Internet Application Development With Flow Files: Tutorial

HELP.BCFESITSFLOW

Release 4.6C

Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of
Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.

HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium,
Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

Internet Application Development With Flow Files: Tutorial

Internet Application Development With Flow Files: Tutorial

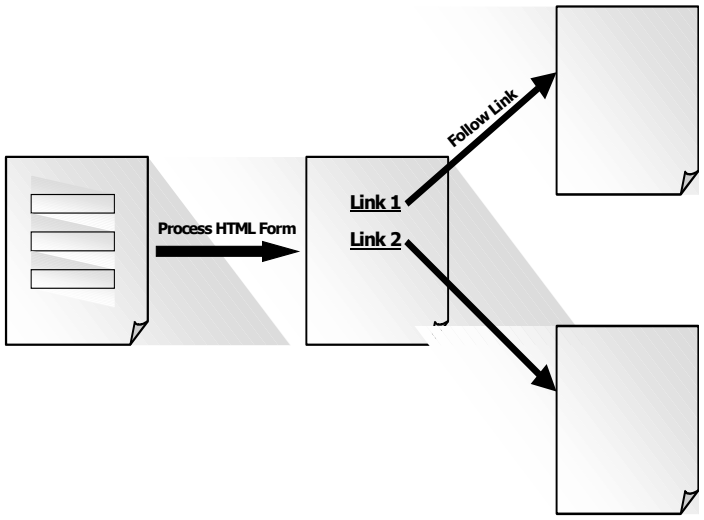
Purpose

This documentation is a step-by-step tutorial that describes how to develop Internet applications that are driven by the Internet Transaction Server (ITS) and use flow files.

The flow file implementation model allows you to develop Internet applications that consist of linked HTML pages, which you can populate with data retrieved from the R/3 System (or any other external system). The pages can offer a range of application functions, and are generated by following hyperlinks or processing HTML forms. The dialog flow is determined on the client side by the user, who can navigate freely between pages.

Since the dialog flow is not fixed in advance, much depends on what the user decides to do. This contrasts with the dialog flow in other business scenarios, where the business application can put restrictions on how users can navigate.

The following graphic illustrates the basic concept:



This tutorial describes how to develop applications that use flow files where the business logic is implemented in remote-enabled function modules (RFCs) called from the R/3 System. You can also implement the business logic with Business APIs (BAPIs).

In future releases, it will be possible to define module calls from any external system.

Internet Application Development With Flow Files: Tutorial

Implementation Considerations

You should consider using this implementation model for applications that offer many application functions on one page, and the dialog flow is not fixed in advance.

Such applications have simple point-and-click user interfaces, limited manual data input, and reduced data formatting requirements. They are often used in e-commerce scenarios.

Integration

To develop applications that use flow files where the business logic is implemented in module calls from the R/3 System, you need to install the following components:

- The ITS

  The ITS forms the interface between the R/3 System and the Internet.

- The SAP@Web Studio

  The SAP@Web Studio is a PC tool for implementing services, which include all the files required by the ITS to drive applications.

- The R/3 System

Features

Like all other implementation models for developing Internet applications driven by the ITS, this model allows you to develop applications that send documents back to the Web browser client in HTML format, since this format can be handled by all major Web browsers.

Like all other implementation models, there is a clear separation between business logic and presentation aspects. In this case, defining the dialog flow is also a separate task.

- You implement a set of modules that comprise the business logic in the R/3 System (or other external system).

  If you are implementing the business logic in the R/3 System, you create Business APIs (BAPIs) or standard remote-enabled function modules with the Function Builder in the ABAP Workbench. This tutorial uses remote-enabled function modules.

- You implement the presentation and the dialog flow in the SAP@Web Studio.

  To do this, you need to create an ITS service, which contains all the files required to implement and run the application.

  - The presentation determines the look and feel of each application.

    You design the presentation by creating a set of HTML^{Business} templates.

  - The dialog flow determines which template is displayed when, depending on what the user decides to do next.

    You implement the dialog flow by defining flow logic in flow files. Flow logic operates like a state machine, because it defines the sequence in which modules are called based on events and exceptions.

    There is one flow file for each HTML^{Business} template that requires a dialog flow definition.

Internet Application Development With Flow Files: Tutorial

When you have defined the business logic in R/3, and created all the files required by the ITS to run an application in the SAP@Web Studio, you can check the ITS files into source control in R/3, and test the application. Users can run the application from any suitable Web browser.

What the Tutorial Covers

This tutorial is designed to be read as a continuous document. It covers the steps you have to take if you want to develop Internet applications using flow files, with reference to a specific example application.

This documentation is a reference manual, which describes the implementation model, details flow logic syntax, and includes two example scenarios.

For a step-by-step introduction to developing Internet applications with flow files, with the help of an example application,

Further Information

For a full description of the implementation model, including a list of flow logic syntax elements and their usage, see:

Internet Application Development With Flow Files: Reference [Ext.]

For further information about other ITS implementation models and other aspects of the ITS, see:

ITS Implementation Models [Ext.]

SAP@Web Studio [Ext.]

HTMLBusiness Language Reference [Ext.]

ITS Administration Guide [Ext.]

If these links do not work from your current location, you can find this documentation:

- In the R/3 System:

  See under *Help → SAP Library → Basis Components → Frontend Services (BC-FES) → ITS / SAP@Web Studio (BC-FES-ITS)*

- In the SAP@Web Studio:

  See under *Help* .

What the Tutorial Does Not Cover

This tutorial does **not** teach you how to create remote-enabled function modules or BAPIs. It assumes that you are familiar with ABAP programming techniques, and that you know how to use the various programming tools available in the ABAP Workbench of the R/3 System.

Further Information

For further information about ABAP programming techniques, ABAP Workbench tools, and BAPI programming, click on the relevant link below.

If the links do not work from your current location, and you are outside the R/3 System, log on to the latest release and follow the directions for the documentation you want to view.

- ABAP Programming [Ext.]

  In the R/3 System, see under *Help → SAP Library → Basis Components → ABAP Programming and Runtime Environment (BC-ABA) → BC – ABAP Programming*

Internet Application Development With Flow Files: Tutorial

- ABAP Workbench Tools [Ext.]

