Web Application Builder

Documentation

Release 46C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Web Application Builder 5
Creating an Internet Service 5
Using Mixed Mode 7
Creating HTML Templates 8
Extending HTML Templates 10
Adding MIME Objects 10
Creating Language Resources 12
Publishing a Service 13
Statuses of Web Development Objects 14
Executing a Service 16
Tools Used for Implementation Support 17
Navigation by Double-Clicking 17
Pattern and Wizards 18
Flow Builder 20
User Settings for Internet Services 24
Tutorial: Implementing Web Applications 26
ITS Architecture Components 27
Step 1: Designing an User Interface 28
Step 2: Creating a Service 28
Step 3: Creating HTML Templates 30
Step 4: Defining a Layout 32
Step 5: Implementing the flow logik 33
Step 6: Publishing the Service 34
Step 7: Executing the Web Application 35

Web Application Builder

Purpose

The Web Application Builder allows you to create Web development objects within the ABAP Workbench. Existing R/3 transactions require these objects to allow them to run as Web transactions in a Web Browser You can also use the Web Application Builder as an integrated environment for creating MiniApps.

Integration

The Web Application Builder allows you to create Web development objects within the ABAP Workbench. The Web development objects created in this tool, such as service files, HTML templates, and MIME Objects are stored in the R/3 Repository and are connected to the R/3 Change and Transport System.

Features

- Creating Internet services for existing R/3 transactions.
- Creating and implementing Web applications based on the flow logic.
- Generating the HTML templates for the screens of a transaction. These contain standard HTML and HTML^{Business} statements that map the screen layout.
- Editing the generated HTML templates using HTML and HTML^{Business} to develop them further.
- Including MIME objects (icons, graphics, Java applets, animations, and so on) to improve the layout.
- Creating language-specific texts (languages resources).
- Publishing the services or individual service components on the Internet Transaction Server (ITS).
- Executing the complete Web application from the ABAP Workbench.
- Connection to the Change and Transport System.
- Connection to Version Management.

Constraints

The following functions are not yet available:

- There is no syntax check
- HTML^{Business} and the flow logic are not yet integrated with the Debugger.

Creating an Internet Service

Use

In order for you to log onto the R/3 System and start a Web application, the Internet Transaction Server requires a relevant Internet service.
Each service consists of a set of components:

- Service description: parameters that describe how the service is to be carried out. The most important of these are the logon data and, details of the transaction that is to be carried out.

- HTML templates: You can create an HTML template for each screen of an Easy Web Transaction (EWT), which means that the layout is more flexible.
- Language resources: A language resource contains all the texts that are required to execute a service in a particular language. They ensure that the service is language-independent.
- MIME objects: These can be icons, graphics, Java applets, or sound/video components that you use to extend the user interface in the Web environment.

A concrete instance of a service is defined by a theme. A theme has its own set of HTML templates, language resources, and MIME objects, and gives a service a particular appearance. The actual function of the service remains unchanged.

Prerequisites

If you want to create a service for an R/3 transaction, you should first check its classification and change it if necessary. The default classification of a transaction is *Professional User Transaction*.

Procedure

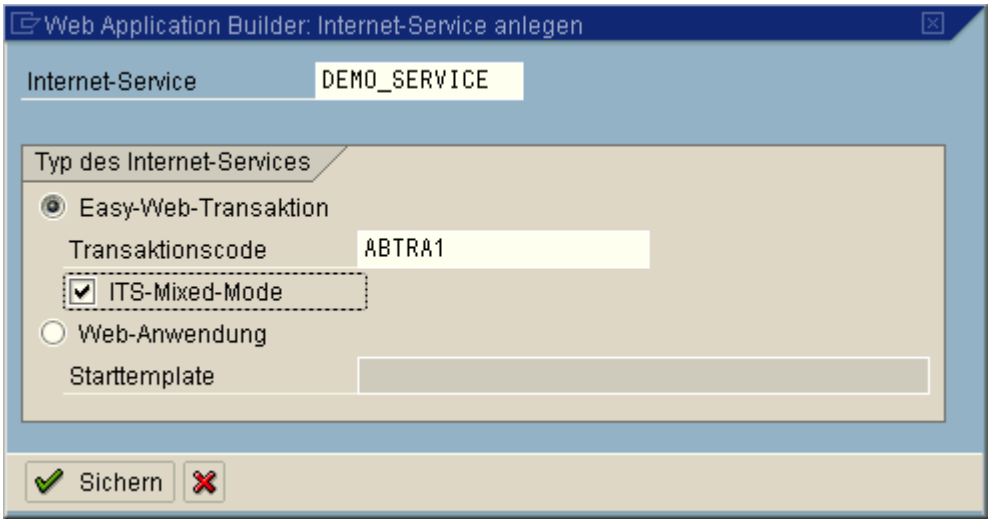
1. Open the *Object Navigator* (transaction: SE80).
2. From the object list, choose *Internet Service*.
3. Enter a name for the service you want to create.

   Note that all Repository objects in the customer namespace begin with Y or Z.
4. Click  or choose *Enter* to confirm your entry.

   The system checks to see whether or not an object with that name already exists. If there is not, the *Create Object* dialog box appears.
5. Choose *Yes* to create the service.

   The *Create Service* dialog box appears.
6. If your Web application is a Web transaction, enter the transaction code of the corresponding R/3 transaction.

   If you only want to generate HTML templates for some of the screens in the R/3 transaction, select the ITS mixed mode option.

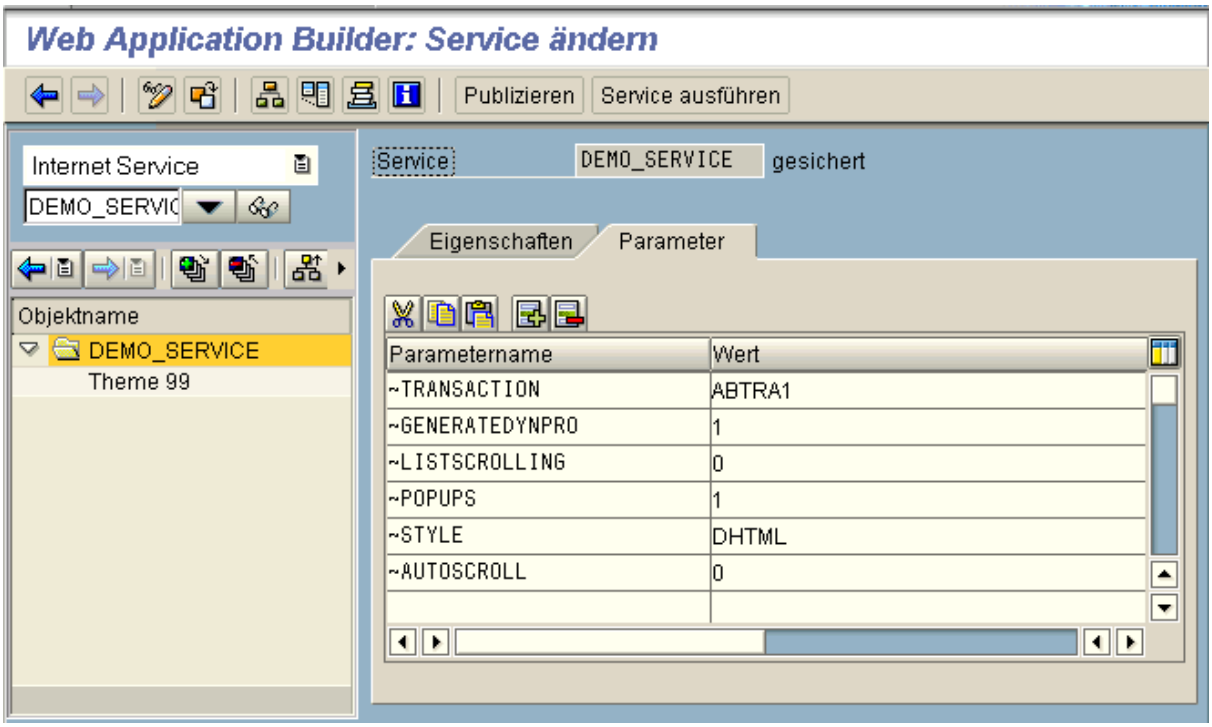
   

7. Choose  .

   The system then displays the *Create object catalog entry* dialog box.
8. Assign a development class to the service.

Result

The Internet service has now been created as a development object in the R/3 Repository, and appears in the object list. The service has been assigned the theme *99*, which is the current theme.
If you entered an R/3 Transaction in step 6, the parameter *~TRANSACTION* will have been generated for the service. If you set the *ITS mixed mode* flag, the corresponding parameters will have been filled with appropriate values.



Using Mixed Mode

Use

Mixed mode allows you to use **both** templates **and** the automatic WebGUI generation within one Web transaction. Screens that do not have templates are generated automatically by the WebGUI.
If a template exists for a screen, the ITS will use it and use HTML^{Business} functions to generate an HTML document for the screen before sending it to the Web browser.

Mixed mode allows you to create templates for selected screens within a transaction. This is particularly useful if particular screens or transactions cannot be reproduced in the WebGUI without errors, or where the layout is inappropriate to your requirements. You can improve the layout of the screens by hand.

Prerequisites

To enable screens to be generated by the WebGUI mechanism, you need to add certain parameters to the Internet Service. The automatic generation is not supported by default for compatibility reasons.

If you did not set the ITS mixed mode flag when you created the service, add the following parameters and values manually. In other cases, they are generated automatically when you create the service and filled with valid values.

Parameter	Value	Description
~generateDynpro	1	Switches on the automatic generation mode for screens that have no template
~listscrolling	0	Simulation of downward scrolling in list reports
~popups	1	Displays dialog boxes instead of suppressing them
~style	DHTML	Specifies which generator should be used
~autoscroll	0	Simulates downward scrolling for step loops and table controls

Creating HTML Templates

Use

When you implement a MiniApp, you must create HTML templates. The dialog logic of a MiniApp runs on the ITS, not in R/3.

For each transaction, you can choose whether you want to generated HTML templates for all screens, for some screens (Mixed Mode), or at all. Templates that you create explicitly are identical to the HTML documents that are generated automatically by the SAP GUI for HTML.

Generating templates explicitly is useful if the SAP GUI features are insufficient for your needs and you would need to adapt the standard generated template anyway. This will particularly be the case if you are trying to improve the layout of a screen or if you want to include hyperlinks.

Standard template generation from the SAP GUI should be sufficient for most transactions. The SAP GUI for HTML can display the screen elements of a simple transaction (text fields, input/output fields, checkboxes, radio buttons, tab controls, table controls, subscreens, and so on) without you having to go to the effort of creating a template.

Prerequisites

- You must already have created the service.
- You have sufficient knowledge of HTML and HTML^{Business} to take advantage of the template-based approach.

