**Resources in web.xml**

The Web application deployment descriptor specifies a series of resources that are instantiated and configured based on the settings you provide. In web.xml, you are required to include the following resources:

* The DTD declaration and Web application name in your web.xml as you would for any other J2EE application
* NucleusServlet, which is the servlet responsible for running Nucleus as a servlet
* A resource (PageFilter or DynamoProxyServlet) that starts the request-handling pipeline
* A <distributable/> tag, in order to enable session failover when running in a cluster. This tag is added automatically when you assemble your EAR using the–distributable flag.

In additional to these required resources, you might want to browse the [Optional Resources](http://docs.oracle.com/cd/E23095_01/Platform.93/ATGProgGuide/html/s0703optionalresources01.html) section later in this chapter for additional resources provided by the ATG platform. Most applications require site-specific resources.

## Request Processing

A request processed by an application server follows this path. The process described here assumes that you configured your Web application to use PageFilter andDynamoProxyServlet as demonstrated in atg\_bootstrap.war.

 The pair does not exist, so PageFilter starts the DAF servlet pipeline by calling DynamoHandler, the first servlet in the pipeline. The DAF servlet pipeline processes through a series of servlets that modify the request and response by extracting path information and providing session, user, and security information. The last servlet in the pipeline is TailPipelineServlet.

A pipeline is an execution mechanism that allows for modular code execution. Oracle ATG Web Commerce uses pipelines to execute tasks such as loading, saving, and checking out Orders .The PipelineManager implements the pipeline execution mechanism.

There are two request-handling pipelines used by Dynamo.

• DAF Servlet Pipeline - It is used to handle the JSP request.

• DAS Servlet pipeline - It is used to handle JHTML request. Because JHTML is a proprietary language, it relies on the page compiler provided in the DAS servlet pipeline to generate JHTML into a servlet that’s rendered as HTML by the application server.

And also there is something called commercePipeline which takes care of order processing.

Request-Handling pipelines and commerce pipelines works in different ways.

**DAS/DAF(ie., request pipelines)**

It's a configuration defined with a series of servlets executed in a sequence on basis of each servlet's output. One of Dynamo's most important tasks is handling HTTP requests. In handling these requests, Dynamo uses session tracking, page compilation, Java Server Pages, and other powerful extensions to the basic Web server model.Request handling can usually be broken down into a series of independent steps. Each step may depend on additional information being available about the request, so order does matter. However, the individual steps are separable. For example, a typical request might go through these steps:

1) Compare the request URI against a list of restricted directories, to make sure that the user has permission to access the specified directory.

2) Translate the request URI into a real file name, taking "index" files into account when the file name refers to a directory.

3) Given the file name's extension, determine the MIME type of the file.

4) From the MIME type, dispatch the request to the appropriate handler.

So DAF/DAS pipelines comes into picture when there is a request. In atg\_bootstrap.war web.xml has the information about the server startup.

When the server starts NucleusServlet.java gets loaded in the app server. This class initializes nucleus and other components and then adds all of them to nucleus namespace. And when a web application is accessed (DynAdmin,CRS,MotopriseJSP), nucleus routes the flow to either daf/das pipeline. If the application MIME type is Jhtml then das pipeline processes the request further. It is routed to DynamoProxyServlet class where it does the further processing by calling list of servlets. And if it is .jsp then Daf pipeline handles the further requests by calling PageFilter class.The reason for using a filter but not a servlet to invoke DAF pipeline is:

JSP pages and fragments are handled by the application server meaning that JBoss, WebLogic , WebSphere is the one responsible for compiling and executing the resulting page code. The best way to hook into this process is by using a Filter. For JHTML pages, that's a different story since the app server (not all app servers)can't parse and compile the pages. A servlet is used to redirect the request down the DAS pipeline where the page can parsed and executed by the ATG page compilationmechanism.

**In case of Commerce Pipeline:**

Pipeline Manager implements commerce pipeline functionality by reading a pipeline definition file ie., commercepipeline.xml. When application is deployed ,Nucleus initializes pricing engine, where OrderManager initializes pipelineManager. OrderManager.processOrder method invokes the pipeline chains in commercepipeline.xm. A pipeline chain would have processors which are simple java classes doing small operations .This xml can be extended by adding a custom processor. But in cases where a single processor need to be called, call runProcess method of pipelineManger by passing processorchaninId .

