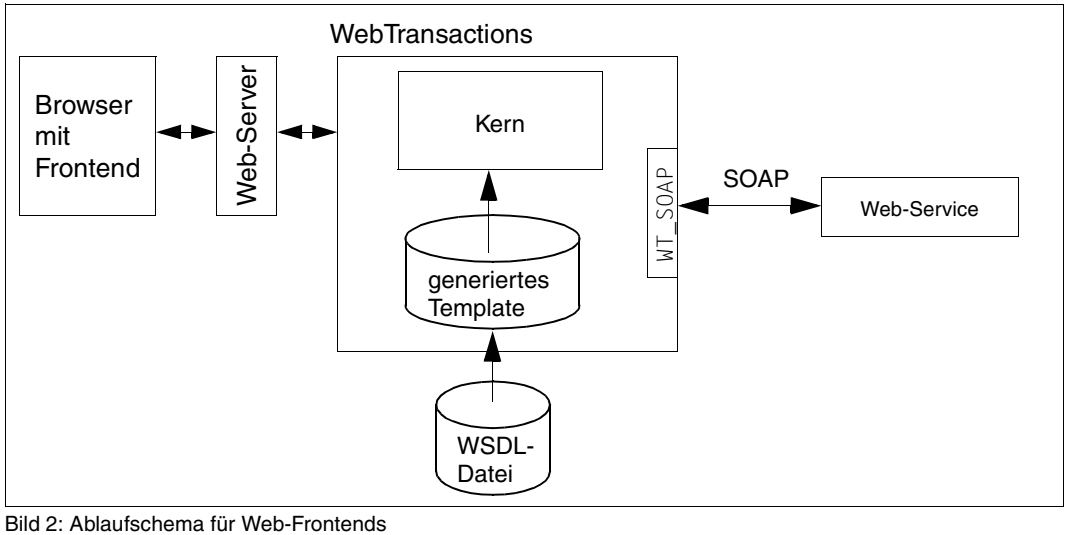


Bild 2: Ablaufschema für Web-Frontends

Aus dem Template erzeugt WebTransactions zur Laufzeit die HTML-Seite für den Browser und eine Instanz der Klasse WT_SOAP für Web-Services. Über diese Instanzen kommuniziert WebTransactions mit der Komponente. Die Kommunikation können Sie beeinflussen und das Template nachbearbeiten.

Um eine Komponente an WebTransactions über eine Basisoberfläche anzuschließen, gehen Sie wie folgt vor:

- Generieren Sie mit WebLab ein Basisverzeichnis.
- Erstellen oder besorgen Sie die WSDL-Datei.
- Generieren Sie das Template für die Basis-Oberfläche.
- Lassen Sie das Template ablaufen.
- Bearbeiten Sie das Template nach Ihren Wünschen.

Die einzelnen Schritte sind in den folgenden Kapiteln beschrieben.

3 Basisverzeichnis erstellen

Nach der Installation von WebTransactions auf dem WebTransactions-Server und von WebLab auf Ihrem persönlichen Windows-Rechner können Sie mit Hilfe von WebLab ein oder mehrere Basisverzeichnisse erzeugen. Ein Basisverzeichnis nimmt alle Dateien auf, die WebTransactions für ein bestimmtes Anwendungsszenario konfigurieren.

Bei einer De-Installation von WebTransactions oder beim Installieren einer neuen Produktversion bleiben die individuellen Konfigurationen also erhalten.

3.1 Basisverzeichnis anlegen mit WebLab

Damit Sie ein Basisverzeichnis für eine WebTransactions-Anwendung anlegen können, muss der WebTransactions-Administrator zuvor eine Benutzerkennung für Sie einrichten. Anschließend muss er für diese Benutzerkennung ein oder mehrere Pools freigeben, damit Sie dort ein Basisverzeichnis anlegen können.

Bevor Sie ein Basisverzeichnis erzeugen, empfiehlt es sich außerdem, zunächst ein Projekt zu erstellen, in dem die wichtigsten Daten gespeichert werden, die WebLab beim Arbeiten mit der WebTransactions-Anwendung benötigt. Beim Anlegen des Projekts wird Ihnen dann automatisch die Option angeboten, ein Basisverzeichnis zu erstellen.

Gehen Sie folgendermaßen vor:

- Rufen Sie WebLab auf, z.B. über **Start/Programme/WebTransactions 7.5/WebLab**

- Als Einstieg zum Anlegen eines Basisverzeichnisses gibt es folgende zwei Möglichkeiten:

  - Wählen Sie den Befehl **Projekt/Neu...** und bestätigen Sie die Abfrage, ob ein Basisverzeichnis erstellt werden soll, mit **Ja**.

  oder

  - Wählen Sie den Befehl **Generieren**/**Basisverzeichnis ...** und geben Sie bei der folgenden Abfrage an, dass ein neues Projekt erstellt wird.

  In beiden Fällen wird das Dialogfeld **Verbinden** geöffnet.

- Tragen Sie im Dialogfeld **Verbinden** die Verbindungsparameter ein und bestätigen Sie mit **OK**.
- Geben Sie im nachfolgenden Dialogfeld Ihre Benutzerkennung mit Passwort an und klicken Sie auf **OK**.
- Machen Sie im Dialogfeld **Basisverzeichnis erstellen** folgende Einträge:
  – Wählen Sie unter den angebotenen Pools den Pool aus, in dem das Basisverzeichnis angelegt werden soll.
  – Tragen Sie den Namen des neuen Basisverzeichnisses ein.
  – Klicken Sie auf **OK**.

