DSP : Dynamic Server Pages

* JSP uses HTML + Java classes to create web pages.
* DSP uses HTML + webMethods tag library code to create web pages.

A DSP is an ordinary HTML document that contains additional tags enclosed in % symbols (i.e tags e.g., %loop%).

When a client requests a DSP, Integration Server executes the action specified by the tag and substitutes the result of that action in the document it returns to the client.

When Integration Server returns a DSP, it always sets the value of the HTTP content-type header field to text/html.

Therefore, a DSP should only contain HTML content and should only be used by clients that recognize and accept this content type.

DSPs are used to build browser-based clients/user-interface than can be built by directly invoking a service from a browser.

Ex - webMethods Integration Server Administrator

DSP files location for IS - Integration Server\_directory\packages\WmRoot\pub directory

DSPs advantages over directly invoking a service with a URL:

* They conceal the INVOKE mechanism and the name of the service from the user.
* They give you the flexibility to change the name of a service or replace one service with another without changing the way in which the end user invokes the service.
* (The user always invokes the same DSP, whose contents you can change as needed.)
* They can easily be updated and extended.
* They allow you to execute multiple services via a single request.
* They allow you to conditionally execute a service based on run-time input.

While Creating DSPs make sure the document that you create resolves into a valid HTML document.

Publishing DSPs:

* To run a DSP, you must publish it on an Integration Server
* Place the DSP file in the pub directory of the required package.
  + To publish a DSP in the orders package,
    - Integration Server\_directory\packages\**orders\pub**
  + To publish a DSP in the status subdirectory within the orders package:
    - Integration Server\_directory\packages\**orders\pub\status**

Requesting DSPs

To process a DSP, you request it from a browser using the following URL format:

http://hostName:portNum/packageName/fileName .dsp

* http://rubicon:5555/showorders.dsp
  + retrieves showorders.dsp from the Default package on a server named rubicon
* http://rubicon:5555/ORDER\_TRAK/showorders.dsp
  + retrieves showorders.dsp from a package named ORDER\_TRAK on a server named rubicon
* http://rubicon:5555/ORDER\_TRAK/STATUS/showorders.dsp
  + retrieves showorders.dsp from the STATUS subdirectory in a package named ORDER\_TRAK on a server named rubicon

When you use DSPs to build a user interface, you will often invoke DSPs from HTML forms and links

<BODY>

<A HREF=/ORDER\_TRAK/showorders.dsp>Show Orders</A>

</BODY>

DSP Tags

To develop a DSP, you embed DSP tags where you want the results of the tags to appear.

Begin...End Constructs

Many DSP tags have both beginning and ending elements.

Ex: %loop%…%end% construct.

the %end% element always ends the current construct

To make your DSP easier to read, you can append a suffix to the %end% element of any construct to visually associate it with its beginning element.

Ex: %loop%…%endloop% construct.

only the first three characters of an %endloop% element are significant.

The DSP Processor ignores any suffixes that you add.

The %end% element always ends the current construct

%invoke% Tag

%invoke% tag to invoke a service in a DSP. When this tag is processed,

Integration Server executes the specified service at the point where the tag appears and

returns the results of the service to the DSP processor.

Basic format

%invoke serviceName%

Block of Code

[%onerror%

Block of Code ]

%end%

Conditional Blocks

%ifvar% Tag

The %ifvar% tag is similar to an “if…then…else” expression

Basic format

%ifvar variableName%

Block of Code

[%else%

Block of Code ]

%end%

variableName specifies the name of the variable that will be evaluated at run time.

Options applied to the %ifvar% tag

* isnull
  + %ifvar backItems -isnull%.
* Notempty
  + %ifvar supplierInfo/email -notempty%
* equals(‘anyString’)
  + %ifvar carrier equals (‘FedEx’)%
* vequals(refVariable)
  + %ifvar supplierInfo/state vequals(buyerInfo/state)%.
* matches(‘regular\_exp’)
  + %ifvar carrier matches(‘UPS\*’)%.