  In the R/3 System, see under *Help → SAP Library → Basis Components → ABAP Workbench (BC-DWB) → BC – ABAP Workbench Tools*

- ABAP Workbench Tutorial [Ext.]

  In the R/3 System, see under *Help → SAP Library → Basis Components → ABAP Workbench (BC-DWB) → BC – ABAP Workbench Tutorial*

- BAPI Programming Guide [Ext.]

  In the R/3 System, see under *Help → SAP Library → Cross-Application Components → Business Framework Architecture (CA-BFA)*

Flow File Application Tutorial: Prerequisites

Before you start working through the tutorial, you should:

- Install the Internet Transaction Server (ITS) Release 4.6C (or higher).
- Install the SAP@Web Studio Release 4.6C (or higher).
- Install the R/3 System Release 4.6C (or higher).

  To view the function modules and ITS objects that make up the example application, you need Release 4.6C or higher.

  To develop applications with flow files, you need Release 4.6A or higher.

  Some Business APIs (BAPIs) are not available in R/3 releases prior to 4.6B. For this reason, SAP recommends that you use R/3 Release 4.6B or higher, if you intend to develop flow file applications that use BAPIs to define the business logic.

  In future, it will be possible to develop applications that use flow files in earlier R/3 releases.

You should also:

- Ensure that the remote-enabled function modules and all the required ITS files that make up the example application are available in the R/3 System.

  The development class is COM_ITS_EXAMPLES.

  – Function group AUCTION_FUNCTIONS

    This function group contains the function modules that used to implement the business logic.

  – Internet service AUCTIONNEW

    This ITS service contains the ITS files – service file description, HTMLBusiness templates, and flow files – used to implement the presentation and dialog flow.

- Set up debugging facilities.

  The ITS debugger allows you to start a Web transaction from your Web browser, and follow the processing on the R/3 side in a SAP GUI window.

  For full details, see Debugging Flow File Applications [Page 36].

If your company has an ITS administrator, most of these tasks have probably been completed.

For any other relevant information, check SAPNet note 191373.

Flow File Application Tutorial: Example Application

This tutorial uses an example application to demonstrate the steps involved in developing an Internet application that uses flow files.

The example application is an integrated auction scenario, which allows registered users to browse items for sale, sell items and bid for items. It also includes administration and search functions.

The example application consists of the following aspects:

- Business logic
- Service file, presentation, and dialog flow

Business Logic

The business logic is defined in a set of standard remote-enabled function modules delivered with your R/3 System.

For a list of function modules used in this application, see Appendix A: Function Modules Used in Example Application [Page 37] at the end of this tutorial.

The example application in this tutorial uses standard remote-enabled function modules, but you can also use Business APIs (BAPIs).

Service File, Presentation, and Dialog Flow

To enable you to run the application in a Web browser, the application also contains the following ITS files.

- Service file

  The service file contains all the parameters required by the ITS to run the application.

- HTML^{Business} templates

  HTML^{Business} templates define the presentation or “look and feel“ of the application.

  There is one template for each application screen.

- Flow files

  Flow files contain the flow logic, which defines the dialog flow of the application.

  There is one flow file associated with each HTML^{Business} template that needs a dialog flow definition.

All these files should be available in your R/3 System.

You can use the example HTML^{Business} templates and flow files as a basis for designing your own.

For list of all ITS files – service file, HTML^{Business} templates, and flow files - used in the application, see Appendix B: ITS Files Used in Example Application [Page 39] at the end of this tutorial.

Running the Example Application

When you have all the function modules and ITS files in your R/3 System, you can run the example application from your Web browser with the URL

```
http://<myhost>:myport/scripts/wgate/auctionnew/!
```

This brings up the first screen of the application, which you can explore to see how it works.

Flow File Application Development: Main Steps

Purpose

This process describes the main steps involved in designing an Internet application that is driven by the Internet Transaction Server (ITS) and uses flow files.

Prerequisites

You should determine that this is the appropriate implementation model for your application.

Process Flow

To create an application that uses flow files, you need to:

1. Design the application [Page 14].

   To do this, you need to have a basic conceptual idea of what your application is supposed to do. Then, you can decide what templates are needed, how and when they are to be called, and what business logic you will need to define to process the data.

2. Define the business logic [Page 20] in the R/3 System.
3. Develop the rest of the application in the SAP@Web Studio.
   a. Set up the environment [Page 22].
   b. Create an ITS service [Page 26] to run the application.

      Each application has one service description, which contains parameters that define how the application is run.

   c. Define the presentation [Page 28].

      Each application consists of several linked HTML pages offering the user functions and hyperlinks, so you need to create a set of HTML^{Business} templates.

      HTML^{Business} templates contain standard HTML and HTML^{Business} statements. HTML^{Business} is an SAP-specific macro language, which allows you to merge R/3 data into templates at runtime.

   d. Define the flow logic [Page 30].

      For each HTML^{Business} template that needs a dialog flow definition, there is an associated flow file, which defines the flow logic that allows the user to navigate in the application.

      Flow files contain Extensible Markup Language (XML) statements, using a predefined subset [Ext.] of XML elements.

   e. Create any other ITS files [Page 33] you require.

      These additional files include language resources and Multipurpose Internet Mail Extension (MIME) files.

      - Language resource files contain all the language-specific texts used in HTML^{Business} templates. These files are optional, but it makes sense to use them, because they allow you to create a single set of HTML^{Business} templates for all languages.
      - MIME files contain any graphics, images, sound or video components that you may want to include in your ITS service. These files are also optional.

f. Publish the service to an ITS site [Page 34].

g. Add the files to ITS Source Control [Page 35].

4. Debug your application [Page 36].

Result

When you have performed these steps, you should have a complete flow file application, which you can run from any suitable Web browser.

Although it is not critical whether you implement the business logic first or define the presentation and dialog flow aspects first, it makes sense to follow the sequence described in this tutorial.

Designing the Flow File Application

Use

If you decide to implement an Internet application using flow files, you need to have a precise conceptual idea of the application you want to develop.

You need to know:

- The main purpose of the application.
- What functions the application should offer the user.
- How the user should be able to navigate.
- How the application should look.