Extending DAF/DAS pipeline and commerce pipeline are not same

we can create our own custom servlets to put it in DAF/DAS pipeline .

extend you own servlet class wither with **PipelineableServletImpl** or **InsertableServletImpl** and re-write the service method depending on what you want to do.Further details are widely available on the internet :)

and coming to commerce pipeline

Commerce pipeline is defined in an xml file located in /B2CCommerce/config/atg/commerce/commercepipeline.xml.PipeLine manager is responsible load the pipeline definition xml and initialize the pipeline chains. Write your processor class. Custom Processor class should be an implementation of PipelineProcessor.

extend your own class by **PipelineProcessor** and re-write runProcess method.You also have to create respective .properties file for your processor.And then in

**B2CCommerce/config/atg/commerce/commercepipeline.xml**

<pipelinechain name=" lastExistingchain" transaction="TX\_REQUIRED" headlink=" lastExistinglink">

……..

<transition returnvalue="1" link=" sampleDemoLink"/>

</pipelinelink>

<pipelinelink name="sampleDemoLink" transaction="TX\_REQUIRED">

<processor jndi="demo/atg/order/processor/MyProcessor"/>

</pipelinelink>

</pipelinechain>

restart ATG server.

Coming to you other question that if we can create our own pipeline manager Answer is yes. Just create /atg/registry/PipelineRegistry/ .properties file in your local config folder.PipelineRegistry is a service where all pipeline managers are registered

this service has property called pipelineManagers just append your pipeline manager component to this property.if you want to use existing commercePipelineManager class but with different bunch of processors executing one aftr the other.create a definition xml file which looks something like this

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE pipelinemanager

PUBLIC "-//Art Technology Group, Inc.//DTD Dynamo Pipeline Manager//EN"

'http://www.atg.com/dtds/pipelinemanager/pipelinemanager\_1.0.dtd'>

<pipelinemanager>

<!-- This chain updates (saves) an Order to the repository -->

<pipelinechain name="updateOrder" transaction="TX\_REQUIRED" headlink="updateOrderObject">

<pipelinelink name="updateOrderObject" transaction="TX\_MANDATORY">

<processor jndi="/atg/commerce/order/processor/SaveOrderObject"/>

<transition returnvalue="1" link="updateCommerceItemObjects"/>

</pipelinelink>

<pipelinelink name="updateCommerceItemObjects" transaction="TX\_MANDATORY">

<processor jndi="/atg/commerce/order/processor/SaveCommerceItemObjects"/>

<transition returnvalue="1" link="updateShippingGroupObjects"/>

</pipelinelink>

<pipelinelink name="updateShippingGroupObjects" transaction="TX\_MANDATORY">

<processor jndi="/atg/commerce/order/processor/SaveShippingGroupObjects"/>

<transition returnvalue="1" link="updateHandlingInstructionObjects"/>

</pipelinelink>

<pipelinelink name="updateHandlingInstructionObjects" transaction="TX\_MANDATORY">

.......

.......

<pipelinechain name="rejectQuote" transaction="TX\_REQUIRED" headlink="quoteRejection">

<pipelinelink name="quoteRejection" transaction="TX\_MANDATORY">

<processor jndi="/atg/commerce/order/processor/RejectQuote"/>

</pipelinelink>

</pipelinechain>

<!-- This pipeline chain should be called when a requested quote is to be completed -->

<pipelinechain name="completeQuote" transaction="TX\_REQUIRED" headlink="completeQuoteRequest">

<pipelinelink name="completeQuoteRequest" transaction="TX\_MANDATORY">

<!-- this is a dummy processor that should be extended to save quote details -->

<processor jndi="/atg/commerce/order/processor/CompleteQuoteRequest"/>

</pipelinelink>

</pipelinechain>

</pipelinemanager>

here you can mention your custom processors.

As you have registered your new pipelinemanager in pipeline registry.It gets automatically initialized.So if you do any operation in JSP related to your pipeline,in background all the processing gets done.