  Struktur und Inhalt des Basisverzeichnisses sind im WebTransactions-Handbuch „Konzepte und Funktionen“ sowie im Abschnitt „Struktur eines Basisverzeichnisses“ auf Seite 16 beschrieben.

Basisverzeichnis auf eine neue Version umstellen

- Wählen Sie **Generieren/Basisverzeichnis aktualisieren**. Das Dialogfeld **Basisverzeichnis aktualisieren** wird geöffnet.
- Wenn nur die Links aus dem Basisverzeichnis in das neue Installationsverzeichnis geändert werden sollen, wählen Sie die Option **Verweise anpassen**. Wählen Sie diese Option, wenn Sie Dateien angepasst haben, die von WebTransactions mitgeliefert oder generiert wurden.
- Wenn alle Dateien, die beim Erzeugen eines Basisverzeichnisses kopiert oder generiert werden, neu erstellt werden sollen, wählen Sie die Option **Dateien überschreiben**.

3.2 Struktur eines Basisverzeichnisses

Dieser Abschnitt beschreibt ausschließlich die für das Web-Frontend für Web-Services spezifischen Besonderheiten des Basisverzeichnisses.

Allgemeingültige Informationen zur Struktur von Basisverzeichnissen finden Sie im WebTransactions-Handbuch „Konzepte und Funktionen“.

Unterverzeichnis wsdl

Dieses Unterverzeichnis wird nicht bei der Generierung des Basisverzeichnisses angelegt, sondern erst dann, wenn Sie Web-Services anbinden. In `wsdl` werden die WSDL-Dateien der genutzten Web-Services gespeichert, die WebTransactions für den Ablauf braucht.

4 Template generieren

Nachdem Sie ein Basisverzeichnis für Web-Services generiert haben oder mit einem solchen Basisverzeichnis verbunden sind, können Sie in WebLab mit dem Befehl **Generieren/Templates/für WebServices ...** die Generierung eines Templates starten.

Damit können Sie Templates für den Anschluss von allgemeinen Web-Services an WebTransactions generieren.

Um die Templates für den Anschluss einer solchen Komponente zu generieren, brauchen Sie

– eine Beschreibungsdatei (WSDL-Datei)

– das entsprechende Master-Template

WSDL-Datei

Pro Generierungslauf können Sie eine oder auch mehrere Beschreibungsdateien, so genannte WSDL-Dateien (**W**eb**S**ervice **D**escription **L**anguage) angeben. Dabei ist jede Komponente in genau einer WSDL-Datei beschrieben. Wenn Sie mehrere WSDL-Dateien angeben, werden auch mehrere Templates für mehrere Komponenten generiert. Die WSDL-Dateien können sowohl lokal auf Ihrem Rechner liegen als auch innerhalb des Basisverzeichnisses.

Ein Web-Service ist in der Regel schon in WSDL beschrieben, so dass Sie die WSDL-Datei im Web beim jeweiligen Web-Service finden.

Master-Template

Für jede WSDL-Datei wird ein Template generiert. Dabei dient das Master-Template `WSDL.wmt` als Schablone.Dieses Master-Template liegt im Installationsverzeichnis von WebLab im Unterverzeichnis `weblab`. Sie können das Master-Template Ihren Anforderungen anpassen. Die entsprechenden Master-Template-Tags sind im WebTransactions-Handbuch „Template-Sprache“ beschrieben.

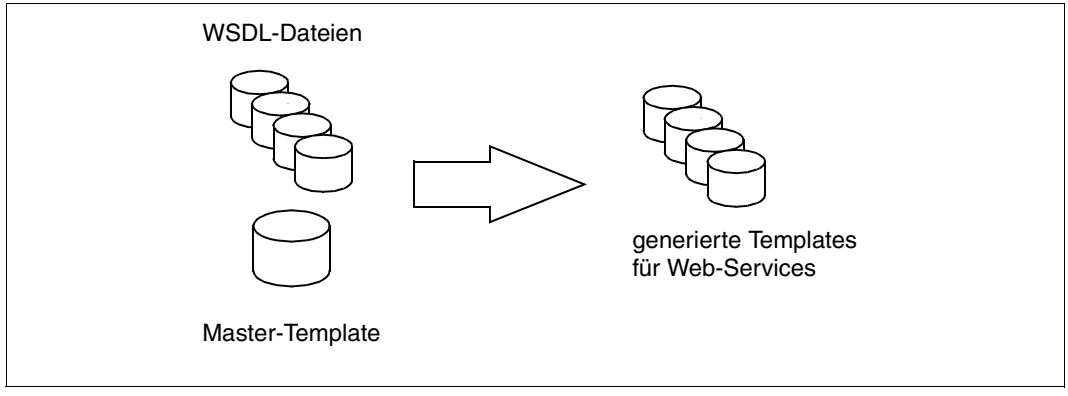


Bild 3: Template für Web-Services generieren

Die Templates, die Sie aus WSDL-Datei und Master-Template für allgemeine Web-Services generiert haben, haben folgenden vordefinierten Namen:

`wtWebService_`*komponentenname*`.htm`

Der *komponentenname* orientiert sich dabei am Namen der WSDL-Datei. Wenn Sie nichts anderes angeben, werden die generierten Templates im Basisverzeichnis unter `config/forms` abgelegt. Ein anderes Zielverzeichnis muss immer unterhalb von *basedir*`/config` liegen.