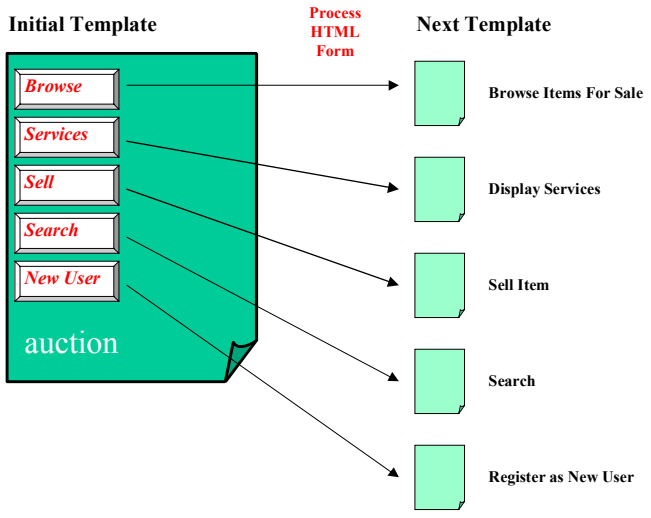
When you can answer these questions, you will know:

- What function modules you need to create in R/3 for the business logic.
- What templates you need and how they should be organized.
- The sequence in which the templates are processed.

Procedure

To map out your application, start with a state diagram.

Using the example AUCTION application as a blueprint, you could envisage an initial state similar to the following:



The initial template of this application offers several functions. If the user chooses one of these functions, you may need to call one or more function modules and display the appropriate template, or you may only need to display the appropriate template. On the next template, the process is repeated.

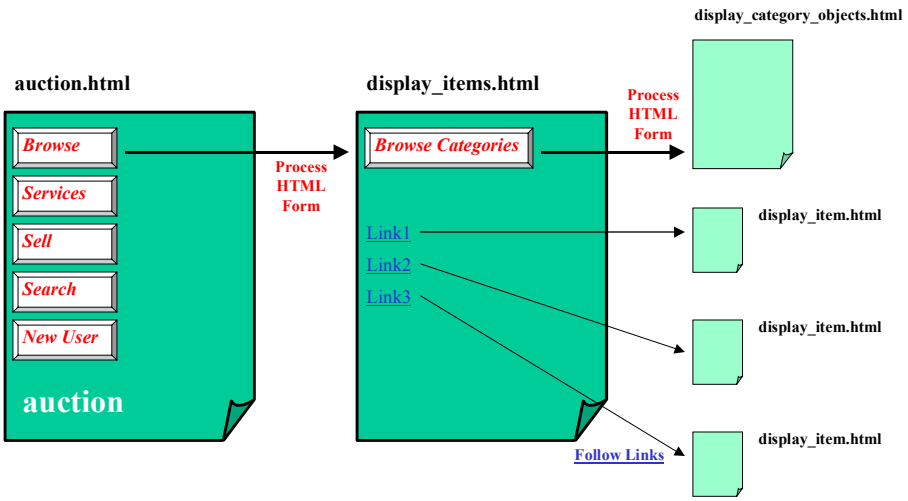
As shown in the above graphic, the initial template **`auction.html`** could offer the following functions on five pushbuttons:

- *Browse*
- *Services*
- *Sell*
- *Search*
- *New User*

In turn, all these functions could take the user to templates that contain further input fields, pushbuttons, and links.

Let's look more closely at each function offered by **`auction.html`** to see how the dialog flow might develop:

Browse



If the user chooses *Browse*, the next template **`display_items.html`** could:

- List all items for sale in the database.

  Each item could be a link to more detailed information about the item. This would include details about the seller and allow the user to bid for the item, register as a new user (if not already registered), and return to the homepage (**`auction.html`**).

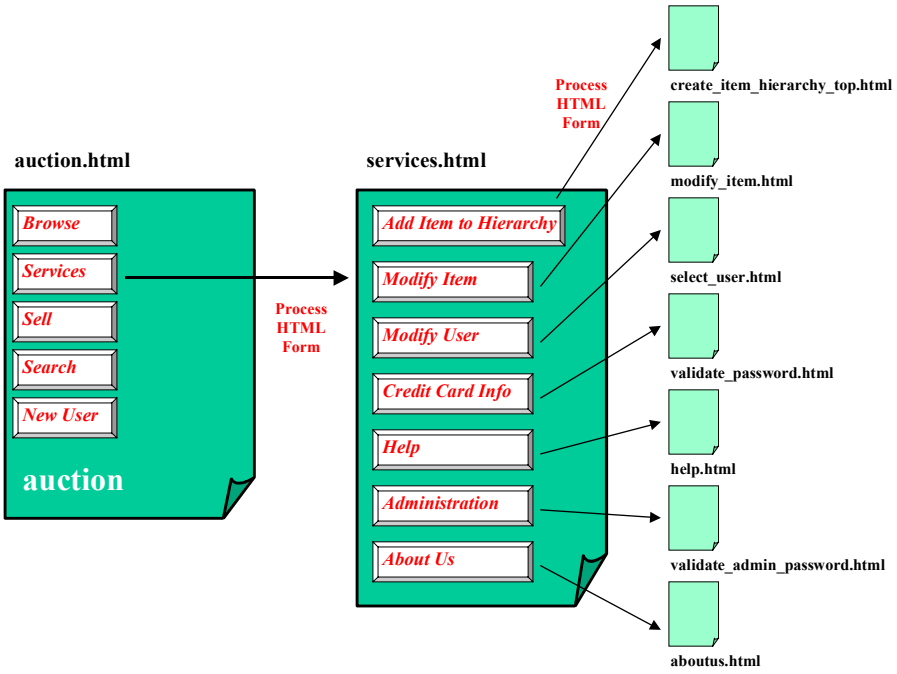
Designing the Flow File Application

- Offer a pushbutton to browse the item categories into which the database is divided.

To display all items for sale on the next template **`display_items.html`**, you need a module that gets all items from the database and displays them. In the example application, this is performed by the module AUCTION_DISPLAY_ITEMS.

Each item ID could serve as a parameter for retrieving detailed information.

Services



If the user chooses *Services*, the next template next template **`services.html`** could offer a further set of pushbuttons:

- *Add Item to Hierarchy*

  This pushbutton displays a template where users can add items to item categories.

- *Modify Item*

  This pushbutton displays a template where users can modify items they have entered in the database. The application must validate each user and check their authorization.

