HTTP session management with WebSphere eXtreme Scale

What this exercise is about

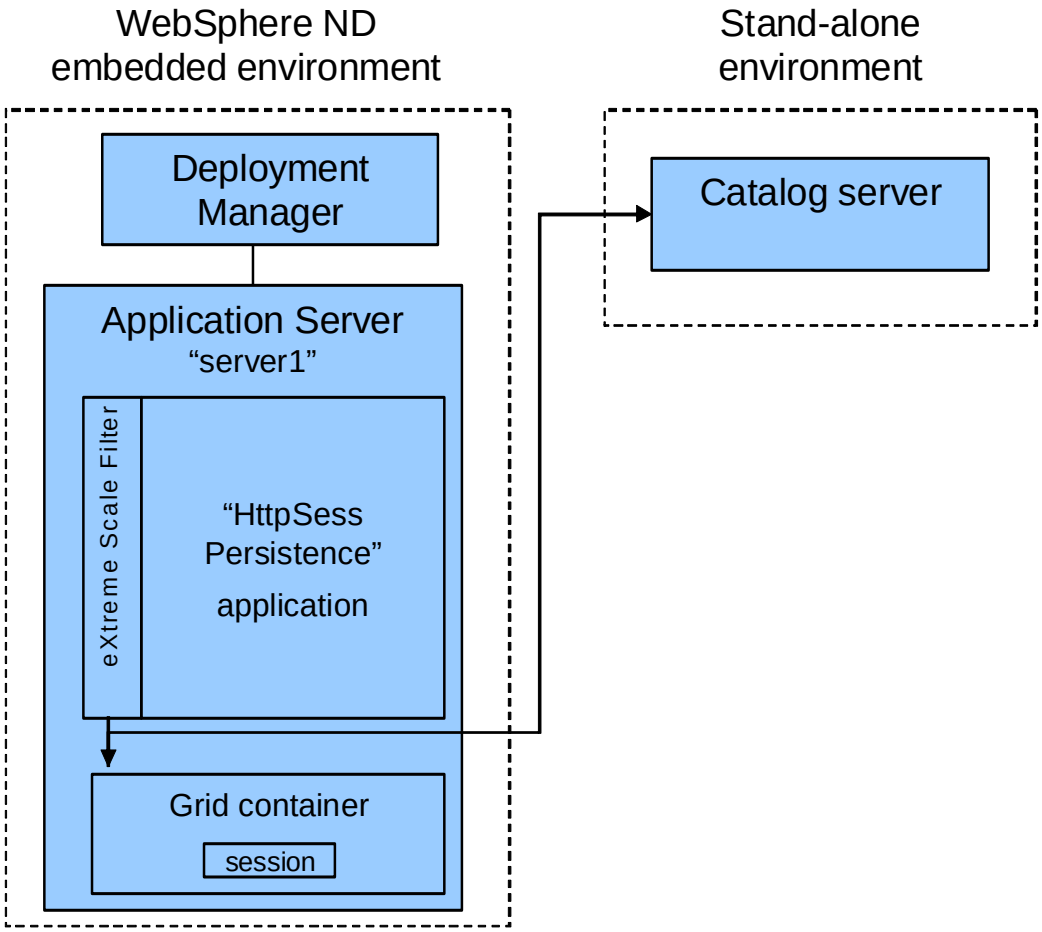
This lab teaches you how to persist HTTP session data into an eXtreme Scale grid. The function demonstrated in this lab will allow for session persistence across multiple cells. In this exercise, one cell will host a grid container inside a WebSphere embedded ObjectGrid server with a shared catalog in a stand-alone ObjectGrid server outside of the WebSphere environment. For simplicity of setup, the stand-alone server is launched from within the WebSphere installation rather than from a separate stand-alone eXtreme Scale installation.

Lab requirements

This lab assumes a configuration similar to this. If your configuration is different you will need to modify some steps.

- The lab requires a single host. This lab uses the host name “hostA”.
- WebSphere Application Server Network Deployment V6.1 or 7.0.
    - ND V7.0 requires fixpack 5, or fixpack 3 with interim fix PK87629
- Profiles for Deployment Manager (Dmgr01) with federated custom node (hostANode01).
- WebSphere eXtreme Scale V7 installed in the WebSphere environment.
    - Dmgr01 and hostANode01 profiles are augmented with WebSphere eXtreme Scale V7.