Uses: %ifvar variableName –options [option value]%

**%loop% tag**

%loop% tag is used to repeat a block of code once

* for each element in a specified array (String list or document list) or
* For each key in a document.

%loop [Variable] [option option option...] %

Block of Code

%end%

Variable specifies the name of the array variable over which you want the enclosed section of code to iterate.

A Variable is

* String list - Each String in the list.
* Document list - Each document in the document list.
* Document - Each key in the document.
  + When you use %loop% to process the elements of a document, you must also use the –struct option in the %loop% tag.

Options: You can specify multiple options, to do this separate the options with spaces.

* Struct: Specifies that Variable is a document and instructs the server to apply the loop **once** to **each key** in that document.
  + i.e You may optionally omit Variable and specify the –struct option to loop over each element in the current scope.
  + When you use the –struct option, you can use the $key variable to retrieve the name of each element in the document.
* eol: Ends the body of the loop at the next end-of-line (EOL) characterin the code.
  + When you use –eol, you can omit the %end% tag.
* $index: Returns the current index number in an array.
  + Use it within a loop to obtain the index number upon which the loop is acting during each iteration.

**Output templates**

Output templates allow you to insert output values from a service into a document that you define.

They contain special “tags” that webMethods Integration Server processes before passing the document back to the client.

Output templates are used

* To customize the HTML page that a service returns.
  + <BODY><P>Contact information for account %value AccountNum% is:</P>
* to format your results as an XML document or
  + <ACCOUNT\_INFO>

<ACCOUNT\_NAME>%value AccountNum%</ACCOUNT\_NAME>

</ACCOUNT\_INFO>

* a comma-delimited record
  + %value AccountNum%,%value CompanyName%
* To return WML or HDML content to wireless devices.

A template is simply a String containing text and one or more tags.

Tags are special commands**, enclosed in % symbols**, that cause webMethods Integration Server to perform a specified action.

When Does the Server Use an Output Template?

The server applies output templates to the results of services that are invoked by HTTP, FTP, or SMTP clients.

You can also arbitrarily apply output templates to the pipeline using the built-in services in the pub.report folder.

If a service has an output template assigned to it, the server automatically applies the template to the results of the service (i.e., the contents of the pipeline) any time that service is externally invoked by an HTTP client.

Guidelines for Using output template

* For html clients, template contains a <meta> tag that sets the Content Type to “text/html”.
* For html clients, template contains a <meta> tag that sets the Content Type to “text/xml”.
* For e-mail clients, the server can apply either XML- or HTML-based output templates.
* For FTP clients, the server will only apply XML-based output templates.
  + If an HTML-based output template is assigned to the service, it is ignored.
* For Internet-enabled wireless device (WML client), template contains a <meta> tag that sets the Content Type to “text/vnd.wap.wml”.
* For Internet-enabled wireless device (HDML client), template contains a <meta> tag that sets the Content Type to “text/xhdml”.

You can arbitrarily apply output templates to the contents of the pipeline using the built-in services in the pub.report folder.

**Creating an Output Template**

To create the contents of an output template file, type all literal text exactly as you want it to appear in the result and then embed any of the DSP tags where you want the server to execute them at run time.

You must give the output template file a name that is unique within the package.

If you plan to assign the output template to a service, the template file must reside in the \templates directory of the package where the service resides.

Integration\_Server\_directory\packages\packageName \templates.

**Assigning an Output Template to a Service**

Use of an output template to format service output is optional. You assign an output template to a service using the Properties view in Designer.