Die generierten Templates können Sie unverändert einsetzen oder als Basis für anspruchsvolle optische Aufbereitung verwenden.

Template für allgemeinen Web-Service via SOAP-Anschluss generieren

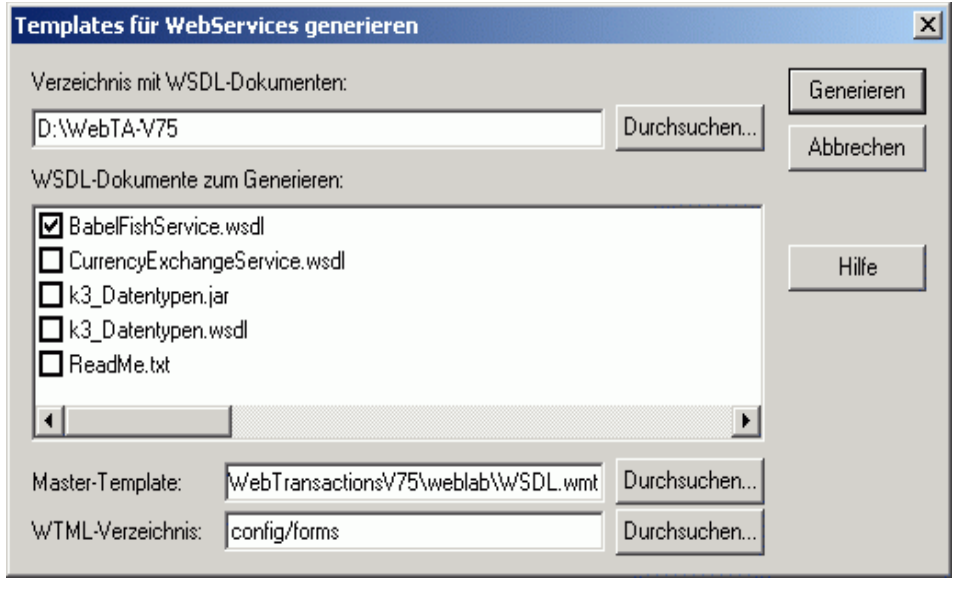
Bevor Sie ein Template generieren können, müssen folgende Dateien vorhanden sein:

– die WSDL-Datei

– das Master-Template `WSDL.wmt`

Um ein Template für Web-Services zu generieren, gehen Sie folgendermaßen vor:

► Wählen Sie in WebLab den Befehl **Generieren/Templates/für WebServices.** Das Dialogfeld **Templates für WebServices generieren** wird eingeblendet.



► Geben Sie das Verzeichnis mit den WSDL-Dateien ein, aus denen Sie Templates generieren wollen oder suchen Sie das Verzeichnis mit der Schaltfläche **Durchsuchen**. Sobald Sie das Verzeichnis ausgewählt haben, werden alle Dateien im Verzeichnis in der Liste **WSDL-Dokumente zum Generieren** angezeigt.

► Wählen Sie aus der Liste **WSDL-Dokumente zum Generieren** die WSDL-Dateien, aus denen Templates generiert werden sollen.

► Prüfen Sie, ob das richtige Master-Template (`WSDL.wmt`) eingestellt ist.

- Geben Sie bei **WTML-Verzeichnis** das Verzeichnis unterhalb des Basisverzeichnisses ein, in dem das generierte Template abgelegt werden soll.
- Klicken Sie auf **Generieren**. Aus Ihren Angaben wird das Template für den Anschluss des Web-Service generiert. Dabei wird das Unterverzeichnis `wsdl` im Basisverzeichnis angelegt und die angegebene WSDL-Datei in dieses Unterverzeichnis kopiert.

5 Web-Frontend testen

Nachdem Sie das Template für den Anschluss eines Web-Service generiert haben, können Sie die Basis-Oberfläche testen. Das generierte Template ist ohne Anpassung ablauffähig und bilden eine modifizierbare Basisoberfläche.

Die WSDL-Datei für den Web-Service muss im Unterverzeichnis `wsdl` liegen. Wenn sich seit der Generierung des Templates etwas am Web-Service geändert hat, empfiehlt es sich, das Template mit der neuen WSDL-Datei neu zu generieren.

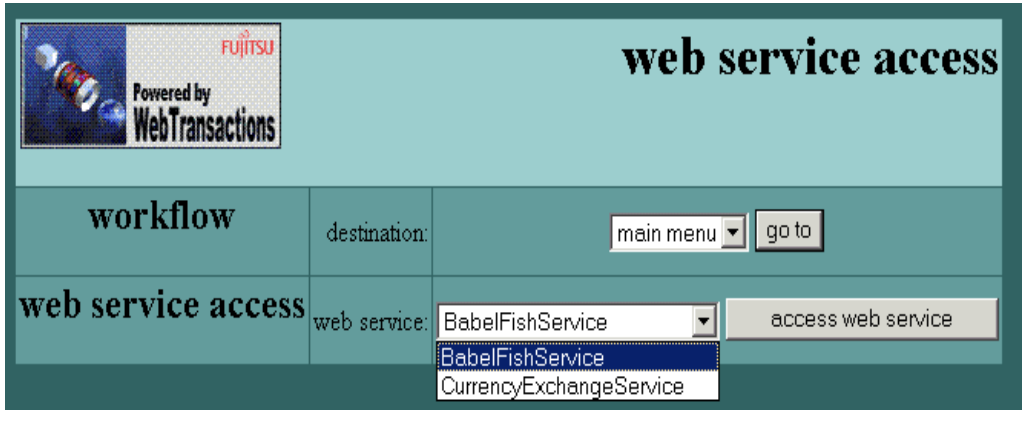
Das Ablaufschema eines Frontends für einen Web-Service finden Sie in Bild „Ablaufschema für Web-Frontends“ auf Seite 13.