.



What you should be able to do

At the end of this lab you should be able to:

- Modify an application to .war to persist HTTP session data to an eXtreme Scale grid.
- Configure WebSphere to connect to an external eXtreme Scale catalog server.
- Persist HTTP session data to a grid container embedded in a WebSphere managed server.

Introduction

WebSphere eXtreme Scale provides the ability to manage your HTTP session data. You can override the default session manager in the base application server to provide HTTP session management capabilities for an associated application. The eXtreme Scale session manager will fully manage the life cycle of HTTP sessions that belong to the application.

eXtreme Scale accomplishes this by providing a session filter. This filter intercepts every request to the Web application. Before passing control to the application servlet or JSP, it wraps the

HTTPServletRequest and HTTPServletResponse objects that the application uses to access the request's session state in an eXtreme Scale HTTPSession object. The eXtreme Scale HTTPSession object overrides the default session manager's implementation, so there is no data duplication between the two session managers. The eXtreme Scale session filter normally stores session state in a grid container in the local process. This provides low latency read and write operations when running with asynchronous replication. The filter will ensure that the session data is persisted to the grid and the grid will ensure the data remains highly available.

Exercise instructions

Some instructions in this lab are Windows® operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to run the appropriate commands and use appropriate files (.sh or .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference variable	Windows location	AIX® or UNIX® location
<WAS_HOME>	C:\WebSphere70\AppServer	/usr/WebSphere70/AppServer /opt/WebSphere70/AppServer
<IRAD_HOME>	C:\Program Files\IBM\RSDP\6.0	
<LAB_FILES>	C:\LabFilesXS	/tmp/LabFilesXS
<TEMP>	C:\temp	/tmp

Note for Windows users: When directory locations are passed as parameters to a Java™ program such as EJBdeploy or wsadmin, replace the backslashes with forward slashes to follow the Java convention. For example, replace C:\LabFiles60\ with C:/LabFiles60/

.

Part 1: Install the sample application

You will install a small servlet application which shows session data. The initial application will only store data in memory. Later you will update this servlet to persist session data to an eXtreme Scale grid.

____ 1. Start the WebSphere environment

__ a. Start the Deployment Manager.

1) On **hostA,** open a command prompt.

2) Change directories to **<WAS_HOME>\profiles\dmgr01\bi**n.

3) Enter this command to start the Deployment Manager:

startManager

4) Wait for the deployment manager to start. Verify this message is printed in the Command Prompt window.

```
ADMU3000I: Server dmgr open for e-business; process id is nnnn
```

__ b. Start the HostANode01's Node Agent.

1) In the command prompt window, change directories to **<WAS_HOME>\profiles\HostANode01\bin**

2) Enter these command to start the node agent on the ODR node:

startNode

____ 2. Open the WebSphere administrative console

__ a. Open a Web browser.

__ b. Enter the URL: http://hostA:9060/ibm/console.

__ c. Enter a user ID of your choice and click Log In.

____ 3. Create the ObjectGrid container server – **server1**

__ a. In the administrative console, navigate to *Servers*, expand *Server Types* and click *WebSphere Application Servers*; click "**New**".

__ b. Select **hostANode01**, and type "**server1**" for the server name; click **Next**

__ c. Click **Next** (accepting the defaults) on the remaining pages, until the **Confirm new server** page opens.

__ d. Click **Finish** to create the new server.

__ e. Click **Review** in the messages area.

__ f. On the Save panel, select the check box **Synchronize changes with Nodes**.

__ g. Click **Save**.

__ h. Click **OK** when the synchronization operation completes.

Because the server was defined on a node augmented with WebSphere eXtreme Scale, the new server is automatically enabled for ObjectGrid container functionality.

____ 4. Determine the **WC_defaulthost** value. You will need to know this later in this lab

__ a. In the administrative console, navigate to the list of WebSphere Application Servers, and click **server1**

__ b. Expand **Ports** (at the right of the details panel, under Communications)

__ c. Record the value of **WC_defaulthost**

Note: The first server on a host is typically assigned 9080. This exercise will assume the port value is "9080" in the instructions. You will need to replace "9080" with your actual value.