Procedure

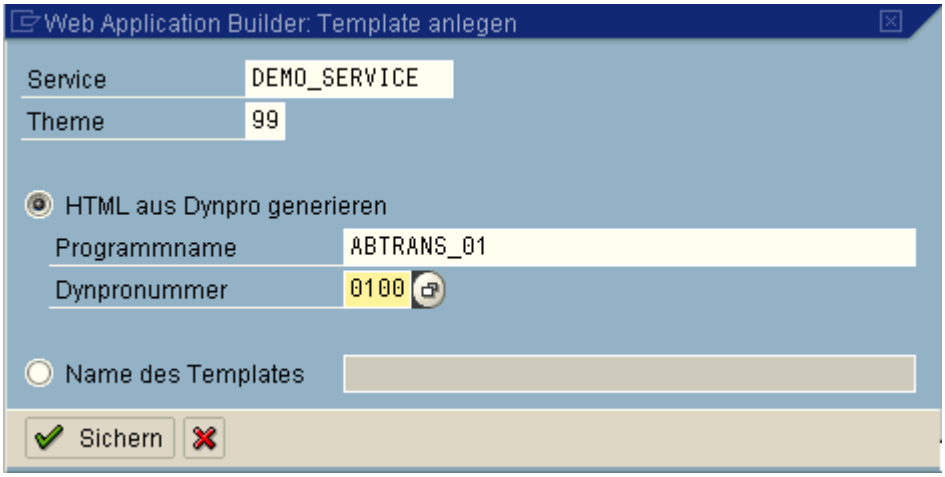
To create an HTML template from the tree display in the object list:

1. Right-click the name of the service.
2. From the context menu, choose *Create* → *Template*.

   The system displays the *Create Template* dialog box.
3. Enter the theme for the service and fill out the remaining fields.

   If the Web application is a Web transaction and you want to generate a template for a particular screen, select *Generate HTML from screen* and enter the program name and screen number.

If the application has no corresponding R/3 screen (MiniApps), select *Name of template* and enter the name.



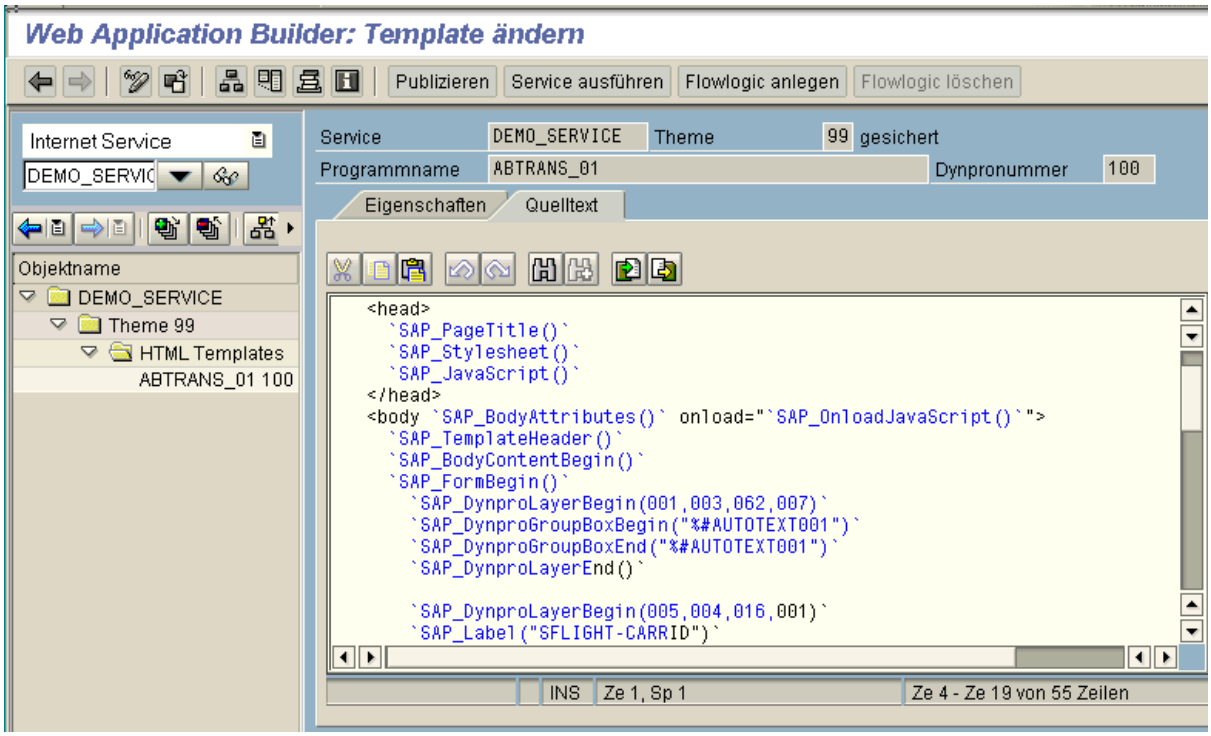
4. Choose *Save*.

   The system then displays the *Create object catalog entry* dialog box.

5. Assign a development class to the template and choose .

Result

The generated template appears in the object list under *Templates*. The generated contents of the template are displayed in the Editor. Only the static screen information is evaluated - an HTML^{Business} function is inserted in the template for each screen element. These are highlighted in blue. You can now change the contents of the template using standard HTML and HTML^{Business}.



Extending HTML Templates

Once you have created an HTML template, you can change the generated source code.

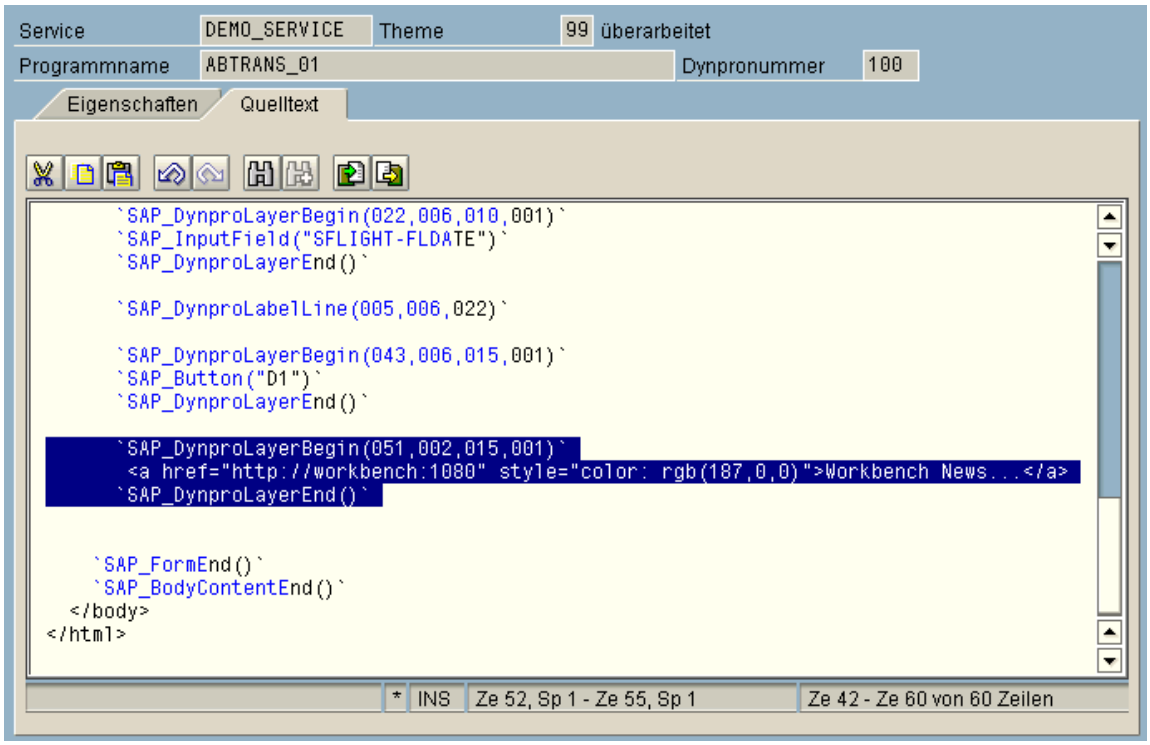
To do this, you must be familiar with the basics of HTML and HTML[Business].

HTML[Business] is an extension of standard HTML developed by SAP to allow R/3 screen data to be merged dynamically with information on HTML templates and to make it easier for the ITS to exchange data between the R/3 System and the Web Server.

For more information, refer to HTMLBusiness Reference

Example

This example sets a hyperlink to a particular position on an HTML page:



Adding MIME Objects

Use

You can use MIME objects (icons, graphics, audio files, animations...) to improve the layout of your Web applications.

Prerequisites

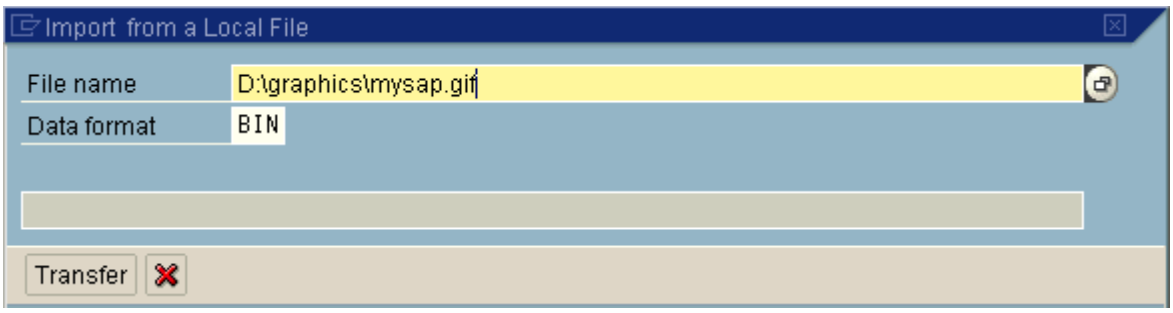
You must already have created an Internet service.

Procedure

To add a MIME object to an Internet service from the object list:

1. Right-click the relevant service.
2. In the context menu, choose *Create* → *Mime*.

   The *Read from Local File* dialog box appears:



3. Enter the path name of the file you want to import, and ensure that the file format is correct.
4. Choose *Import*.

   The *Create Mime* dialog box appears.
5. Enter the theme and the name for the MIME object.
6. In the Name field, you can create a subdirectory, separated from the name of the MIME object by a forward slash ("/").

7. Choose  to continue.

   The *Create Object Catalog Entry* dialog box appears.
8. Assign the MIME object to a development class and choose  .

Result

The MIME object has been inserted in the R/3 Repository as a standalone object. It appears under Mimes in the object list display, and, if it is a graphic, its contents are displayed.

You can now use this object in your interface design.

When you publish the service, the MIME objects are not stored in an ITS directory. Instead, they are stored on the HTTP server under the name and subdirectory you specified in step 6 above.

Creating Language Resources

Use

A language resource contains all of the language-specific elements of an Internet service and enables you to make your application multilingual. Compared with hard-coded text in the HTML template, it also makes it easier for you to understand and maintain your source code.

For each language-specific text in your Web interface, you insert a placeholder into the HTML template (similarly to text elements in an ABAP program). The actual texts are maintained through the theme parameters with the same name. At runtime, the ITS recognizes the placeholders for each template and replaces them with texts in the appropriate language.

Prerequisites

You must already have created the HTML templates for your Internet service.