Dieses Kapitel beschreibt:
– das allgemeine Start-Template
– die Basis-Oberfläche

5.1 Allgemeines Start-Template

Um das Frontend für einen Web-Service zu starten, gehen Sie folgendermaßen vor:

- ► Starten Sie in WebLab eine Sitzung mit dem Befehl **Datei/Sitzung starten**.
- ► Geben Sie im Dialogfeld **Sitzung starten** die erforderlichen Daten zu dem Rechner an, auf dem WebTransactions läuft, sowie das Basisverzeichnis.
- ► Geben Sie als Start-Template `wtstart` ein und bestätigen Sie mit **OK**. Das Start-Template wird im voreingestellten Browser eingeblendet.
- ► Klicken Sie auf den Knopf **WebService**. Das Start-Template für den Web-Service wird im Browser eingeblendet.



In der Liste **web service access** werden alle Templates angezeigt,

- – deren Name mit `wtWebService_` beginnt
- – die in dem Verzeichnis stehen, das die Systemobjekt-Attribute `STYLE` und `LANGUAGE` festlegen

- ► Wählen Sie einen Web-Service und bestätigen Sie mit dem Knopf **access webservice**.

5.2 Basis-Oberfläche für einen Web-Service

Ein Web-Service wird durch ein WebTransactions-Template repräsentiert. Die Oberfläche ist mit Hilfe des WTBean `wtcTabs` gestaltet. Es gibt eine Registerkarte für den Web-Service, die auch nach dem Web-Service benannt ist. Dort können Sie generelle Einstellungen vornehmen, beispielsweise Proxy-Einstellungen für den Anschluss via SOAP. Für jede Methode gibt es eine Registerkarte, die nach der Methode benannt ist und folgende Angaben enthält:

– den Ausgabe- bzw. den Rückgabewert
– die zuletzt eingegeben Parameter
– die vorgeschriebenen Eingabeparameter

Aufgerufen wird eine Methode mit dem Knopf **execute**.

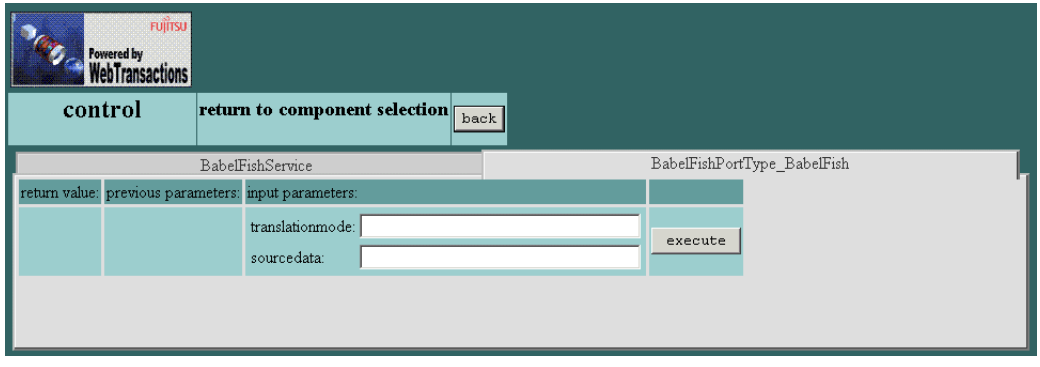


Bild 4: Registerkarte mit Methode des Web-Service BabelFisch

6 Templates bearbeiten

Die generierten Templates können Sie als Template-Programmierer mit WebLab weiterbearbeiten, z.B. um das Layout der generierten Oberfläche ansprechender zu gestalten.

WebTransactions unterstützt Sie bei der Arbeit mit WebLab, insbesondere im Gebrauch von Objekten z.B. zur Parameterübergabe. Zu diesem Zweck erzeugt WebTransactions für jede verwendete Struktur ein Objekt. Da diese Objekte nach einem Template-Durchlauf existieren und somit auch im Objektbaum von WebLab sichtbar sind, können Sie diese mit WebLab-Mitteln in den Code aufnehmen.

Dieses Kapitel informiert über folgende Themen:

– Zugriff auf Objekte
– grafische Aufbereitung der generierten Oberflächen
– Master-Template

Am Ende des Kapitels finden Sie eine kommentierte Auflistung des generierten Templates.

6.1 Zugriff auf die Objekte

6.1.1 Anschluss via SOAP

Für den Anschluss via SOAP wird bei der Generierung der Basisoberfläche jeder Web-Service auf ein WT_SOAP-Objekt abgebildet.