- *Modify User*

  This pushbutton displays a template where users can modify their own personal details, including payment information. The application must validate each user and ensure that they cannot modify details other than their own.

- *Credit Card Info*

  This pushbutton displays a template where users can enter credit card information. The application must validate each user.

- *Help*

This pushbutton displays a template where users can get help.

- *Administration*

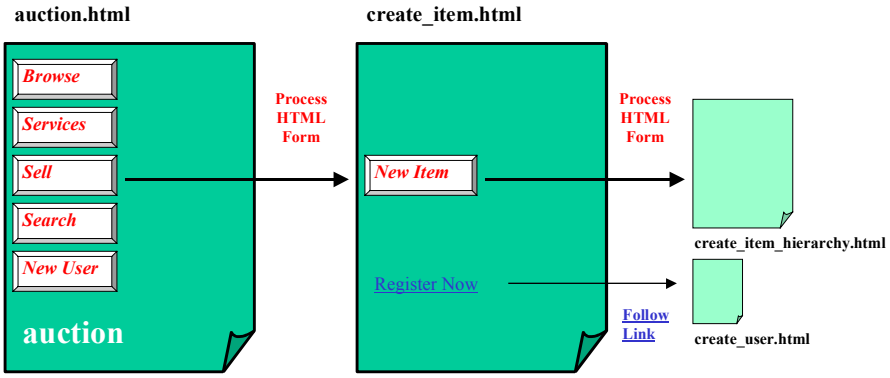
  This pushbutton displays a template that offers a range of administration functions. To use these functions, the user must have administrator rights.

- *About Us*

  This pushbutton displays a template that displays general information about the company running the auction scenario.

In this case, no business logic is required, so you do not need to call any function modules. You only need to point to the next template **`services.html`**.

Sell



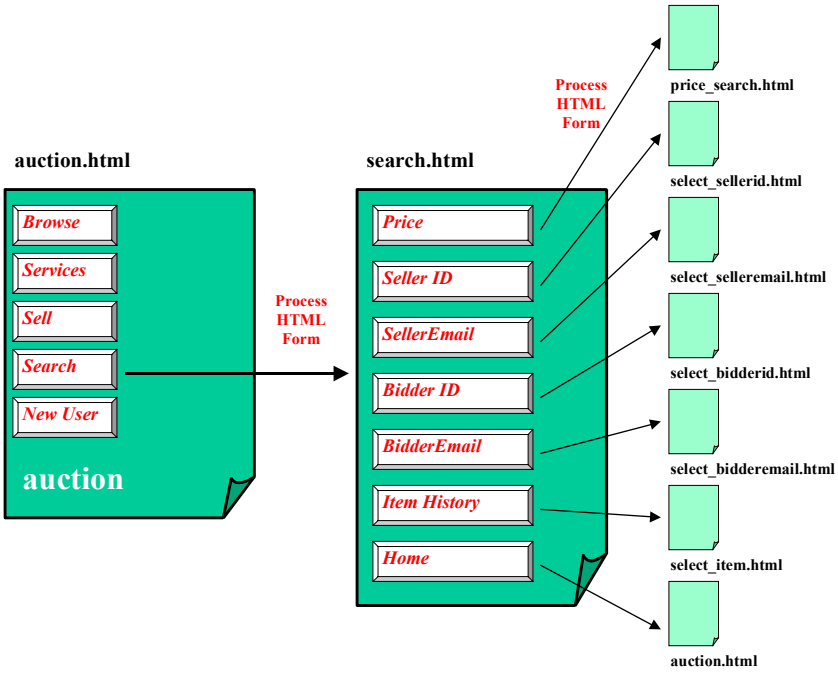
If the user chooses *Sell*, the next template **`create_item.html`** could allow users to enter a new item for sale and add it to the database. New users must register first by clicking on the link *Register Now*.

When allowing users to create new items, you need to know which users are already registered, and the existing item categories, so you need to call the following function modules:

- AUCTION_DISPLAY_BIDDERS
- AUCTION_DISPLAY_CATEGORIES

You also need to provide a pushbutton to add the item to the database and a link for new users to register.

Search



If the user chooses *Search*, the next template **`search.html`** could offer a further set of pushbuttons:

- *Price*

  This pushbutton displays a template where users can search for items within a specified price range.

- *Seller ID*

  This pushbutton displays a template where users can search for sellers by seller ID.

- *Seller Email*

  This pushbutton displays a template where users can search for sellers by seller e-mail.

- *Bidder ID*

  This pushbutton displays a template where users can search for bidders by bidder ID.

- *Bidder Email*

  This pushbutton displays a template where users can search for bidders by bidder e-mail.

- *Item History*

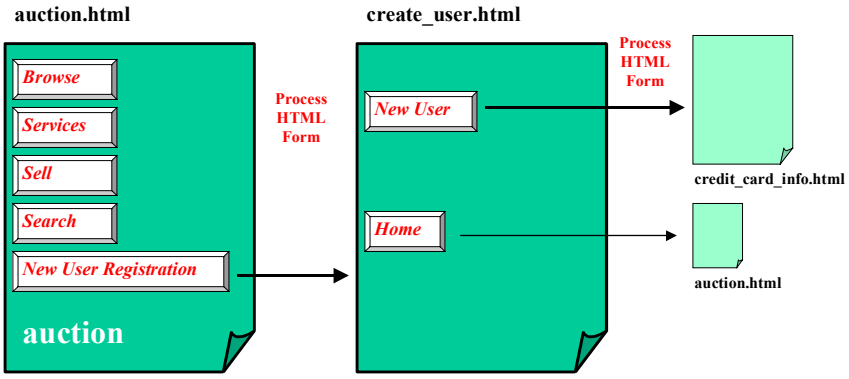
  This pushbutton displays a template that displays the history of each item in the database, including such details as item name, item description, price, as well as the deadline (date and time) for bids.

- *Home*

This pushbutton takes users back to the auction scenario homepage.

In this case, no business logic is required, so you do not need to call any function modules. You only need to point to the next template **`search.html`**.

New User Registration



If the user chooses *New User Registration*, the next template **`create_user.html`** allows new users to register. It also offers a pushbutton that takes users to the auction scenario homepage.

In this case, no business logic is required, so you do not need to call any function modules. You only need to point to the next template **`create_user.html`**.