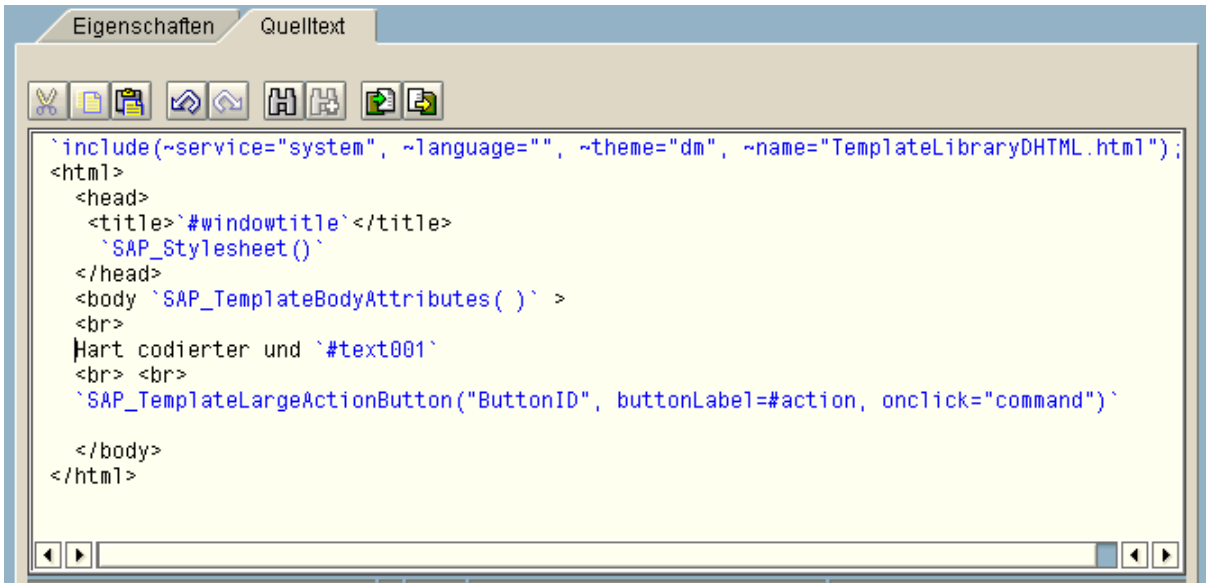
Procedure

Adding Placeholders to an HTML Template

1. Open the relevant template.
2. Enter the placeholder for the language-specific texts in the HTML source code.

   Use the following syntax: `` `#Placeholder` ``

   Suppose we defined three placeholders (`` `#windowtitle` ``, `` `#text001` `` and `` `#action` ``) in the source code.



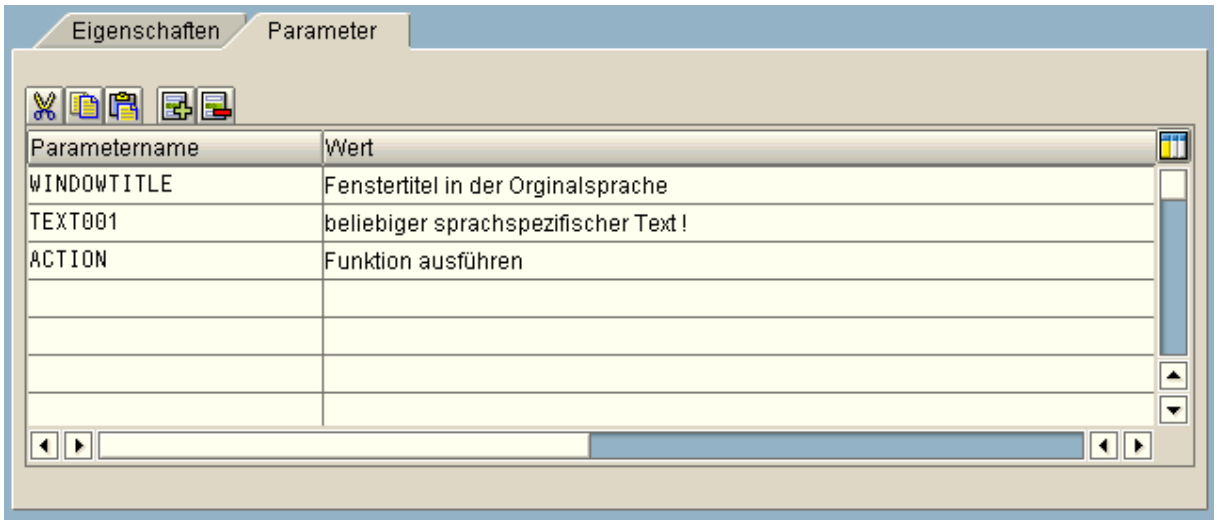
Entering Language-Specific Texts

1. Double-click the theme.

   The system displays the theme parameters.
2. Choose the *Compare parameters* icon.

   Those place-holders for all the templates used by the service that have already been maintained are added to the parameter lit.

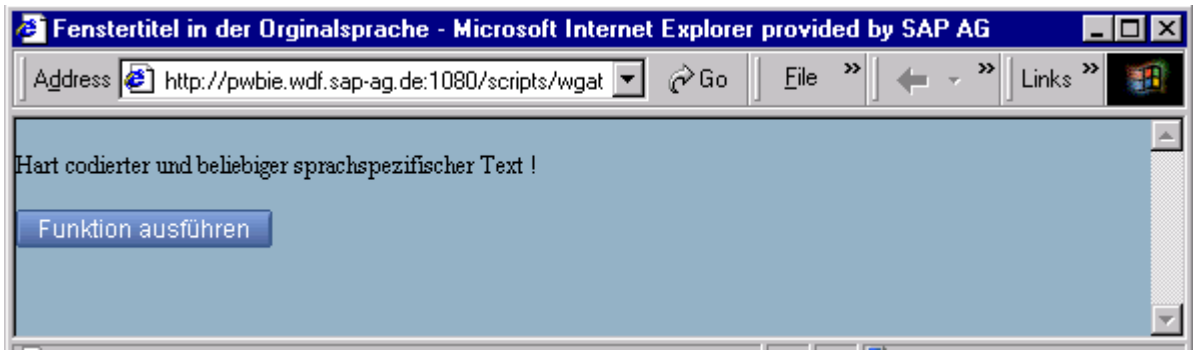
3. Enter the language-specific text in the original language as a value for each parameter.



Result

The theme parameters are now part of the service. They are translation-relevant parts of the R/3 Repository object, and as such will enter the translation workflow when you release the service.

When you start the service in the original language, the texts appear in the relevant language.



If there is no translation of the language-specific texts in the logon language, no text is displayed when the user executes the service.

Publishing a Service

Use

In order for an Internet service to be executed by the ITS, it must be stored in the ITS file system. This is known as publishing the service. You can choose to publish the entire service or just parts of it. When you publish the whole service, the corresponding Internet service and its HTML templates are placed in the file system of the AGate server, and the MIME objects are placed in the file system of the WGate server.

Note that by default, the Internet service is published on all Internet Transaction Servers. However, you can choose to restrict the publication to an ITS assigned

to your particular R/3 System. To do this, choose *Utilities* → *Settings* and then, under *ITS*, enter the required server. For further information, refer to User Settings

Prerequisites

- At least one ITS must have been assigned to the R/3 System, and it must be active.
- In the user settings for the ITS, you have chosen at least one ITS instance, on which you want to publish the service.

Procedure

To publish an entire Internet service from the object list:

9. From the object list, select the appropriate service.
10. From the context menu, choose *Publish* → *Complete service*.

    If an error occurs while the system is publishing the service, a log is generated containing the relevant message texts.

    If no errors occur, the system displays the message *The object has been published successfully*.

    The current status of the object changes after it has been published. For information on how the status of a Web development object is determined, and how you can display its publishing details, see Statuses of Web Development Objects

Result

Once you have published the entire service, you can start your Web application.

Statuses of Web Development Objects

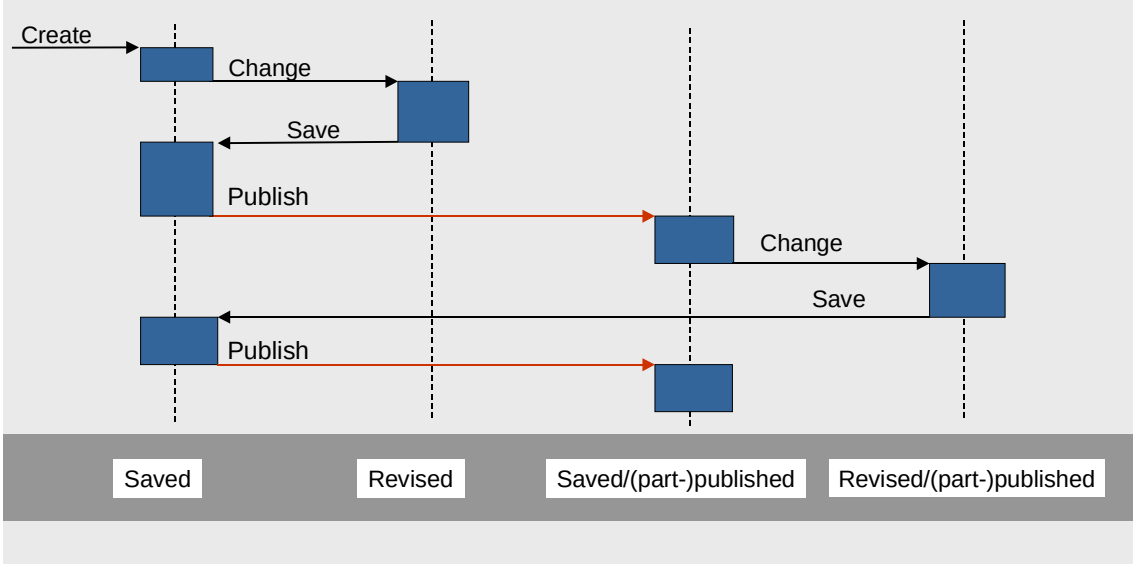
You can see the current status of each Web development object at any time in the Web Application Builder in the **status display** to the right of the object's name. Web development objects can be either an Internet service or one of its components (such as an HTML template, a language resource, or a MIME object).

Object States

The possible status of each Web development object is determined by:

The database status:		
	Saved	This version of the object is identical to that saved in the database.
	Revised	This version of the object is different than the one saved in the database.
Publishing status:		
	Published	The current version of the object has been published on all ITS instances assigned to the system
	Part-published	The current version of the object has been published on at least one ITS instance assigned to the system This version is not, however, available on all ITSs.
	No status displayed	The current version of the object is not available on any ITS instance.

The object status is displayed as follows, depending on what has been done to the object:

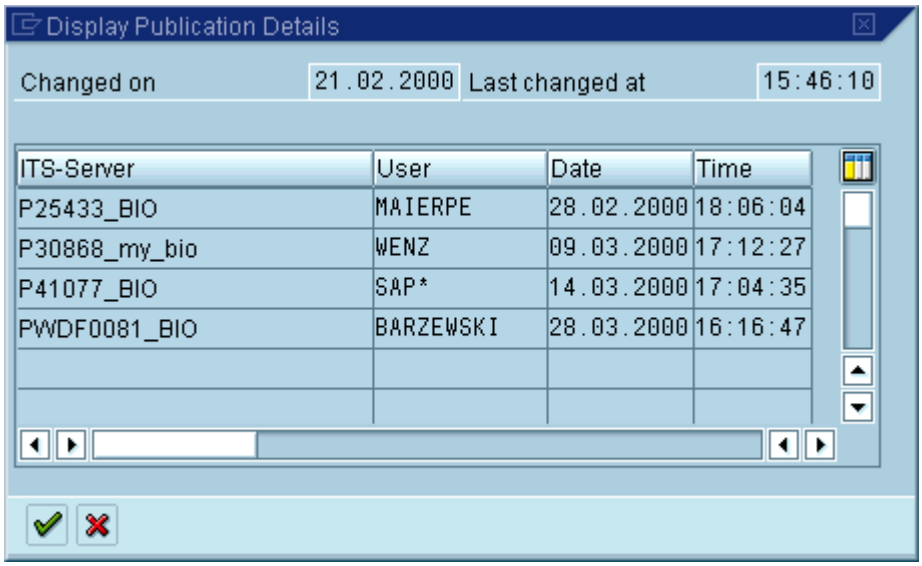


Displaying Other Publishing Status Details

For each Web development object, you can also display: the ITS instance on which the object has been published, when, and by whom.