____ 5. Install the **HttpSessPersistence** Application

__ a. Using the administrative console, navigate to *Applications* → *New Application*.

__ b. Click **New Enterprise Application**.

__ c. Next to Local file system, click **Browse** to open the file

<LAB_FILES>\SessPersistenceLab\HttpSessPersistence.ear

__ d. Click **Next**.

__ e. Click **Step 2: Map modules to servers**.

__ f. From the Clusters and Servers list, select **server1**. From the Module list, select the **HttpSessPersistence** module. Click **Apply**.

__ g. Click **Step 3: Summary**.

__ h. On the Summary panel, click **Finish**.

__ i. Once the installation completes, click **Review**.

__ j. Make sure that **Synchronize changes with nodes** is selected and click **Save**.

__ k. Click **OK** when the sync operation completes.

____ 6. Start the server associated with the **HttpSessPersistence** Application

__ a. In the administrative console, navigate to *Servers*, expand *Server Types* and click **WebSphere Application Servers**.

__ b. Select **server1** and click **Start** to start the server. When the server starts, it will show a green arrow  in the status column.

__ c. Navigate to *Applications*, expand *Application Types*, and click **WebSphere enterprise applications**.

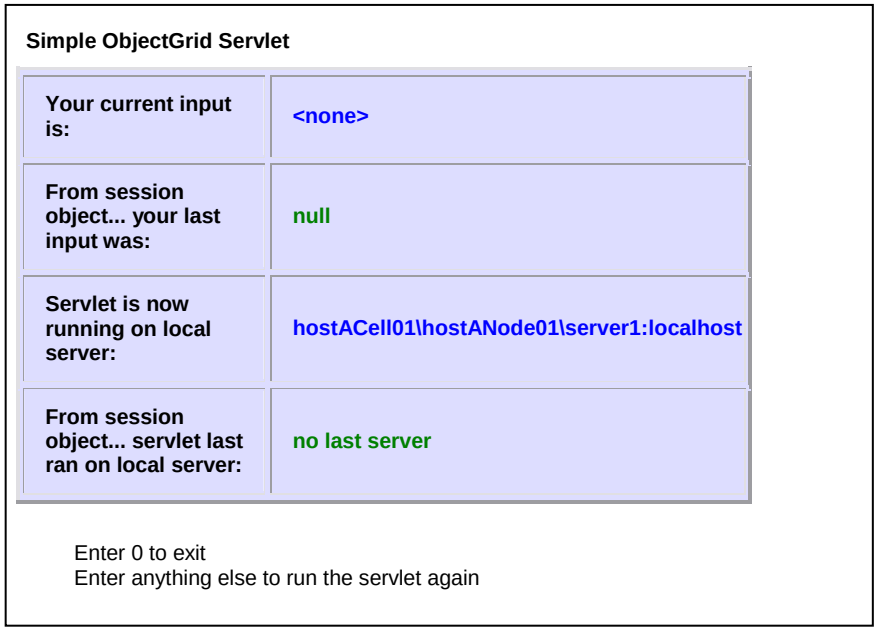
__ d. Verify the **HttpSessPersistence** is running. The application status should show a green arrow  .

__ e. In a Web browser (replacing the port value “9080” with the **WC_defaulthos**t port value you previously obtained) navigate to

http://localhost:9080/HttpSess/Application

This will invoke the application servlet. The application at this point stores session data in local memory. On initial display, there is no previous data from the session.