Flow Logic

To determine the dialog flow of the application, depending on what the user decides to do at a particular point, some templates have flow files where you define the flow logic.

In the example auction application, the flow file for the initial template **`auction.html`** is called **`auction.flow`**.

Result

When you have designed the application, you can define the business logic in R/3 [Page 20].

Defining the Business Logic in R/3

Use

To implement the business logic in your flow file application, you can create a set of remote-enabled function modules in the R/3 System.

Prerequisites

You have designed your application [Page 14].

Procedure

1. In the ABAP Workbench, create a development class (if necessary).
2. In the Function Builder, create a function group for your application.
3. Create a function module.

   To get and display all items in database, the example application uses a function module called AUCTION_DISPLAY_ITEMS.

   Here is the code:

```
FUNCTION AUCTION_DISPLAY_ITEMS.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  EXPORTING
*"     VALUE(RETURNSTATUS) TYPE  BAPIRETURN
*"  TABLES
*"      IAUCTIONITEMS STRUCTURE  AUCTIONITEMS
*"----------------------------------------------------------------------
select * from AUCTIONITEMS
into table IAUCTIONITEMS
order by primary key.
if sy-subrc <> 0.
   RETURNSTATUS-TYPE = 'E'.
   exit.
endif.

RETURNSTATUS-TYPE = 'S'."all went well

ENDFUNCTION.
```

Result

You have defined the business logic for your application. Now, you can design the rest of the application in the SAP@Web Studio. Start by setting up your environment [Page 22].

When you have done this, you can design the presentation aspects of your application by creating HTML^{Business} templates, and define the dialog flow in flow files.

For further information about creating function modules in the Function Builder, and using other tools in the ABAP Workbench, see ABAP Workbench Tools [Ext.].
If this link does not work from your current location, you may be outside the R/3 System. In this case, log on to the latest release of the R/3 System and see under *Help → SAP Library → Basis Components → ABAP Workbench (BC-DWB) → BC – ABAP Workbench Tools*.

For a list of all funcion modules used in the example application, see Appendix A: Function Modules Used in Example Application [Page 37] at the end of this tutorial.

SAP@Web Studio Setup

Purpose

To create the necessary Internet Transaction Server (ITS) files – service file, HTML^{Business} templates, and flow files – to run your application, you can use the SAP@Web Studio.

When you have created all these files, you can publish them to an ITS site and add them to ITS source control in the R/3 System. There, you can assign them to a change request, just like any other development object.

This process describes how to set up the environment in the SAP@Web Studio.

Prerequisites

The SAP@Web Studio must be installed.

Process Flow

To set up the environment in the SAP@Web Studio, you need to:

1. Start the SAP@Web Studio [Page 23].
2. Create an ITS project [Page 24].
3. Define an ITS site [Page 25].

Result

When you have set up the environment, you can begin to create all the other ITS files required to run your application. Start by creating an ITS service [Page 26].

Starting the SAP@Web Studio

To create the Internet Transaction Server (ITS) files required to run your application, you need to start the SAP@Web Studio and set up your environment.

Prerequisites

The SAP@Web Studio must be installed.

Procedure

On the Windows NT desktop, choose *Start* → *Programs* → *SAP@Web Studio* → *Studio <release number>*.

Result

The SAP@Web Studio opens on your desktop. Now, you can create an ITS project [Page 24].

Creating an ITS Project

While you are developing an Internet application in the SAP@Web Studio, you can store all the constituent files in an Internet Transaction Server (ITS) project while you work on them.

This procedure describes how to create an ITS project.

Procedure

1. Start the SAP@Web Studio as described in Starting the SAP@Web Studio [Page 23].
2. Choose *File* → *New* and select the *Projects* tab.
3. Enter a name for the project in the *Project Name* field and choose *OK*.

   The SAP@Web Studio creates your project, and automatically opens it in the *File View*.

   If you close the SAP@Web Studio and return to work on your application later, you can open the project in either of the following ways:

   - Choose *File* → *Open Project* and browse the file system to find the project.
   - Choose *File* → *Recent Projects* and select the project from the list.

     When you open a project, it becomes the default location for any files you create.

Result

You have created an ITS project. Now, define an ITS site [Page 25].

Defining an ITS Site

Use

When you develop an Internet application in the SAP@Web Studio, you need to define an Internet Transaction Server (ITS) site.

An ITS site is the location to which you publish (transfer) all the files you create in the SAP@Web Studio to implement an application.

- The location is an R/3 System.
- The files are ITS files.
  - Dynamic files such as service descriptions, HTML^{Business} templates, flow files, and language resources are stored in the appropriate ITS server directories.
  - Static Multipurpose Internet Mail Extension (MIME) files containing graphics, images, and video elements are stored in the Web server directory.

During the development stage, the SAP@Web Studio uses the ITS site to access all the information it needs to generate a service.

Prerequisites

You have created an ITS project [Page 24] to store your files.

Procedure

To define an ITS site:

1. Start the SAP@Web Studio as described in Starting the SAP@Web Studio [Page 23].
2. Choose *Project* → *Site Definition*.
3. Choose *New*.
4. Enter a site name, and choose *Next*.
5. Enter the Web server host name, and choose *Next*.
6. Enter the ITS server host name, and choose *Next*.
7. Enter an ITS virtual instance name, and choose *Next*.
8. Enter additional Web server information as necessary.
9. Choose *Finish* to confirm
10. Choose *OK* to exit.

Result

The SAP@Web Studio displays the ITS site in the list box located on the left of the application toolbar. You can now create the ITS files required to run your application. Start by creating an ITS service file [Page 26].

Creating an ITS Service File

Use

To run your application, you need to create an Internet Transaction Server (ITS) service. You may also wish to specify a theme.

- An ITS service is the set of external components required by the ITS to run an application.

  The service file of a flow file application must include a service description, HTML^{Business} templates, and flow files. It may also contain language resources and Multipurpose Internet Mail Extension (MIME) files.
  Each ITS service is defined by the ITS service description, which specifies essential parameters for running the application.

- An ITS theme is an instance of an ITS service with its own set of HTML^{Business} templates and language resources. An ITS theme lends an ITS service a particular look and feel, but the functionality remains the same. Unless you specify a different theme in the service file, the application uses the default value 99, which is set in the global service file **`global.srvc`**.