You can then judge how up-to-date the latest published version of a service or its components is.

To show the publishing details for an object, double-click the **status display**.



Executing a Service

Use

Use this function to test a Web transaction or MiniApp from the ABAP Workbench.

Prerequisites

You must already have published the entire service on the ITS. The ITS must be active.

Procedures

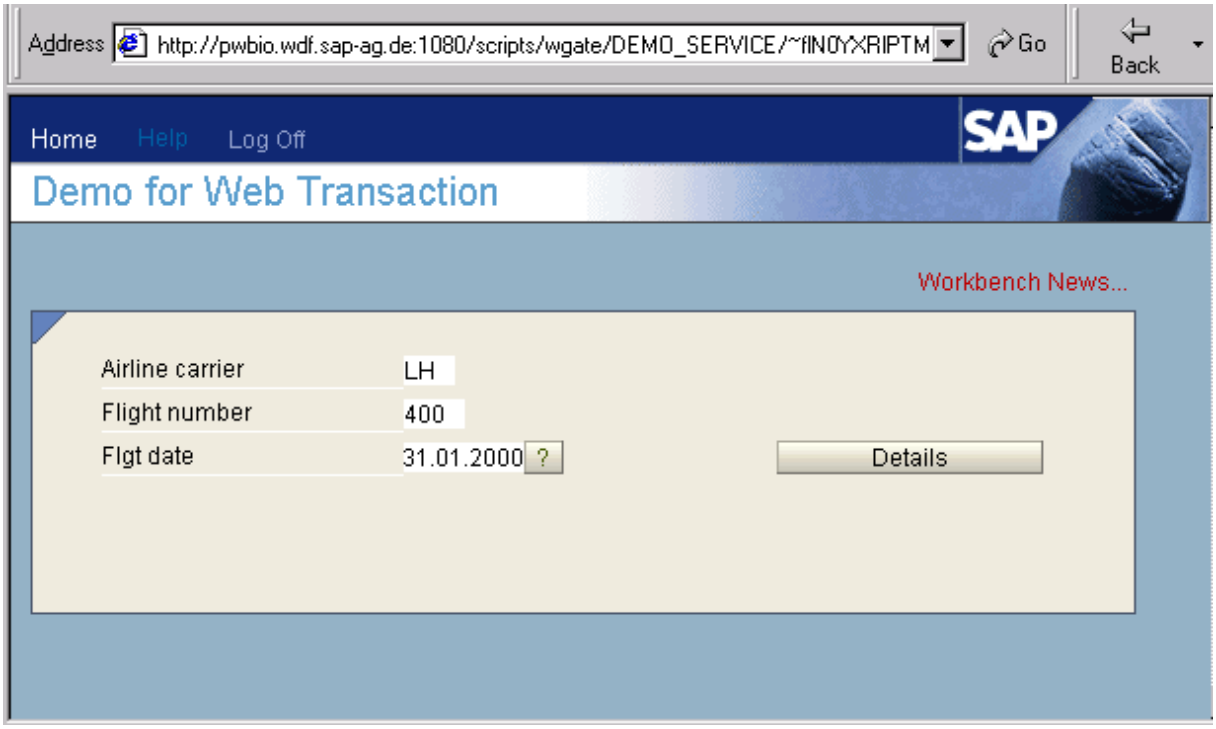
To start the Web application from the Object Navigator:

1. Select the relevant service.
2. Choose *Execute*.

   The system starts the Web browser and displays a logon window.
3. Check the logon language, and log onto the ITS by choosing *Logon*.
4. Run the Web application.

Result

The service is started using the HTTP address
http://<ITS>:<Port>/scripts/wgate/<service>/!



The way in which a Web transaction is displayed depends on the classification of the underlying R/3 transaction. *Professional User Transactions* are displayed with the normal R/3 menus, command field, standard toolbar, and application toolbar in the Web browser. *Easy Web Transactions* (EWTs), on the other hand, are displayed with a special EWT header.

Tools Used for Implementation Support

Overview

Editor	Use	Notes
HTML Editor	Used to edit the source code in the HTML templates for Web applications. Source code generally includes HTML elements, keywords, expressions, statements, and **HTML**[Business] functions. You can also define JavaScript functions.	Use **Patterns** and **Wizards** for HTML[Business] functions to ensure that the source code you are editing is free of errors.
Flow Editor	Used to implement the dialog logic of an HTML template using **flow logic**. Note that you must edit the necessary language elements of the flow logic manually, including its attributes.	You must be familiar with the exact syntax of the flow logic. The Flow Editor does not carry out a syntax check.
Flow Builder	Used to create flow logic step-by-step, without you having to know the exact syntax.	The **correct flow syntax** is entered automatically. You can also use the **check function** to check the dialog logic for inner consistency.

Navigation by Double-Clicking

Navigation in the HTML Editor

By double-clicking, you can navigate from an HTML template to a referenced line in the flow logic; to language resource maintenance; or to another HTML template.

The following table provides details of the navigation options in the HTML Editor:

Double-click	To navigate to
The name of an HTML template	This template in the HTML Editor.
Placeholder for a language resource **(Example: `#action`)**	The theme parameter that defines the language-specific text. (This is equivalent to double-clicking a text element in an ABAP program to navigate to the appropriate maintenance environment).
The name of the event **(Example: `~event="EventName"`)**	The declaration for this event in the Flow Editor.

Navigation in the Flow Editor

The Flow Editor also lets you navigate by double-clicking.

Double-click	To navigate to
The name of the template entered in NEXT_TEMPLATE	The flow logic of the relevant template in the Flow Editor.
The name of the state entered in NEXT_STATE	The point where the relevant state is defined in the flow logic.
The name of the module (either RFC or BAPI)	The function module that is to be defined in the Function Builder.

Patterns and Wizards

Use

Patterns allow you to insert source code patterns for HTML^{Business} functions to an HTML template simply. You can use wizards to create more complex elements, such as tabs, group boxes, or buttons that trigger events.

Procedure

Inserting a pattern

1. In the HTML Editor, make sure you are in *Change* mode and position your cursor where you want to insert the pattern.
2. From the button bar, choose *Business HTML pattern*.

   The *Business HTML pattern* selection dialog box appears.
3. Select a pattern function.
4. Double-click this function to display the description, parameters, and documentation associated with it.
5. Confirm your choice by clicking *Insert.*

   The system inserts the source code needed to call the HTML^{Business} function you have chosen. Note that the function's optional parameters are commented out.

You can also insert a pattern in the template by using **Drag&Drop** in the selection dialog box.

Using Wizards

1. From the button bar, choose *Business HTML pattern*.

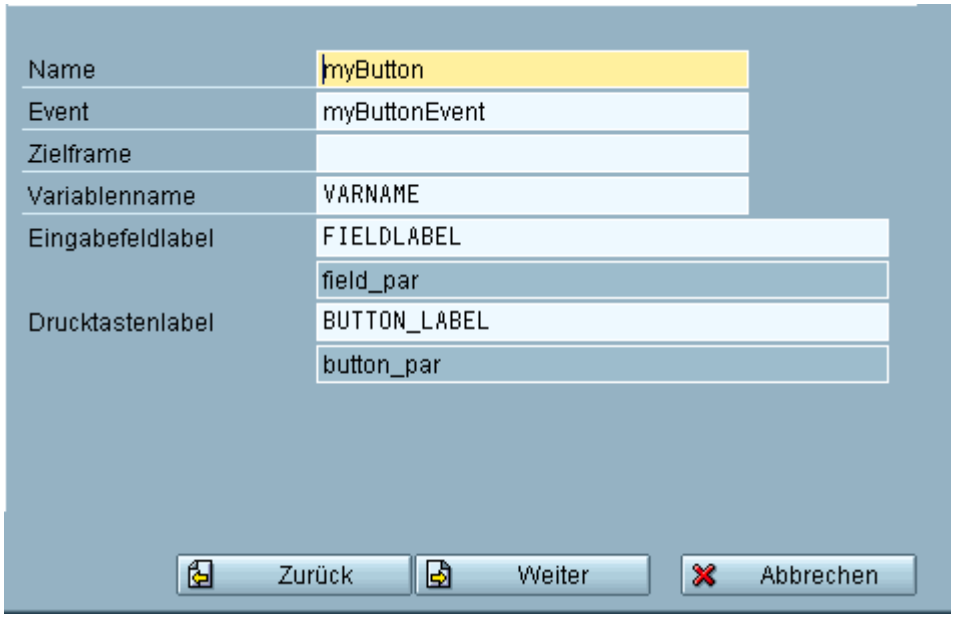
   The *Business HTML pattern* selection dialog box appears.
2. Choose the wizard you want to use and insert it into the HTML template using Drag&Drop.

   The system opens this wizard in a new dialog box.

The wizard guides you through the complete process until the element you are maintaining is ready for use

3. Make the required entries.

The following example indicates the entries you should make to generate a button that triggers a flow event.



4. Choose *Complete* to trigger generation of the source code.

The system inserts this source code for the HTML[Business] functions or other functions (JavaScript functions) into the HTML template.

The above example inserts the following source code:

```
<script language="JavaScript">
function myButton_onClick() {
  document.myButton_form.elements['~event'].value = 'myButtonEvent';
  document.myButton_form.submit();
}
</script>

<form name="myButton_form"
    method="post">
  <input type="hidden" name="~event" value="">
  `SAP_TemplateEditableField("myButton_INPUTFIELD"
              ,fieldLabel=#FIELDLABEL
              <!--,fieldLabelWidth=""-->
              <!--,type="SAP_WEBGUI"-->
              ,name="VARNAME"
              ,value=VARNAME
```

```
                <!--,size=20-->
                <!--,maxlength=20-->
                <!--,inspectionText=""-->
                <!--,align=""-->
                <!--,required=""-->
                <!--,width=""-->
                <!--,onchange=""-->
                <!--,accesskey=""-->
                <!--,tabindex=""-->
                <!--,title=""-->)`
  `SAP_TemplateUpstateButton("myButton_BUTTON"
                <!--,type="SAP_WEBGUI"-->
                ,buttonLabel=#BUTTON_LABEL
                ,onclick="myButton_onClick()"
                <!--,quickinfo=""-->
                <!--,disableOnClick=""-->
                <!--,enableDownState=""-->
                <!--,state="enabled"-->
                <!--,width=""-->)`
</form>
```

Flow Builder

Use

To implement the flow logic for an HTML template in the Web Application Builder, you can use either the **Flow Editor** or the **Flow Builder**. In the Flow Editor, you must edit the necessary language elements of the flow logic manually, including its attributes. Conversely, the Flow Builder allows you to create the flow logic step-by-step, without you having to know the exact syntax. The system generates an initial representation, from which you create a flow logic – that is, you define a flow of states and events. The Flow Builder supports you by providing the input help and navigation functions you need.