__ f. The application screen will look similar to this after the initial invocation:



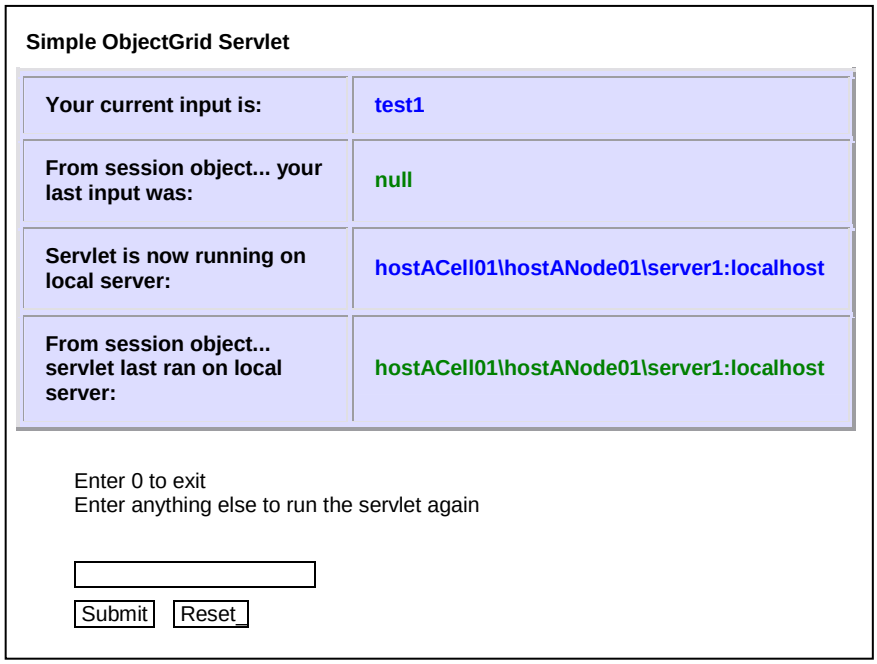
Simple ObjectGrid Servlet

Your current input is:	**<none>**
From session object... your last input was:	**null**
Servlet is now running on local server:	**hostACell01\hostANode01\server1:localhost**
From session object... servlet last ran on local server:	**no last server**

Enter 0 to exit
Enter anything else to run the servlet again

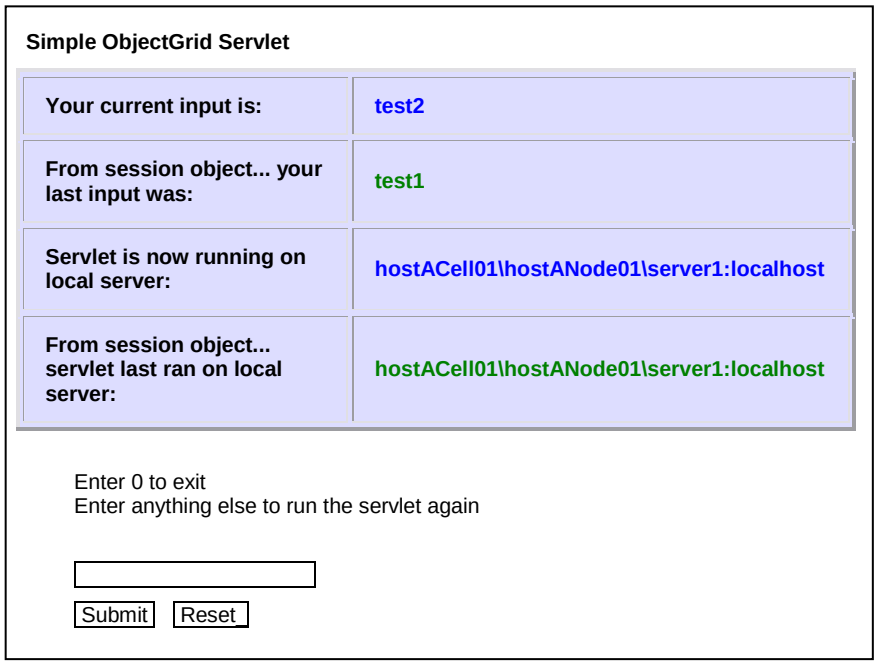
__ g. Type "**test1**" into the entry box and click **Submit**.

The application screen will now look like this:



__ h. Type "**test2**" into the entry box, and click **Submit**.

The application screen will now look like this:



The servlet output shows the current input just processed, the host on which it ran, the input from the previous request and the host on which the previous request ran. The persisted information is currently held only in the application server container's local memory.

You will now change the application so that the information from the session is persisted in an object grid, rather than in memory.

Part 2: Set up catalog server

In this part of the exercise you will configure and start a stand-alone eXtreme Scale catalog server. You will then configure WebSphere Application Server so its eXtreme Scale container server (server1) connects to the catalog server.

____ 1. Configure and start ObjectGrid Catalog Server – **cat1**

__ a. Open a command window and change directory to **<WAS_HOME>\bin**

__ b. Run this command, replacing <hostA> with your fully-qualified host name:

```
startOgServer.bat cat1 -listenerHost <hostA> -listenerPort 28201
-catalogServiceEndpoints cat1:<hostA>:28202:28203
```

You might want to type this command in your own batch file or shell script.