At runtime, the ITS merges the values you specify in the service file with the values specified in the global service file, which contains settings that apply to all services. The values in the service-specific service file override those in the global service file. If essential logon information such as client, user name, password, or logon language is missing from both these files, the ITS displays a logon screen, so that the user can enter the missing values.

Prerequisites

You have created an ITS project [Page 24] and defined an ITS site [Page 25].

Procedure

To create an ITS service in the SAP@Web Studio:

1. If it is not already running, start the SAP@Web Studio as described in Starting the SAP@Web Studio [Page 23].
2. Choose *File* → *New* and select the *Files* tab.
3. Select *Service File*.
4. In the *Service Name* field, enter a service name for your application and choose *OK*.

   The SAP@Web Studio creates the service in the *File View* of the project workspace and opens the grid control editor where you can specify all the required parameters.
5. In the *Name* column, place the cursor on the first empty row, and double-click.

   You see the default name **`~param1`**, but you can overwrite this. When you have entered a parameter name, you can tab to the *Value* column and enter an appropriate value.
6. Enter the following parameters and parameter values:

Parameter	Value	Description

`~xgateway`	`sapxginet`	The Xgateway is a plug-in interface that supports the various ITS implementation models This parameter must be set to the value `sapxginet`.
`~initialTemplate`	`<template name>`	This is the first template to be displayed when you run your application. The name of the initial template in the example application is `auction`.

To run an application that uses flow files, your service file **must** contain the above parameters. If you do not specify them, your application will fail.

7. Specify other parameters as required.

   If you wish, you can also define service-specific system and logon information, but none of these parameters are mandatory.

8. Save your work.

Result

The SAP@Web Studio creates a service file containing all the information required by the ITS to drive your application.

Here is the minimal service file needed to run the example auction application:

Parameter	Value
`~xgateway`	`sapxginet`
`~initialTemplate`	`auction`

Now define the presentation [Page 28].

Defining the Presentation

Use

To design a user interface for your flow file application, you need to create HTML[Business] templates for the Internet Transaction Server (ITS) service you have just created.

HTML[Business] templates determine the look and feel of your application. There is one template for each application screen.

HTML[Business] templates contain standard HTML code and HTML[Business] statements. When a user starts an application where the modules that define the business logic reside in the R/3 System, the ITS merges R/3 data, R/3 texts, R/3 error messages, non-R/3 texts and graphics, and dynamically generated URLs into a template by interpreting the HTML[Business] statements. The finished template is displayed in the user's Web browser.

Prerequisites

You have already created an ITS service file [Page 26].

Procedure

To create HTML[Business] templates for your application:

1. If it is not already running, start the SAP@Web Studio as described in Starting the SAP@Web Studio [Page 23].
2. Choose *File* → *New* and select the *Files* tab.
3. Select *HTML Business Template*.
4. In the *Template Name* field, enter a template name and choose *OK*.
5. Enter the HTML[Business] code that defines the template layout you require.
6. Save your work.

Result

You have created the initial template for your application. Now, define the flow logic [Page 30].

The initial template in the example application displays five pushbuttons:

Here is the HTML[Business] code for the initial template:

```
<html>

<form action="`wgateURL()`" method="post" target="_blank" >

<table>
  <tr>
    <td align=left>
```

```
        <input type="submit" name="~event" value="Browse">
    </td>
    <td align=left>
        <input type="submit" name="~event" value="Services">
    </td>
    <td align=left>
        <input type="submit" name="~event" value="Sell">
    </td>
    <td align=left>
        <input type="submit" name="~event" value="Search">
    </td>
    <td align=left>
        <input type="submit" name="~event" value="New User
Registration">
    </td>
  </tr>
</table>

</form>

</html>
```

Although it is convenient to use the editor in the SAP@Web Studio, you can create HTML[Business] templates with any suitable HTML editor.

Defining the Flow Logic

Use

Each HTML[Business] template in your flow file application that requires a dialog flow definition to respond to user actions must have an associated flow file that contains the flow logic.

The flow logic defines logical transitions between application states, depending on what the user chooses to do.

If there is an HTML[Business] template called **<template name>.html**, the asssociated flow file (of there is one) is called **<template name>.flow**. This flow file contains a set of events and states. Events define the entry points to different logical state(s), or how to load directly a different template.

You define flow logic using a subset of Extensible Markup Language (XML) elements.

Prerequisites

You have defined the presentation [Page 28] by creating an HTML[Business] template.

Procedure

1. If the HTML[Business] template for which you want to generate a flow file is not already open in the main work area of the HTML[Business] editor, open it by double-clicking.

   The SAP@Web Studio opens the template in the main work area.

2. Choose *Add Flow File* or *Edit* → *Flow File*.

   The SAP@Web Studio creates and open a flow file with initial **<flow>** and closing **</flow>** tags.

   You can toggle between the HTML[Business] code defined in the HTML[Business] template and the flow logic defined in the flow file by using the HTML Source and Flow Source buttons located in the status bar of the HTML[Business] editor.

3. Enter the flow logic between the **<flow>** and **</flow>** tags.

   For the initial template **auction.html** of the example auction application, we need a flow file that executes modules depending on what the user decides to do.

   auction.html contains 5 events represented by the pushbuttons *Browse*, *Services*, *Sell*, *Search*, and *New User Registration*.

   Some of these pushbutton go directly to the next template. Others contain processing.

   – If the user chooses *Browse*, the flow logic instructs the ITS to execute the module **auction_display_items**, which gets all items from the database.

     - If items exist, the application gets the next template **display_items.html** and displays the items.
     - If no items exist, the ITS gets and displays the default tempate **empty.html**.

- If the user chooses *Services*, no data has to be displayed, so the flow logic instructs the ITS to display the next template **`services.html`**.
- If the user chooses *Sell*, the flow logic instructs the ITS to execute the following modules:

  - **`auction_display_categories`** gets and displays all categories currently in the database.
  - **`auction_display_bidders`** gets all bidders currently in the database.

    The flow logic also instructs the ITS to display the template **`create_item.html`**.
- If the user chooses *Search*, no data has to be displayed, so the flow logic instructs the ITS to display the next template **`search.html`**.
- If the user chooses *New User Registration*, no data has to be displayed, so the flow logic instructs the ITS to display the next template **`create_user.html`**.

