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.