Benefits of using the Flow Builder

- You do not need to be familiar with the flow logic syntax.
- The system provides input help when you enter modules, events, subsequent states, and subsequent templates.
- If you use BAPIs r function modules, the system provides default values for the RFC parameters.
- A wide range of navigation options is available:
  - Tree display ↔ double-click ↔ maintenance in the Flow Builder ↔ Function Builder
  - Tree display ↔ context menu ↔ maintenance in the Flow Builder ↔ Function Builder
- The correct flow syntax is entered automatically.

- Using the check function, you can guarantee that the flow logic is internally consistent and semantically correct. (For example, you can check to see whether the subsequent state or template is available in the system).

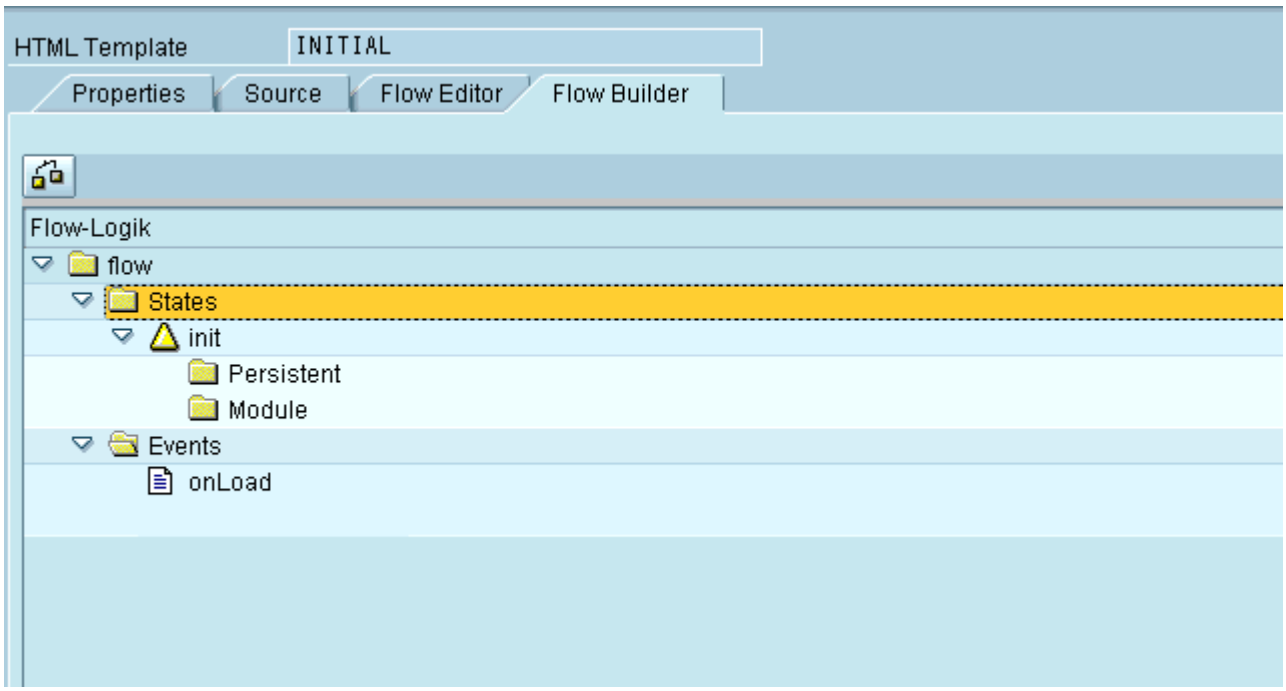
Using the Flow Builder

To define the flow logic:

Call the Flow Builder:

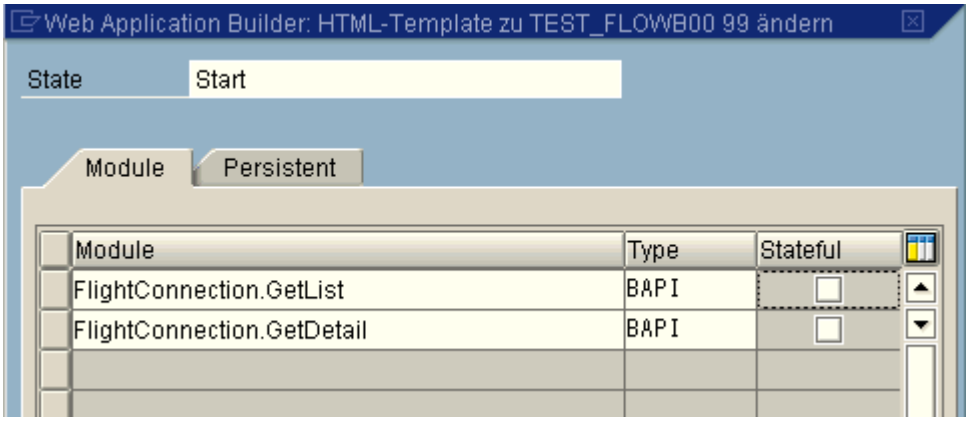
1. In the Object Navigator (transaction SE80), choose *Internet Service*.
2. Select an HTML template.
3. If the dialog logic has **not** been created, Choose *Edit* → *Create flow logic*.
4. Choose the *Flow Builder* tab.

For each new HTML template, the Flow Builder provides an initial representation of the flow logic with the initial state *Start* and the initial event *onLoad*.



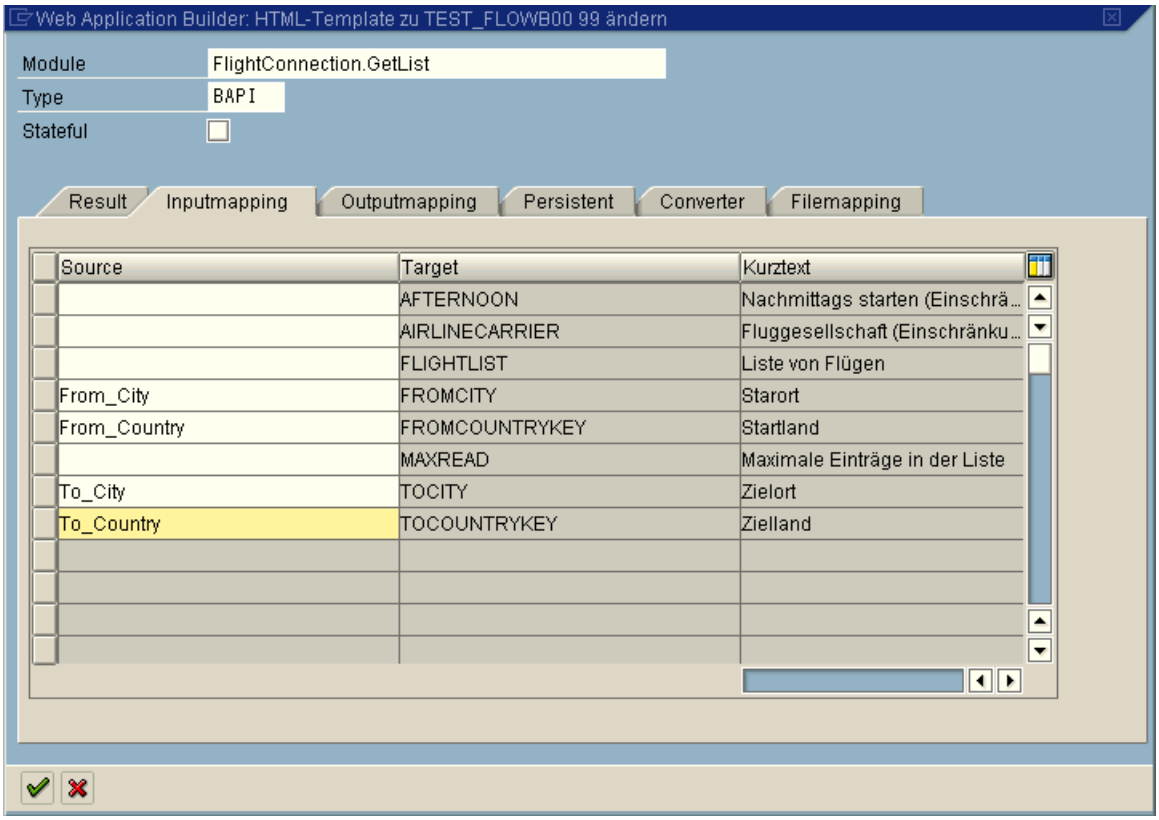
Define the states:

1. In the tree display, double-click the state in which you want to include a module or modules. Alternatively, choose the appropriate function from the context menu.
2. Enter the modules (BAPIs or RFC function modules). Use the possible entries help (F4), entering an asterisk (*) as a place-holder during your search.
3. You can also enter other attributes for the module if you wish.



4. Double-click a module name to modify its specification.
5. If necessary, enter the mapping parameters and other attributes.

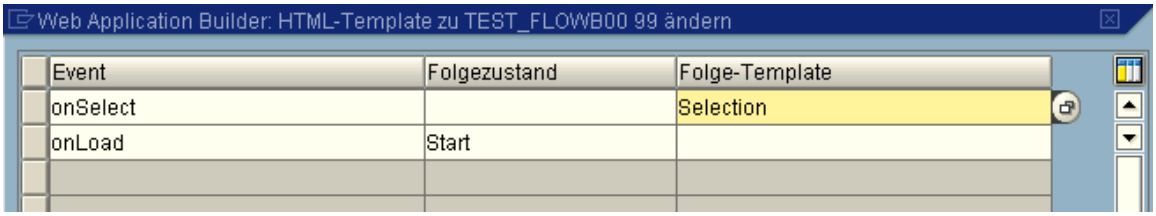
   The RFC parameters are already shown, either as *Target* (input parameters) or *Source* (output parameters). The same applies to exceptions for function modules.



6. Choose *Save* to confirm your entries.

Define the events:

7. To create a new event, double-click the initial event *onLoad* (in the Events node) or choose the appropriate function from the context menu.
8. In the dialog box that appears, choose to insert a line.
9. Enter the name of the state, then either the name of the next state (within the same template) or a subsequent template.



10. Choose ✔ to confirm your entries.

Special features when using the Flow Builder

- Note that output is different in display mode than in change mode:

  In display mode, the system shows only those elements of the module or events that have already been maintained.

- Navigating from the Flow Editor to the Flow Builder
  If you create a flow logic in the Flow Editor, it may contain syntax errors. When you switch to the Flow Builder, these errors will trigger a warning, since the Flow Builder automatically checks the flow logic using the XML parser.

User Settings for Internet Services

Use

You can use the ITS settings to

- Find out which ITS instances are assigned to your SAP System.
- Restrict the ITS instances on which services are published to a single instance.
- Find out the name of the default Web server for starting services.
- Find out the name of the default Web server used to start services.
- Specify a Web server other than the default for starting services.