Klasse WT_SOAP

Die Klasse WT_SOAP ist im Template wtSOAP.htm implementiert und ermöglicht einen Zugriff auf Web-Services über das SOAP-Protokoll.

Eine Instanz dieser Klasse repräsentiert einen Web-Service, der in einer WSDL-Datei beschrieben sein muss. WT_SOAP analysiert die WSDL und stellt die Operationen eines Web-Services als Methoden (Proxy-Methoden) der Klasse WT_SOAP zur Verfügung.

Die ausführliche Beschreibung der Klasse WT_SOAP und der zugehörigen Methoden finden Sie im WebTransactions-Handbuch „Zugriff auf dynamische Web-Inhalte“.

Repräsentation eines Web-Service durch ein WT_SOAP-Objekt

Ein Objekt der Klasse WT_SOAP repräsentiert einen Web-Service. Für jeden Web-Service, wird eine Instanz der Klasse WT_SOAP erzeugt. Diese Instanz erhält den Namen WSDL_*laufende_Nummer*.

Aufruf der Operationen eines Web-Service

An einem WT_SOAP-Objekt können Proxy-Methoden aufgerufen werden. In WebLab können Sie einen Methodenaufruf bequem durch Ziehen mit der Maus aus dem Objektbaum in das aktuelle Template einfügen.

Abbildung von SOAP-Datentypen auf WTScript-Datentypen

Bei der Generierung wird für jede Operation eines Web-Service ein Methodenaufruf mit allen benötigten Parametern generiert. Die Datentypen der Parameter werden wie folgt abgebildet:

SOAP-Datentyp	**WTScript-Datentyp**
string	string
boolean	boolean
float	number
double	number
int	number
short	number
long	number
decimal	number
struct (complexType)	Object
array	Array

6.1.2 Beispiel für einfache Datentypen

Bei der Basisoberfläche wird für jeden Parameter einer Methode ein Eingabefeld sowie der zugehörige Wert generiert. Falls der Parameter von einfachem Datentyp ist, wird die Eingabe des Benutzers direkt an die Methode weitergegeben.

Beispiel einer Basisoberfläche für einen Web-Service

Der folgende Web-Service übersetzt einen Text. Im ersten Parameter wird die Übersetzungsrichtung spezifiziert, im zweiten Parameter wird der Text übergeben, der übersetzt werden soll.

Darstellung der Eingabe

Es werden zwei Eingabefelder erzeugt:

```
<TABLE BGCOLOR="#99CCCC">
            <TR>
              <TD>translationmode:</TD>
              <TD><INPUT TYPE="TEXT" CLASS="BOX"
NAME="translationmode_Translate" SIZE=32></TD>
            </TR>
            <TR>
              <TD>sourcedata:</TD>
              <TD><INPUT TYPE="TEXT" CLASS="BOX"
NAME="sourcedata_Translate" SIZE=32></TD>
            </TR>
          </TABLE>
```

Aufruf des Web-Service

```
WSDL_0.service.Translate.port.TranslatePort.operation.Translate(
WT_POSTED.translationmode_Translate, WT_POSTED.sourcedata_Translate);
```

Für die grafische Aufbereitung der Oberfläche können Sie die Darstellung verändern, indem Sie z.B. eines der Eingabefelder durch eine Auswahlliste ersetzen:

```
<TR>
  <TD>translationmode:</TD>
  <TD>
  <SELECT  NAME="translationmode_Translate">
    <OPTION VALUE="de_en" >German->English</OPTION>
    <OPTION VALUE="en_de" >English->German</OPTION>
  </SELECT>
 </TR>
```

Da die Auswahlliste den gleichen Namen hat wie das Eingabefeld, bleibt der Aufruf der Operation unverändert.

6.1.3 Beispiel für komplexe Datentypen

Wenn Sie komplexe Datenstrukturen in die Oberfläche mit einbeziehen wollen, müssen Sie dafür sorgen, dass der Anwender an der Oberfläche Werte für die komplexe Datenstruktur eingeben kann, da bei der Generierung für jeden komplexen Datentyp nur ein Eingabefeld erzeugt wird. Gehen Sie wie folgt vor:

- Ersetzen Sie an der entsprechenden Stelle im Template das generierte `INPUT`-Tag für die gesamte Struktur durch mehrere `INPUT`-Tags für jedes Attribut dieser Struktur.
- Kopieren Sie nach dem Senden diese geposteten Daten im Receive-Script vor dem Methodenaufruf auf die Objekt-Attribute.

Beispiel einer Basisoberfläche für einen Web-Service via SOAP

```
<wtOnCreateScript>
pO_Object_getTest = new Object(); //Objekt für die komplexe Datenstruktur
</wtOnCreateScript>

<input type="text" name="pO_Object_getTest" value=""/>

<wtOnReceiveScript>

/* Methodenaufruf; als Parameter wird das Objekt für die komplexe
   Datenstruktur übergeben */

...getTest( proxyObjects.getTest_pO, ...);

</wtOnReceiveScript>
```

Das Script müssen Sie wie folgt ergänzen:

- Geben Sie für jedes Attribut ein eigenes Input-Tag an:

```
...
<input type="text" name="p0_Object_getTest_minutes" value=""/>
...
```

- Ergänzen Sie im Receive-Script vor dem Methodenaufruf für jedes Attribut wie folgt (hier am Beispiel des Attributs `minutes` dargestellt):

```
...
proxyObjects.getTest_p0.minutes = WT_POSTED.p0_Object_getTest_minutes;
...
<input type="text" name="p0_Object_getTest" value=""/>
```

So erzeugen Sie das Attribut und weisen ihm den Wert zu, den der Anwender an der Benutzeroberfläche eingegeben hat.