  Here is the code for the flow file **`auction.flow`**, which drives the initial template **`auction.html`**:

```
<flow>

<event name = "Browse" next_state="Browse">
</event>

<state name = "Browse">
      <module name= "auction_display_items" type="RFC">
            <default next_template = "empty"></default>
            <result next_template = "display_items">
              <expr>IAUCTIONITEMS-ITEMID.dim > 0 </expr>
              </result>
      </module>
</state>

<event name = "Services" next_template="Services">
</event>

<event name = "Sell" next_state="create_item">
</event>

<state name="create_item">
      <module name= "auction_display_categories" type="RFC">
      </module>
      <module name= "auction_display_bidders" type="RFC">
      <default next_template = "CREATE_ITEM"></default>
      </module>
</state>

<event name = "Search" next_template="SEARCH">
</event>

<event name = "New User Registration"
next_template="create_user">
</event>
```

```
</flow>
```

4. Save your work.

Result

You have created a flow file for your HTML[Business] template. Now, create the remaining templates [Page 28] and flow files.

When you have created all the HTML[Business] templates and flow files, do the following:

1. If necessary, create other ITS files [Page 33] you need for your application.

   These files could include languages resources and Multipurpose Internet Mail Extension (MIME) files.

2. Publish the files [Page 34] to an ITS site.

For a list of all the XML elements you can use when defining flow logic in flow files, see Flow Logic Syntax [Ext.].

Creating Other ITS Files

When developing your application, you may wish to include other Internet Transaction Server (ITS) files such as:

- Language resources
- Multipurpose Internet Mail Extension (MIME) files

Although these files enhance the performance and appearance of your application, they are not mandatory.

Language Resources

To keep the ITS service that runs your application language independent, you can create language resources.

Language resource files contain all the texts required to run an ITS service in a particular language. For each text, you specify a placeholder and the appropriate text in the language resource file. You then use the placeholders in HTML^{Business} templates instead of the texts themselves. This means that you can create a single set of templates for all languages, rather than a separate set for each language. At runtime, the ITS detects the placeholders in each template and replaces them with the correct language text from the language resource file.

Not all ITS services use language resources. Some HTML^{Business} templates contain hard-coded language texts and are therefore language-dependent. To keep your ITS services as flexible as possible, you should try to use language resources whenever possible.

Unless you need to work with language-specific HTML^{Business} templates, you should create language resource files for the languages in which you are developing your application.

MIME Files

MIME files include any graphics, images, and video elements that may serve to to customize or enhance your application.

Procedure

For further information about creating language resources and MIME files, see the SAP@Web Studio [Ext.] documentation.

Result

When you have created all the ITS files you need for your application, publish [Page 34] them to an ITS site.

Publishing the ITS Service

When you have created all the Internet Transaction Server (ITS) files that make up your application, you can publish the service to the ITS site [Page 25] you defined earlier.

If you follow the procedure below, you can run the application immediately.

Procedure

To publish the ITS files that make up your application to an ITS site, and test it immediately:

1. Choose *Project* → *Set Active Service* and select the relevant service.

   This makes the service you select the currently active service and activates the *Publish*, *Build*, and *Go* buttons in the publish toolbar immediately above the work area.

2. Choose *Options* → *Studio Properties* and select the *Web Browser* tab.
3. Select *Open Browser window in Studio* (if not already selected) and choose *OK*.

   This allows you to run the application directly from the SAP@Web Studio.

4. Choose *Go*.

   When you do this, the SAP@Web Studio:

   - Publishes the service to the currently active ITS site.
   - Opens a Web browser.
   - Starts the application.

Result

When the application starts, the SAP@Web Studio:

- Displays the initial template of your application in the Web browser window.

  The initial template is the template you defined in the **`~initialTemplate`** parameter of your application's service file.

  The initial template of the example application is **`auction.html`**.

- Displays a brief log in the message area immediately below the work area.

If you don't want to run the application, you can also use the *Build* and *Publish* buttons in the SAP@Web Studio's publish toolbar:

- To publish the entire service to the currently active site, choose *Build*.
- To publish individual files to specified sites, choose *Publish*.

Now add the files to ITS source control [Page 35] in the R/3 System.

For further information about publishing and testing ITS files, see ITS Service Publishing [Ext.] in the SAP@Web Studio documentation.

Adding Files to ITS Source Control

When you have published and tested the service, you can check all the Internet Transaction Server (ITS) files into ITS source control in the R/3 System. This protects against simultaneous modification of files, because all users have read access to files, but only a single user is allowed to check files out and modify them.

Procedure

To add the files to ITS source control:

1. Connect to the R/3 System where you want to add the files.
   a. Choose *Tools* → *Source Control* → *Connect to R/3*.
   b. Select an R/3 System and choose *OK*.
   c. If necessary, enter logon information such as client, user, password, and language.
   d. Choose *OK*.
2. Add the files.
   a. In the SAP@Web Studio project workspace, select the file(s) you want to add.
   b. Click the right mouse button and choose *Add to Source Control*.
   c. Check the files you want and choose *OK*.
3. Assign the files to a change request.

   If you fail to do this, you will not be able to check the files out again for modification.

   a. Choose *Tools* → *Web development* → *Web object administration*.
   b. In the *Service name* field, enter the service name for the files you have just added.
   c. Choose *Execute*.
   d. Select the service and choose *Transport*.
   e. Enter a valid change request number or create a new one in the Workbench Orgranizer.

Result

The service is assigned to a change request in the R/3 System. You can now manage the files in ITS source control. You can also check the files out, modify them, and check them back in again.

At subsequent check-ins, the R/3 System remembers the relevant change request and replaces old versions of files with new ones.

If you release a change request for transport, you must create a new change request. If you fail to do this, you cannot check the files out again.

For further information about managing versions of ITS files in ITS Source Control, including more details about check-in and check-out procedures, see ITS Source Control [Ext.] in the SAP@Web Studio documentation.

Debugging Flow File Applications

Use

To debug Internet applications that use flow files, you must enable debugging for the appropriate ITS instance, set up the ITS debugger, enable RFC debugging in the service file, and set breakpoints in the ABAP code.

Remember that the RFC connection is stateless, so the debug session will close as soon as you finish executing the ABAP program of interest.