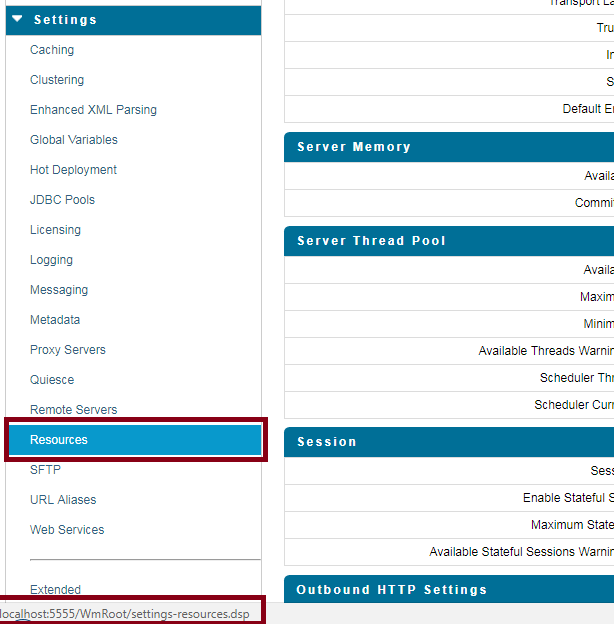
* A service can have at most one output template assigned to it at a time.
* You can assign the same output template to more than one service.
* You can dynamically change the output template assignment at run time.
* the output template must reside in the Integration Server\_directory\packages\packageName \templates directory,
  + Where packageName is the package in which the service is located.
* If the template file has a file extension other than .html, rename the file extension as “.html” so that Designer will recognize its contents.

For steps for assigning an output template to a service, see webMethods Service Development Help.

Securing [against cross site scripting (XSS) attacks] Documents Created from Output Templates

* set the watt.core.template.enableFilterHtml parameter to true (default)
  + so that the output from a %value Variable% tag, including XML and JavaScript, is HTML encoded.
* if you do not want Integration Server to HTML encode the output from a %value Variable% tag, use the encode(none)
  + (%value Variable encode(none)%)
* If you do not want Integration Server to HTML encode the output from any %value Variable% tag in all documents set the watt.core.template.enableFilterHtml parameter to false.

**DEP-Flow**



When you bring your mouse pointer to Resources it displays settings-resources.dsp

This implies clicking on resource will bring you to the page settings-resources.dsp

Dsp interact with java code as service.

Now look snippet of settings-docstores-edit.dsp file

<FORM NAME="editform" ACTION="settings-docstores.dsp" METHOD="POST" onsubmit="return confirmEdit('%value isBrokerConfigured encode(javascript)%')">

<input type**="hidden"** name="action" value="edit" />

<TR>

<TD class="heading" colspan=3>Default Document Store</TD>

</TR>

.

.

.

<TR>

<TD class="action" colspan=3>

<input type="**submit**" name="submit" value="Save Changes"/>

</TD>

</TR>

</FORM>

Now notice few important points from above DSP

* ACTION="settings-docstores.dsp"
* A hidden variable action="edit"
* input type="submit

If we combine all these three, it implies on click of button "Save Changes", control goes to another DSP file settings-docstores.dsp and before going there it sets hidden variable action as edit.

Now have a look into settings-docstores.dsp file

* It is a dsp file without Form/Action property.
* It has normal tags to form UI for docstore
* It has some if condition (on value of action) which decide which java code (web service) to be called.

<BODY onLoad="setNavigation('settings-docstores.dsp', '/WmRoot/doc/OnlineHelp/wwhelp.htm?context=is\_help&topic=IS\_Settings\_DocStoresScrn');">

<TABLE width="100%">

<!-- update the Message Store Settings -->

%ifvar action equals('edit')%

%invoke wm.server.dispatcher.adminui:setDocStoreSettings%

%ifvar message -notempty%

<script>writeMessage("%value message encode(html\_javascript)%");</script>

%else%

<script>writeMessage('Settings changed successfully');</script>

%endif%

%onerror%

<script>writeMessage("%value errorMessage

encode(html\_javascript)%");</script>

%endinvoke%

%endif%

<!-- update the XA Recovery Settings -->

%ifvar action equals('editXA')%

%invoke wm.server. xarecovery:setXARecoveryStoreSettings%

:

:

These if controls are executed as soon as this page is loaded. (Need to check this assumption)

How the value is returned from java and handled in Dsp.

Before loading the DSP page all invoke (i.e. public service) are executed. And because of this pipeline data (note services are called without param and they work on pipeline data) are modified or may be new are added. These pipeline data an be accessed in DSP using %value variable%, where variable is part of pipeline data. (Need to check this assumption)

How to display error msg/exceptions in DSP, thrown from java code.