__ c. The final message should state that "the ObjectGrid server cat1 is ready to process requests."

Notes:

startOgServer	The command to start container and catalog servers
"cat1"	The catalog server name you assign
-listenerHost	The host where the catalog server accepts ORB or default bind requests
-listenerPort	The port assigned for ORB or default bind requests
-catalogServiceEndpoints	Ports necessary for server configuration; for catalog servers this represents <client_port> and <peer_port>

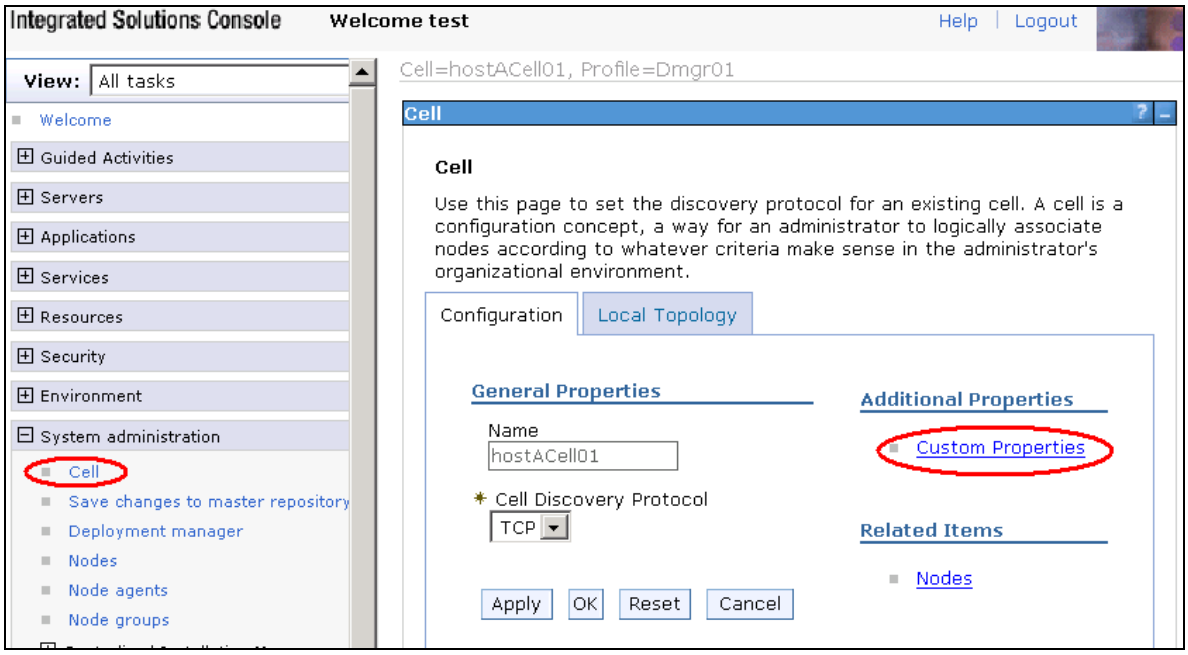
____ 2. Stop ***server1***

__ a. In the administrative console, navigate to *Servers*, expand *Server Types* and click **WebSphere Application Servers**

__ b. Select **server1** and click **Stop**. The server will stop and display  in the status column.

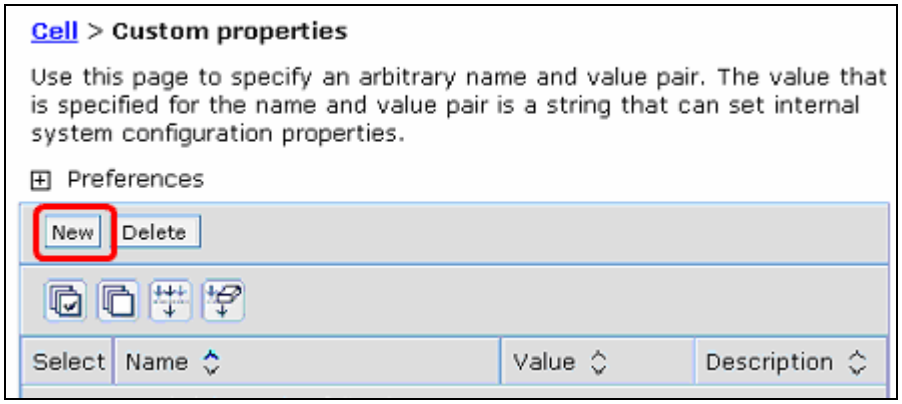
____ 3. Configure the application server ***server1*** (which contains an embedded grid container because the profile was augmented with eXtreme Scale) to connect to the catalog server ***cat1***. In this lab you set a cell custom property `catalog.services.cluster`. However you can set this property at the cluster or individual server level, allowing you to associate multiple independent catalog servers with your WebSphere cell.