Weitere Ersetzungen, die vorgenommen werden müssen, sind:

```
<input type="text" name="p0_Object_getTest.minutes" value="">
<input type="text" name="p0_Object_getTest.seconds" value="">
<input type="text" name="p0_Object_getTest.hours" value="">
<input type="text" name="p0_Object_getTest.date" value="">
<input type="text" name="p0_Object_getTest.month" value="">
<input type="text" name="p0_Object_getTest.year" value="">
```

6.2 Master-Template

Das Master-Template `WSDL.wmt` ist vorgegeben.

Das vorgegebene Master-Template wird im Verzeichnis *Installationsverzeichnis*`/weblab` installiert.

Folgende neue Master-Template-Tags (MT-Tags), die nur für die Protokolloption `WSDL` gültig sind, stehen zur Verfügung:

MT-Tag	Erläuterung
`%%SOURCE%`	wird ersetzt durch den Namen der kopierten WSDL-Datei im Basisverzeichnis (z.B. `WSDL/BabelFishService.wsdl`).
`%%ObjectCreate%`	löst die Generierung der Objekte für komplexe Strukturen aus.
`%%MethodInterface%`	löst die Generierung der Darstellung und der Weiterverarbeitung der einzelnen Methoden aus.

Folgende MT-Tags dürfen nicht im Master-Template für die Generierung von Web-Frontends für Web-Services verwendet werden:

- `%%Lines...%`
- `%%ReceiveCopies%`

6.3 Generiertes Template

Dieser Abschnitt beschreibt das generierte Template und die Präsentation der zugehörigen Oberfläche für die Unterstützung allgemeiner Web-Services via SOAP-Anschluss.

Der folgende Text zeigt das Template `wtWebService_BabelFishService.htm`, das im Abschnitt „Template für allgemeinen Web-Service via SOAP-Anschluss generieren“ auf Seite 19 generiert wurde.

Generierungsoptionen

```
<html>
<wtrem>**************************************************************************
**</wtrem>
<wtrem>** WTML document: BabelFishService
**</wtrem>
<wtrem>**************************************************************************
**</wtrem>
<wtrem>**
**</wtrem>
<wtrem>** Document generation based on Master Template :
**</wtrem>
<wtrem>**  C:\Programme\WebTransactionsV75\weblab\WSDL.wmt
**</wtrem>
<wtrem>**
**</wtrem>
<wtrem>** Generated at Mon Jan 10 11:06:12 2010
**</wtrem>
<wtrem>**
**</wtrem>
<wtrem>** Options used by the generator :
**</wtrem>
<wtrem>**   -  %OPTIONS:
**</wtrem>
<wtrem>**      CommObj = WSDL_0
**</wtrem>
<wtrem>**      Source = wsdl/BabelFishService.wsdl
**</wtrem>
<wtrem>**************************************************************************
**</wtrem>
<wtrem>** WebTransactions V7.5       Fujitsu Technology Solutions GmbH, 2010
**</wtrem>
<wtrem>**************************************************************************
**</wtrem>
```

Proxyobjekt mit `WT_SOAP` erzeugen

```
<wtOnCreateScript>
<!--
include ('wtSOAP');
if ( typeof WT_SYSTEM.WSDL_0 == 'undefined' )
  WT_SYSTEM.WSDL_0 = new Object();
if ( typeof WT_SYSTEM.WSDL_0["BabelFishService"] == 'undefined' )
  WT_SYSTEM.WSDL_0["BabelFishService"] = new Object();
if ( typeof WT_SYSTEM.WSDL_0["BabelFishService"].instance == 'undefined' )
{
  try
  {
    WSDL_0 = WT_SYSTEM.WSDL_0["BabelFishService"].instance = new
WT_SOAP('wsdl/BabelFishService.wsdl');
    WSDL_0.proxyObjects = proxyObjects;
  }
  catch ( exc )
  {
    document.writeln(exc);
    WT_SYSTEM.WSDL_0["BabelFishService"].ret_value = exc;
  }
}
else {
  WSDL_0 = WT_SYSTEM.WSDL_0["BabelFishService"].instance;
  proxyObjects = WSDL_0.proxyObjects;
}
WSDL_0_system = WT_SYSTEM;

//-->
</wtoncreatescript>

<wtif (WSDL_0_system.PROLOG)>
  <wtinclude Name="##WSDL_0_system.PROLOG#">
</wtif>
```