Activities

To change the ITS settings:

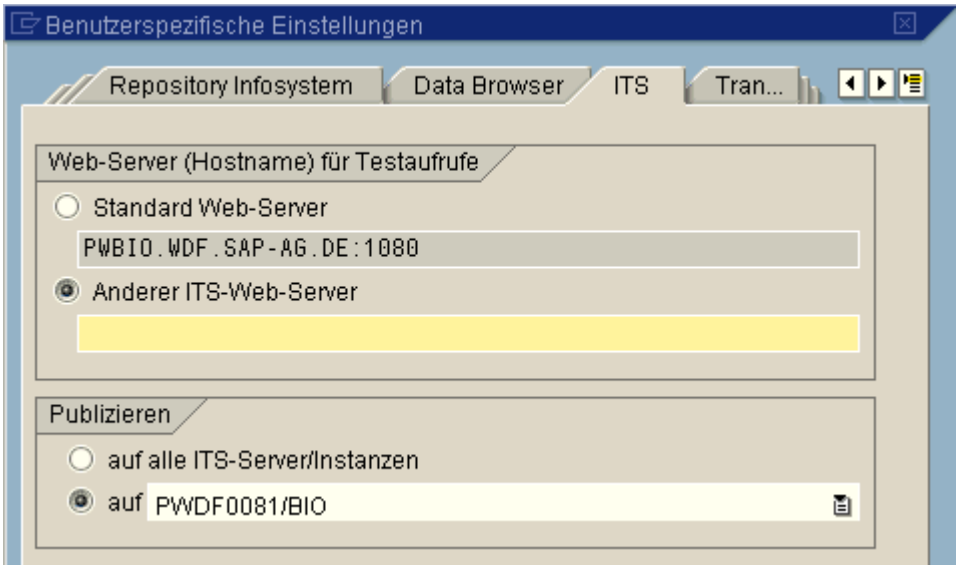
1. Choose *Utilities* → *Settings*.
2. Under *ITS*, change the required filter settings.
3. Choose *Save* to confirm your entries.

The new settings will apply to all subsequent service publications.

Filter settings

Setting		Explanation
Web server (Hostname) *for tests*	*Default web server*	The name of the default web server used to start services in the current SAP System. This name is entered in the Customizing table **TWPURLSVR** (transaction **SM30**). If there is **no entry** for this setting, the name of the web server has not yet been entered. In this case, you should inform your system administrator (unless you have authorization to maintain the table yourself).
	Other ITS web server	You can enter the name of any other web server on which you want to run services. **Note**, however, the assignment of the web server to an SAP System. If you specify a web server belonging to the ITS instance of a different SAP System, the service will be started on a different SAP System to the one from which it was published.

Publish	*On all ITS instances*	The service is published on all ITS instances assigned to the SAP System. The message that follows successful publication is only displayed if the service could be published successfully on all ITS instances. Errors occurring on any instance are logged. Normally, this setting should no longer be used.
	on <individual instances>	From the dropdown box, choose a single ITS on which you want to publish the service. If necessary, choose the default ITS. The name of the ITS instance is derived from the RFC destination details as maintained in transaction **SM30**. If the list box contains no entries, no destinations have been created. To maintain destinations, start transaction **SM30**, enter the view **IACORDES** and create the RFC destination. If you do not have authorization for this, inform your system administrator.



Tutorial: Implementing Web Applications

Task

To create, implement, and execute a simple Web application (in the Web Application Builder), which will display all flights between a given departure and destination airport.
The dialog logic will be implemented using **flow logic**.
The business logic has already been implemented in the *GetList* Business Application Programming Interface (BAPI), in the *FlightConnection* business object.

Overview: Content of the Tutorial

In this tutorial, you will learn how to:

- ✓ Create Internet services for Web applications
- ✓ Create and define simple HTML templates using HTML and HTML^{Business} source text.
- ✓ Implement the dialog logic for the templates using flow logic.
- ✓ Publish and execute the service on the Internet Transaction Server (ITS).
- ✓ Outline which ITS architecture components are used when the system executes a Web application.

To implement a **MiniApp**, the procedure is similar to the one described below, except that you must maintain additional entries in the MiniApp catalog and comply with special design criteria.

Prerequisites

Systems and installed software

- ☐ SAP System, Release 4.6C or higher

  The SAP System should also contain the appropriate business data, for you to test the Web application successfully.
- ☐ The Internet Transaction Server (ITS) is installed and assigned to the SAP System.
- ☐ A standard Web Server is installed and assigned to the ITS, so that you can start Internet services in the current SAP System.

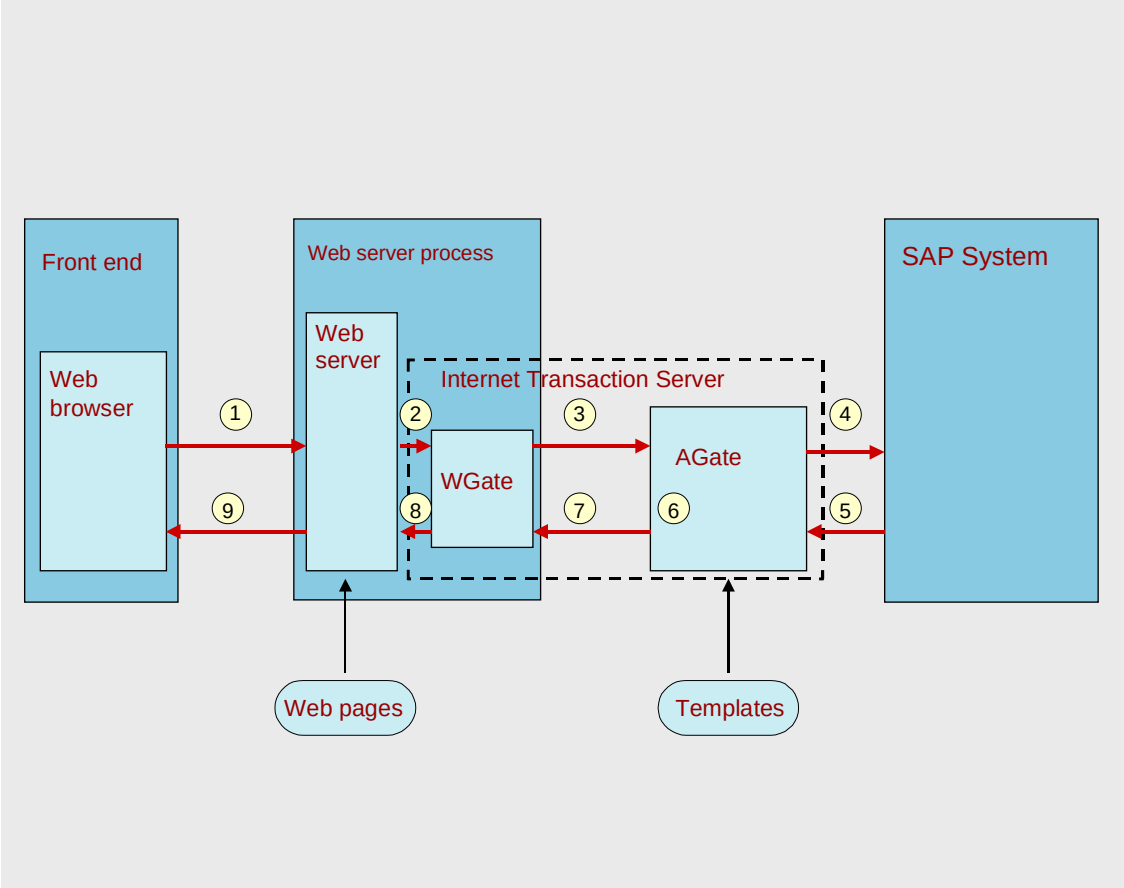
Knowledge and Authorizations

- ☐ Basic knowledge of HTML and HTML^{Business} is recommended.
- ☐ Basic understanding of the flow logic is recommended.
- ☐ Familiarity with the interface definition of the *GetList* BAPI in the *FlightConnection* business object (you can access this using the transaction: BAPI).

ITS Architecture Components

This topic introduces the main components of the ITS environment, which are used when a Web application is executed as a standalone application (rather than integrated into the Workplace architecture). You can use the ITS architecture to trace each step in the process, from the user's entering of data in the input form, to the list displayed in the Web Browser.

ITS Architecture (without Workplace integration)



Comments

The user inputs the data. The Web browser then sends an HTTP request to the Web server (**1**).

The Web server calls the Web gateway (WGate) (**2**), which passes this request to the ITS process, the application gateway (AGate).

AGate loads the appropriate service and opens the connection to the SAP System. The data is sent to the application server of the SAP System using the appropriate context.

The SAP System calls the BAPI and passes the resulting output data to the ITS AGate. (**5**).

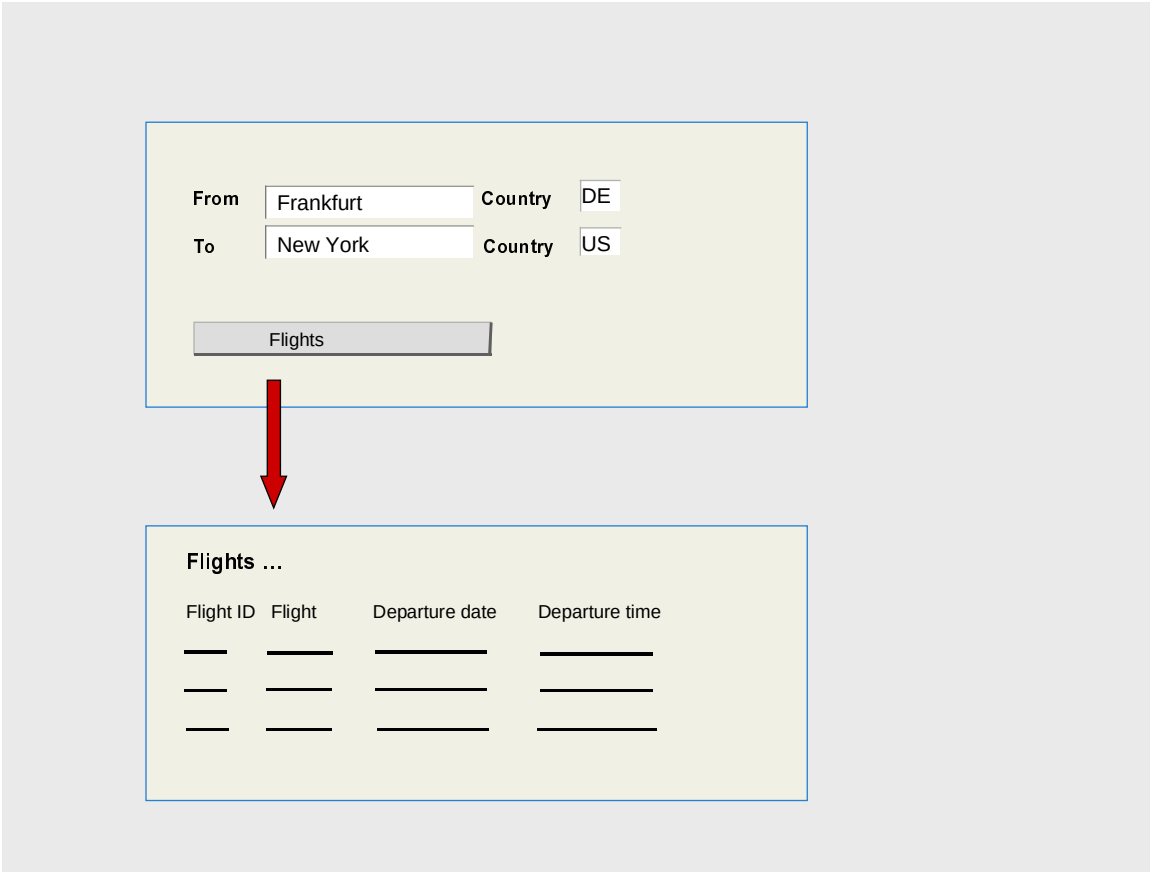
The AGate finds the correct HTML template and inserts the output data using an interpreter.