Procedure

To debug your application, perform the following steps:

1. Enable debugging for the appropriate ITS instance (as described in Enabling and Disabling Debugging [Ext.]).

   You (or your system administrator) can do this in ITS Administration.

2. Set up the ITS debugger by creating a logon to an R/3 instance that matches your ITS application server:
   a. In the SAP Logon box, choose *New*
   b. Enter values in the following fields:
      - *Description*

        Use any meaningful name.
      - *Application Server*

        This should be the same of your ITS server.
      - *System Number*

        The default is 0.
   c. Choose *OK*.

      A new entry is created in the SAP Logon box.

3. In the service file of your application, set the **`~rfcDebugging`** parameter to 1.
4. Set breakpoints as required in the ABAP code.
5. Run the application, as described in Flow File Application Tutorial: Example Application [Page 10].

Result

The application is interrupted where you set your breakpoints.

Appendix A: Function Modules Used in Example Application

The business logic required to run the example application is defined in a set of remote-enabled function modules in the function group AUCTION_FUNCTIONS.

The following table lists the function modules in the function group AUCTION_FUNCTIONS, and provides a brief description of each:

Function Module	Description
AUCTION_BROWSE_ITEMS_BY_PRICE	Gets items based on price range.
AUCTION_CREATE_BID	Creates bid for an item.
AUCTION_CREATE_CATALOG	Creates catalog.
AUCTION_CREATE_CATALOG_HIER	Creates catalog hierarchy.
AUCTION_CREATE_CREDIT_CARD	Creates credit card information.
AUCTION_CREATE_ITEM	Creates item in catalog(s).
AUCTION_CREATE_ITEMS_HIER	Creates item under catalog(s).
AUCTION_CREATE_USER	Registers a user.
AUCTION_DISPLAY_BIDDERS	Displays list of registered bidders.
AUCTION_DISPLAY_CATEGORIES	Displays categories.
AUCTION_DISPLAY_CATEGORY_OBJS	Displays sub-categories and items in a category.
AUCTION_DISPLAY_ITEM	Displays item details, highest bid, and seller information.
AUCTION_DISPLAY_ITEMS	Displays items in a category.
AUCTION_DISPLAY_ITEM_HISTORY	Displays bidding history for an item.
AUCTION_DISPLAY_USER	Displays information about a registered user.
AUCTION_GET_PAYMENT_INFO	Gets credit card information.
AUCTION_MODIFY_ITEM	Modifies an existing item.
AUCTION_MODIFY_PAYMENT_INFO	Modifies a user's credit card information.
AUCTION_MODIFY_USER	Modifies user information.
AUCTION_MODIFY_USER_ADMIN_USE	Administrator function to modify user information.
AUCTION_PURGE_ITEMS	Purges sold items.
AUCTION_SEARCH_ITEMS_BY_BEMAIL	Finds items bid on by bidder email address.
AUCTION_SEARCH_ITEMS_BY_BIDDER	Finds items that a user has bid on by bidder ID.
AUCTION_SEARCH_ITEMS_BY_SELLER	Finds items for sale by seller ID.
AUCTION_SEARCH_ITEMS_BY_SEMAIL	Finds items for sale by seller email address.
AUCTION_VALIDATE_ADMIN_PSWD	Validates administrator password.

AUCTION_VALIDATE_USER_PSWD	Validates registered user.

Appendix B: ITS Files Used in Example Application

The ITS files required to run the example application include:

- A service file that contains the parameters to drive the application.
- A set of HTML[Business] templates that define the presentation.
- A set of flow files that define the dialog flow.

  There is one flow file associated with each HTML[Business] template that requires a dialog flow definition.

Service File

The service file is called **`auctionnew.srvc`**.

HTML[Business] Templates and Flow Files

The following table lists the HTML[Business] templates and associated flow files used in the example application. Not all templates have flow files.

HTML[Business] Template	Flow File
`aboutus.html`	
`admin_modify_users.html`	`admin_modify_users.flow`
`administration.html`	`administration.flow`
`auction.html`	`auction.flow`
`auth_failed.html`	
`authenticate.html`	`authenticate.flow`
`bid_successful.html`	`bid_successful.flow`
`browse_items_by_price.html`	
`create_bid.html`	`create_bid.flow`
`create_catalog.html`	`create_catalog.flow`
`create_category_hierarchy.html`	`create_category_hierarchy.flow`
`create_category_hierarchy_top.html`	`create_category_hierarchy_top.flow`
`create_item.html`	`create_item.flow`
`create_item_hierarchy.html`	`create_item_hierarchy.flow`
`create_item_hierarchy_top.html`	`create_item_hierarchy_top.flow`
`create_user.html`	`create_user.flow`
`credit_card_info.html`	`credit_card_info.flow`
`credit_card_info1.html`	`credit_card_info1.flow`
`display_catalog.html`	`display_catalog.flow`
`display_category_objects.html`	`display_category_objects.flow`

display_credit_card_detail.html	display_credit_card_detail.flow
display_credit_card_info.html	display_credit_card_info.flow
display_item.html	display_item.flow
display_item_history.html	display_item_history.flow
display_items.html	display_items.flow
display_items_bidderemail.html	
display_items_bidderid.html	display_items_bidderid.flow
display_items_sellerid.html	display_items_sellerid.flow
display_users.html	display_users.flow
displayarea.html	
empty.html	
enter_catalog.html	enter_catalog.flow
failed.html	
header.html	header.flow
help.html	
invaliduser.html	
modify_item.html	modify_item.flow
modify_user.html	modify_user.flow
modify_user_template.html	modify_user_template.flow
no_bids_made.html	
price_search.html	price_search.flow
pricefailed.html	
purgecompletedsuccessfully.html	
registration.html	registration.flow
search.html	search.flow
select.html	select.flow
select_bidderemail.html	select_bidderemail.flow
select_bidderid.html	select_bidderid.flow
select_item.html	select_item.flow
select_seller.html	select_seller.flow
select_selleremail.html	select_selleremail.flow
select_sellerid.html	select_sellerid.flow
select_user.html	select_user.flow
services.html	services.flow

start.html	start.flow
suggestions.html	
validate_admin_password.html	validate_admin_password.flow
validate_password.html	validate_password.flow