HTML-Kopf: Format WebTransactions definieren

```
<head>
<title>WebTransactions V7.5 - WebService BabelFishService</title>
##WT_SYSTEM.CGI.HTTP_USER_AGENT.indexOf( 'MSIE' ) >= 0 ?
 '<meta http-equiv="Pragma" content="no-cache"/>' :
 '<meta http-equiv="Cache-Control" content="no-cache"/>'#
<wtif (WT_BROWSER.acceptClass)>
  <style type="text/css">
    input {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
```

```
    }
    input.box {
      border:       0 solid;
      padding:      1px 0 1px 0;
      margin-left:  -1px;
      margin-top:   ##WT_BROWSER.marginTop#px;
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      color:        #000000;
      background-color: #FFFFFF;
    }
    input.button {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      border-width: 1pt;
      margin-left:  -1pt;
    }
    select {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
    }
    pre {
      font-size:    ##WT_BROWSER.charSize#px;
      font-family:  courier new, monospace;
      margin:       0;
    }
  </style>
</wtif>
</head>
  <body bgcolor="#336666" text="#000000">
   <form WebTransactions name="BabelFishService">
    <table cellspacing="1" cellpadding="2">
      <tr>
        <td colspan="3" align="right">
          <img src="##WT_SYSTEM.CGI.SCRIPT_NAME#/logo" align="left"/>
        </td>
      </tr>
      <tr>
        <td>
          <wtif (WSDL_0_system.FORMTPL)>
            <wtinclude Name="##WSDL_0_system.FORMTPL#">
          </wtif>
        </td>
      </tr>
```

Knopf zur generellen Steuerung

```
     <tr bgcolor="#99CCCC">
        <td colspan="3" align="center">
          <h2>control</h2>
        </td>
        <td align="center"><h3>return to component selection</h3>
        </td>
        <td align="center">
          <input type="submit" name="Control" value="back"/>
        </td>
      </tr>
    </table>
```

Generelle Steuerung

```
    <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  -->
    <!-- main template control section
-->
    <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  -->
    <wtonreceivescript>
      if (WT_POSTED.Control)
      {
        WT_SYSTEM.FORMAT = WT_SYSTEM.PREVIOUS_CONTROL_FORMAT;
        delete WT_SYSTEM.PREVIOUS_CONTROL_FORMAT;
        exitReceiveProcessing();
      }
      if (WT_POSTED.SetParameter)
      {
        if (WT_POSTED.Proxy)
        {
          if (WT_POSTED.ProxyPort)
                WSDL_0.setProxy(WT_POSTED.Proxy, WT_POSTED.ProxyPort);
            else WSDL_0.setProxy(WT_POSTED.Proxy);
        }
        if (WT_POSTED.ProxyUser)
        {
          if (WT_POSTED.ProxyPwd)
                WSDL_0.setProxyAuthorization(WT_POSTED.ProxyUser,
WT_POSTED.ProxyPwd);
            else WSDL_0.setProxyAuthorization(WT_POSTED.ProxyUser);
        }
      }
    </wtonreceivescript>
```

Kopf für den Web-Service

```
    <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  -->
    <!-- begin of service operations section
-->
    <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  -->
    <wtc name="wtcTabs" type="inline" version="7.5">
<wtcTitle>Tab Control <wtcPar
name="objectname">ComponentTabs</wtcPar></wtcTitle>
<script language="Javascript"
src="##WT_SYSTEM.WWWDOCS_VIRTUAL#/javascript/wtcTabs_browserFunctions.js"
type="text/javascript">
</script>
<script>
<wtcPar name="objectname">ComponentTabs</wtcPar> = new wtcTabs(<wtcPar
name="width">950</wtcPar>,<wtcPar name="tabHeight">20</wtcPar>,'<wtcPar
name="ident">compTabs</wtcPar>',<wtcPar
name="foregroundColor">'#333333'</wtcPar>,<wtcPar
name="backgroundColor">'#DDDDDD'</wtcPar>,<wtcPar
name="borderColor">'#FFFFFF'</wtcPar>,<wtcPar
name="foregroundColorDimmed">'#999999'</wtcPar>,<wtcPar
name="backgroundColorSelectedDimmed">'#CCCCCC'</wtcPar>,<wtcPar
name="borderColorDimmed">'#DDDDDD'</wtcPar>);
</script>
<!------------------------------------>
<!-- begin of list of tabs         -->
<!------------------------------------>
<wtcList name="Tab-List" minLength="1">
<wtcTitle>List of Tabs</wtcTitle>
```

Registerkarten, um den Zugriff über HTTP zu steuern

```
<!----------------------------------->
<!--     begin of list item        -->
<!----------------------------------->
<wtcListItem>
    <script language="JavaScript" type="text/javascript">
<wtcPar name="objectname">ComponentTabs</wtcPar>.addTab(<wtcPar
name="titlename">'BabelFishService'</wtcPar>, <wtcPar
name="titlewidth">150</wtcPar>);
    </script>
<wtcBlock>
<wtcTitle><wtcPar name="titlename">'BabelFishService'</wtcPar></wtcTitle>
    <table cellspacing="3" cellpadding="2">
      <tr bgcolor="#669999">
        <td>
          further parameters:
        </td>
        <td>
          HTTP parameters:
        </td>
      </tr>
      <tr bgcolor="#99CCCC">
        <td>
          <table>
            <tr>
              <td>##WT_POSTED.Proxy || ' '#</td>
              <td> </td>
            </tr>
            <tr>
              <td>##WT_POSTED.ProxyPort || ' '#</td>
              <td> </td>
            </tr>
            <tr>
              <td>##WT_POSTED.ProxyUser || ' '#</td>
              <td> </td>
            </tr>
            <tr>
              <td> </td>
              <td> </td>
            </tr>
          </TABLE>
        </td>
        <td>
          <TABLE>
            <tr>
              <td>Proxy:</td>
              <td><input type="text" name="Proxy" size="32"/></td>
```

```
            </tr>
            <tr>
              <td>Proxy-Port:</td>
              <td><input type="text" name="ProxyPort" size="6"/></td>
            </tr>
            <tr>
              <td>Proxy-User:</td>
              <td><input type="text" name="ProxyUser" size="32"/></td>
            </tr>
            <tr>
              <td>Proxy-Password:</td>
              <td><input type="password" name="ProxyPwd" size="32"/></td>
            </tr>
          </table>
        </td>
        <td>
          <input type="submit" name="SetParameter" value="set parameters"/>
        </td>
      </tr>
    </table>
    </wtcBlock>
    </div>
    </wtcListItem>
    <!------------------------------------>
    <!-- end of list item              -->
    <!------------------------------------>
```