The resulting, formatted Web page is passed to the WGate (**7**) and then, via the Web server (**8**), to the Web browser (**9**). The browser then displays the results of the BAPI call.

Step 1: Designing an User Interface

The Web application that you implement will simply provide a summary of all flights between a given departure and destination airport.

The form used to enter this data consists of four input fields, for each of the mandatory input parameters for the BAPI. When the user presses a button, the system displays the flight data in a separate window.



Summary:

You will be implementing an application that includes a screen change. Each screen contains only a few areas. The application as a whole features simple, sequential navigation and intuitive use.

Step 2: Creating a Service

Use

To start a Web application and log on to the SAP System, there must be a service for the ITS. The service file is represented in SE80 using a development object of the *Internet Service* type. The system also needs a set of specific parameters to assign the Web application to a service.

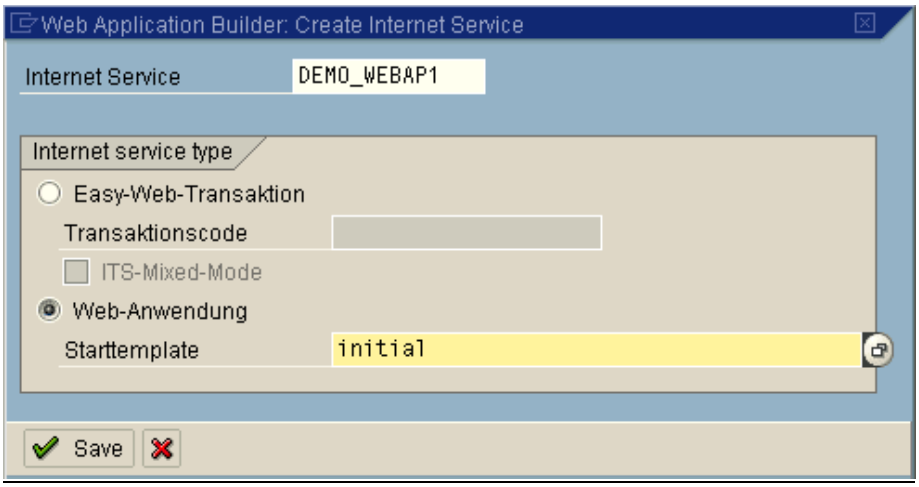
Procedure

9. Open the *Object Navigator* (transaction: SE80).

10. From the object list, choose *Internet Service*.
11. Enter a name for the service you want to create.
12. Click  or choose *Enter* to confirm your entry.

    The system checks to see whether or not an object with that name already exists, and if not, opens the *Create object* dialog box.
13. Choose *Yes* to create the service.

    The system displays the *Create service* dialog.



14. Choose *Save*.

    The system then displays the *Create object catalog entry* dialog box.

8. Assign a development class to the service.

    The system displays the service you have created in the object list tree structure. By default, *Theme 99* is automatically assigned to the service as the current theme.
9. In the object list, double-click on the new service.
10. Choose *Change* mode and click the *Parameters* tab.
11. Enter other parameters and values as necessary.

You can use the possible entries help to display a list of all service parameters, including a short description of each one.

Parameter	Value	Description
~INITIALTEMPLATE	<Template name> for example, initial	Name of the HTML template that is used to start the Web application (choose any name you want). This parameter is mandatory for flow-based Web applications.
~WEBAPPLICATIONTYPE	miniapp	Type of service This parameter is mandatory for MiniApps
~XGATEWAY	sapxginet	X gateway name This parameter is mandatory for flow-based Web applications.
~LOGIN		For MiniApps, this parameter is entered without a value

~PASSWORD		For MiniApps, this parameter is entered without a value

If you create a MiniApp, you must enter the two service parameters, **~LOGIN** and **~PASSWORD** without a value, so that later, when the MiniApp is integrated into the Workplace, the system can get the user data from the cookie. (When users log on to the Workplace, their User ID and password are stored in a Cookie; otherwise Single Sign-On does not work).

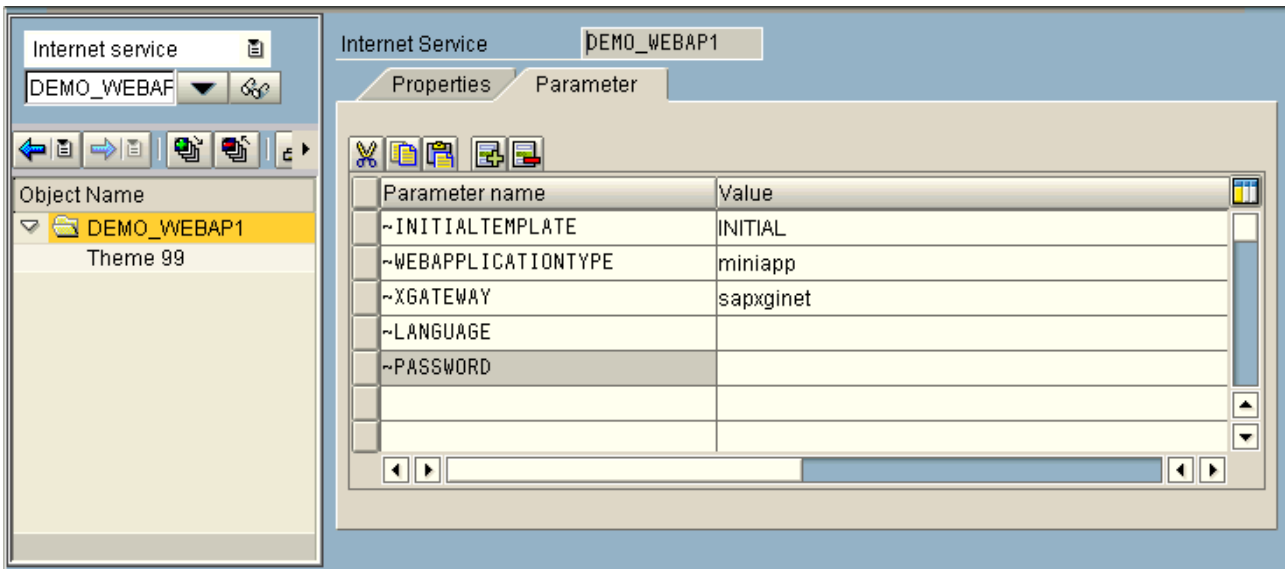
When you test the Web application as a standalone application (in Step 7 of this tutorial), the system displays a message relating to these two parameters

12. Choose .

Result

The Internet service is stored in the R/3 Repository as a development object.

Theme 99 is created as the current theme for the service. The additional parameters specify the type of service and the way the ITS will execute it. The example in this tutorial deals with a Web application of the **MiniApp** type, which is started with the `initial` template using the X Gateway **`sapxginet`**.



Step 3: Creating HTML Templates

Use

To implement Web pages, you must first create HTML templates for each flow-based Web application. You must provide a layout and dialog logic for each template. The Web application in our example includes a screen change, so we need to create two HTML templates.

Prerequisites

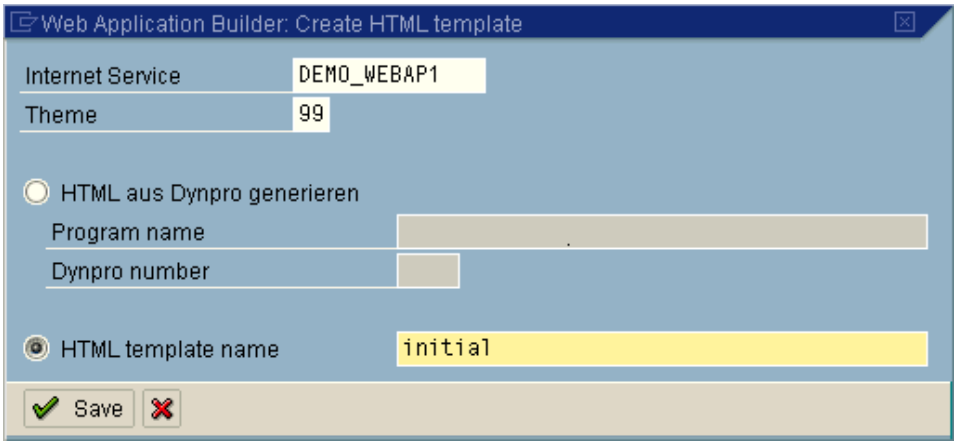
You have already created an Internet service.

Procedure

11. From the object list, select the appropriate service and choose *Create* → *Template* from the context menu.

    The system displays the *Create Template* dialog box.

12. Enter a theme for the service and a name for the first HTML template in the appropriate fields (in this tutorial, we suggest `INITIAL`). This name matches the value of the `~INITIALTEMPLATE` parameter.



13. Choose *Save*.

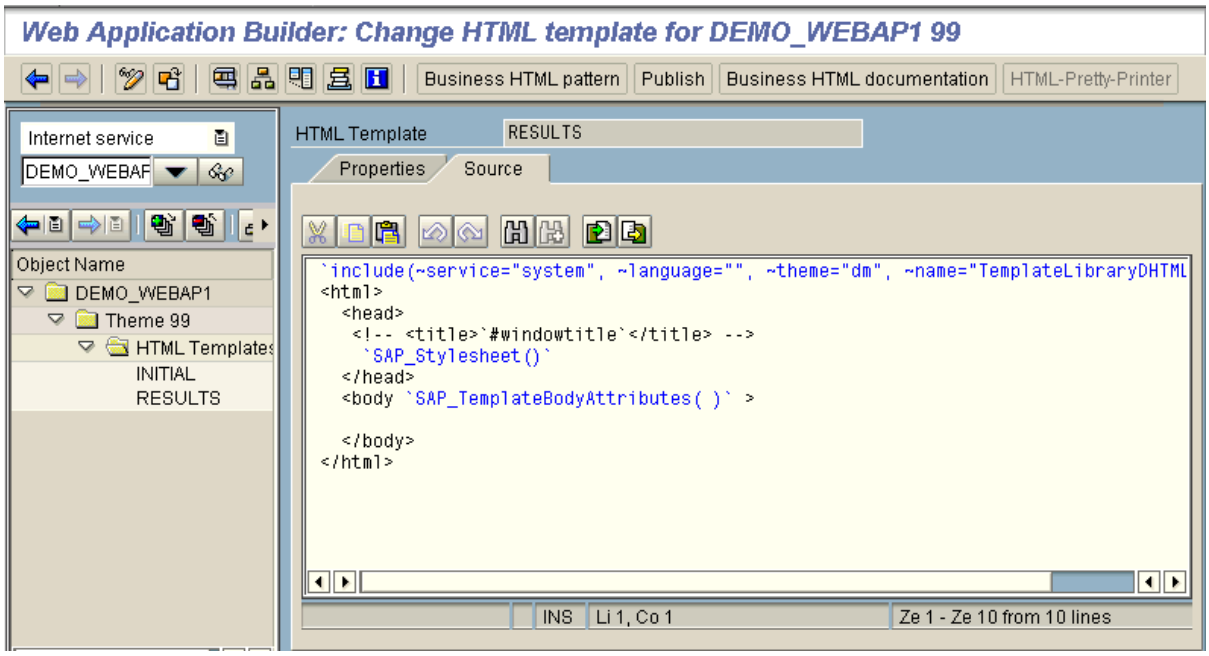
    The system then displays the *Create object catalog entry* dialog box.