Für jede Methode gibt es folgenden Block, der hier nur einmal stellvertretend dargestellt ist.

```
<!------------------------------------>
<!-- begin of list item            -->
<!------------------------------------>
<wtcListItem>
  <script language="JavaScript" type="text/javascript">
  <wtcPar name="objectname">ComponentTabs</wtcPar>.addTab(<wtcPar
name="titlename">'BabelFishPortType_BabelFish'</wtcPar>, <wtcPar
name="titlewidth">120</wtcPar>);
  </script>
  <wtcBlock>
    <wtcTitle><wtcPar
name="titlename">'BabelFishPortType_BabelFish'</wtcPar></wtcTitle>
```

Benutzeroberfläche

- Anzeige der Rückgabe-Information
- Anzeige der zuletzt angegebenen Parameter
- Erzeugen von Eingabefeldern für die Eingabe-Parameter der Methode
- Knopf zum Auslösen der Methode

```
<table cellspacing="3" cellpadding="2">
  <tr bgcolor="#669999">
    <td>
      return value:
    </td>
    <td>
      previous parameters:
    </td>
    <td>
      input parameters:
    </td>
    <td> </td>
  </tr>
```

Rückgabe-Information anzeigen

```
   <tr bgcolor="#99CCCC">
        <td>
          <table>

<tr><td>##WT_SYSTEM.WSDL_O["BabelFishService"].ret_BabelFishPortType_BabelFis
h#</td>
</tr>
          </table>
        </td>
```

Zuletzt angegebene Parameter anzeigen

```
        <td>
          <table bgcolor="#99CCCC">
            <tr>
              <td>
                ##WT_POSTED.translationmode_BabelFishPortType_BabelFish ||
' '#
              </td>
            </tr>
            <tr>
              <td>
                ##WT_POSTED.sourcedata_BabelFishPortType_BabelFish ||
' '#
              </td>
```

```
            </tr>
          </table>
        </td>
```

Eingabefelder für die Eingabe-Parameter der Methode erzeugen

```
        <td>
          <table bgcolor="#99CCCC">
            <tr>
              <td>translationmode:</td>
              <td><input type="text"
name="translationmode_BabelFishPortType_BabelFish" size="32"/></td>
            </tr>
            <tr>
              <td>sourcedata:</td>
              <td><input type="text"
name="sourcedata_BabelFishPortType_BabelFish" size="32"/></td>
            </tr>
          </table>
        </td>
```

Knopf zum Auslösen der Methode

```
        <td>
          <input type="submit" name="BabelFishPortType_BabelFish"
value="execute"/>
        </td>
      </tr>
    </table>
```

Proxy-Methode mit typgerechten Parametern aufrufen, entsprechende Bereiche werden für alle Methoden generiert

```
    <wtOnReceiveScript>
      if ( WT_POSTED.BabelFishPortType_BabelFish)
    {
      try {
        WT_SYSTEM.WSDL_O["BabelFishService"].ret_BabelFishPortType_BabelFish
= WSDL_O.service.BabelFishService.port.BabelFishPort.operation.BabelFish(
WT_POSTED.translationmode_BabelFishPortType_BabelFish,
WT_POSTED.sourcedata_BabelFishPortType_BabelFish);
        }
        catch ( exc ) {

WT_System.WSDL_O["BabelFishService"].ret_BabelFishPortType_BabelFish = exc;}
        exitReceiveProcessing();
      }
```

```
    </wtonreceivescript>
    </wtcBlock>
    </div>
  </wtcListItem>
<!------------------------------------>
<!-- end of list item               -->
<!------------------------------------>
```

Ende des Templates: Registerkarten aktivieren

```
</wtcList>
<!------------------------------------>
<!-- end of list of tabs            -->
<!------------------------------------>
</span>
<script language="JavaScript" type="text/javascript">
<wtcPar name="objectname">ComponentTabs</wtcPar>.adjust();
<wtcPar name="objectname">ComponentTabs</wtcPar>.redefineFocus();
<wtcPar
name="objectname">ComponentTabs</wtcPar>.activate(##WT_POSTED['wtcTabs_<wtcPa
r name="ident">compTabs</wtcPar>_ActiveTab']||0#);
</script>
</wtc>

    <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  -->
    <!-- end of service operations section
-->
    <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
-  -->

    </form>
  </body>
  <wtif (WSDL_0_system.EPILOG)>
    <wtinclude Name="##WSDL_0_system.EPILOG#">
  </wtif>
</html>
```

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
Konzepte und Funktionen
Einführung

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

Stichwörter