5. Assign a development class to the template and choose .

6. Repeat steps 1-5 for the second template. In step 3, enter a name for the template (we suggest `RESULTS`).

Result

The system inserts the templates you have created in the object list tree structure. It then displayed the pre-generated framework for the HTML and HTML[Business] source text in the HTML Editor.



Step 4: Defining a Layout

Prerequisites

- You have already created the HTML templates.
- You are familiar with the interface definition for the BAPI.
- You know enough about HTML and HTML[Business] to implement a simple layout for the two Web pages.

Procedure

To create the layout for the first template (`INITIAL`), insert the following form definition between the *<body>* and*</body>* tags:

```
<!-- Form definition -->
<form ACTION="`wgateURL()`" METHOD="POST">
<br>
<table>
  <tr>
    <td>From   </td>
    <td><INPUT TYPE="TEXT" NAME="FromCity" Value="FRANKFURT"></td>
    <td>   Country  
        <INPUT TYPE="TEXT" NAME="FromCountry" Value="DE"SIZE=2
MAXLENGTH=2>
    </td>
  </tr>
  <tr>
    <td>To    </td>
    <td><INPUT TYPE="TEXT" NAME="ToCity" Value="NEW YORK"></td>
    <td>   Country  
       <INPUT TYPE="TEXT" NAME="ToCountry" Value="US" SIZE=2
MAXLENGTH=2></td>
  </tr>
  <tr>
    <td colspan=3> <br> <br>
   <!-- To raise a Flow Logic event clicking on the button -->
    <INPUT TYPE="SUBMIT"  NAME="~Event" VALUE="Get Flights">
    </td>
  </tr>
</table>
</form>
```

To create the layout for the first template (RESULTS), insert the following source text between the <*body*> and<*/body*> tags:

```
<!-- BAPI RETURN message in case of error -->
`if ( RETURN-TYPE == "E" )`
    `RETURN-MESSAGE`
`else`
    <br>
    Flights from `FromCity` (`FromCountry`) to `ToCity` (`ToCountry`):<br>
<br>
`end`
<!-- Listing output data -->
<table border=1 cellspacing=0>
  <tr>
    <th>Flight ID</th> <th>Flight Connection</th> <th>Flight Date</th>
<th>Departure Time</th>
  </tr>
  `repeat with I from 1 to FLIGHTLIST-CARRID.dim`
  <tr>
    <td width=40 align=center> `FLIGHTLIST-CARRID[i]`</td>
    <td width=60 align=center> `FLIGHTLIST-CONNID[i]`</td>
    <td width=80 align=center> `FLIGHTLIST-FLDATE[i]`</td>
    <td width=80 align=center> `FLIGHTLIST-DEPTIME[i]`</td>
  </tr>
  `end`
</table>
```

Result

You have defined a simple representation of the interface for the two templates (without dialog logic) using HTML and HTML[Business].

Step 5: Implementing the flow logic

Use

To implement the dialog logic for each template, use the language elements of the flow logic. When this flow logic is implemented, the system calls the BAPI and automatically transfers the data from the BAPI to the HTML template and back. The flow logic also specifies how the templates are to be filled with data.

Prerequisites

- You have already created two HTML templates.
- You are familiar with the basic principles of the flow logic.

Procedure

1. In the object list tree structure, double-click the initial template, `INITIAL`.
2. Choose *Edit* → *Create flow logic.*
3. For the first template (`INITIAL`), enter the following flow logic:

```
<flow>
    <event name = "Get Flights"   next_template="results">
    </event>
</flow>
```

Repeat steps 1-3 for the second template (`RESULTS`), entering the following flow logic:

```
<flow>
  <state name="GetList">
    <module name="BAPI_SFLIGHT_GETLIST"  type="RFC">
       <inputmapping source="FromCountry" target="FROMCOUNTRYKEY" />
       <inputmapping source="FromCity" target="FROMCITY" />
       <inputmapping source="ToCountry" target="TOCOUNTRYKEY" />
       <inputmapping source="ToCity" target="TOCITY" />
    </module>
  </state>
  <event name = "onLoad"  next_state = "GetList">
  </event>
</flow>
```

You can use the type `BAPI` instead of the type `RFC` when the module is called. If you do, you must enter `FlightConnection.GetList` as the module name.

Result

You have implemented the dialog logic for both HTML templates and can publish the whole service on the ITS.

Step 6: Publishing the Service

Use

To allow users to execute the Internet service you have created in the Workbench, you must copy it (and all the sub-objects belonging to it) to the ITS file system – that is, you need to publish the entire service.

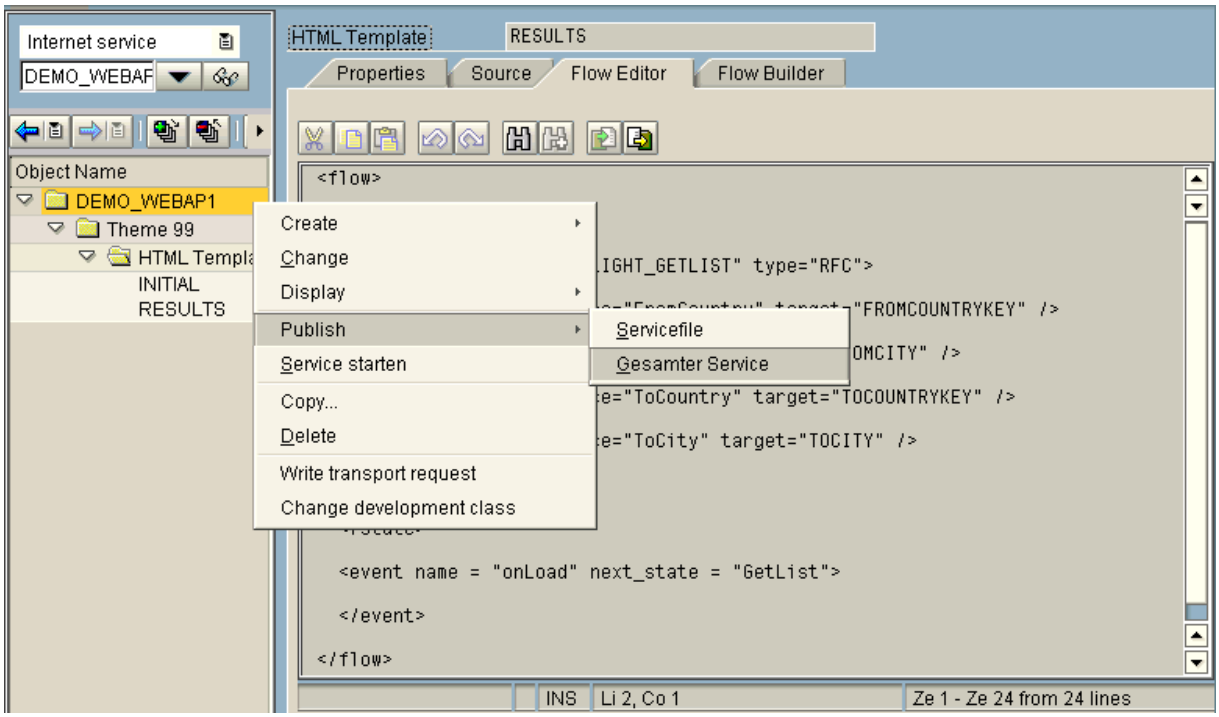
Prerequisites

You have assigned at least one ITS instance to the relevant SAP System and have ensured that it is active.

To find out which ITS instances are assigned to your SAP System, check your user settings. Choose *Utilities* → *Settings* and then the *ITS* tab.

Procedure

1. From the object list, select the appropriate service.
2. From the context menu, choose *Publish* → *Complete service*.



Result

Unless a error occurs during publishing, the system displays the message *Object created successfully* in the status bar. The status display (to the right of the object name) shows the new status of the object: *Saved/published* or *Saved/partly published*.
After you have published the whole service, you can start the Web application.

Step 7: Executing the Web Application

To test the Web application, we will execute it as a standalone application (without it being integrated into the Workplace architecture).

Prerequisites

The complete Internet service has been published successfully. Both the Web server and the ITS are active when the Web application is executed.

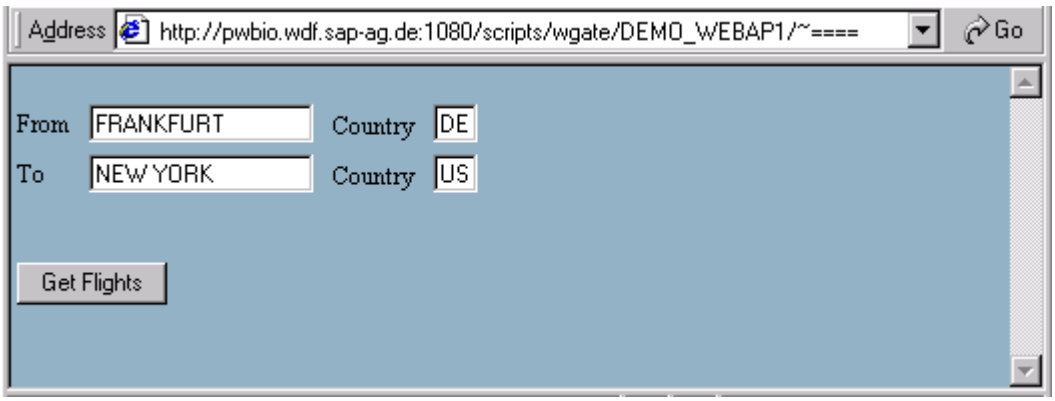
Procedure

5. From the *Object Navigator*, select the appropriate service.
6. From the context menu, choose *Start service* (or choose F8).

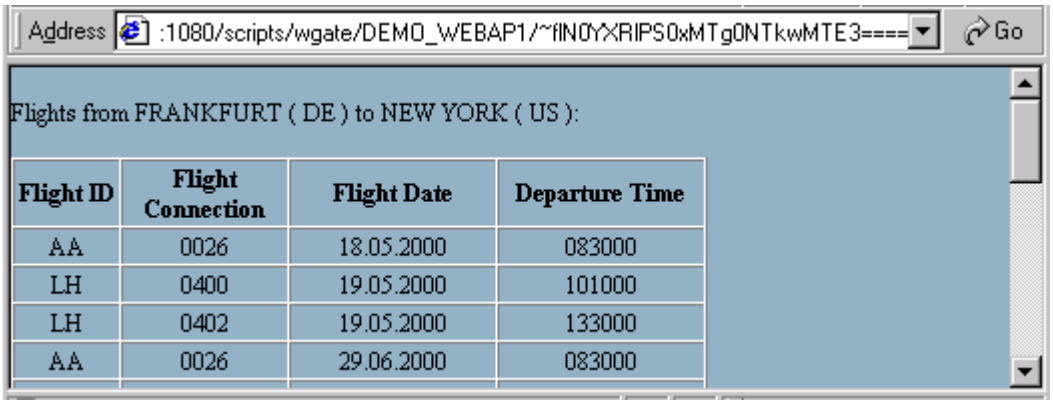
   The system displays the Web browser and a logon dialog box.
7. To log on to the ITS, enter you User ID and password and choose *Logon*.

Result

The system starts the service, using the HTTP address, **http://<web_server><web_path_prefix>/<service>/!**, and displays the first page of the Web application in the browser.



Using the *Flights* button, you trigger an event in the flow logic, which in turn triggers the BAPI call. The results of the BAPI call are inserted in the output template and displayed on the next page of the Web application.



For more information on the sequence in which the system executes each step when the Web application is being tested, see ITS Architecture Components