__ a. In the WebSphere administrative console, expand *System administration* and click **Cell**. Note **Custom Properties** in the right panel, under "**Additional Properties**".

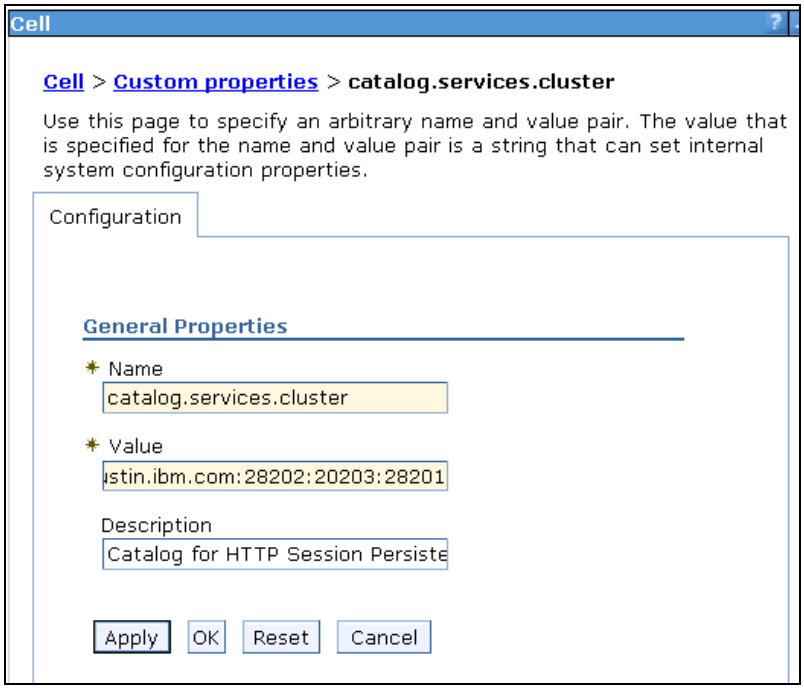


__ b. Click **Custom Properties**, located in Configuration tab under "**Additional properties**".

__ c. Click **New** to create a new custom property.



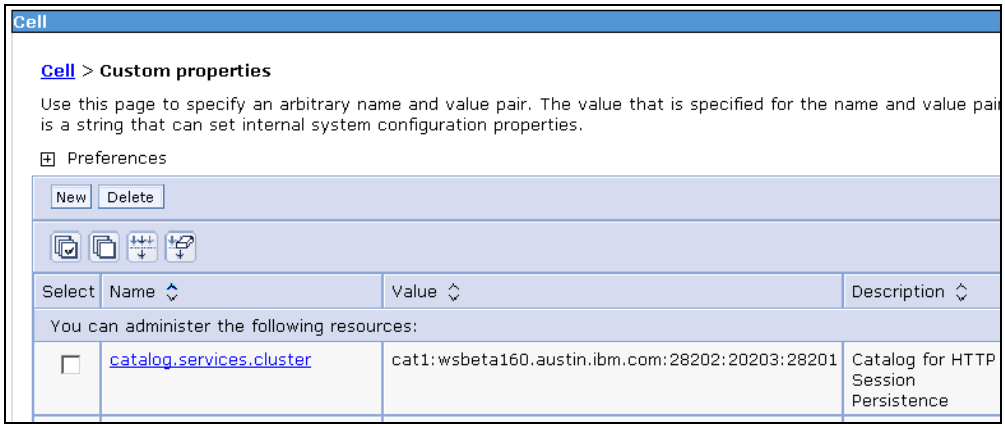
__ d. Type "**catalog.services.cluster** "in the *Name* field. In the *Value* field, type "**cat1:<hostA>:28202:20203:28201**", replacing **<hostA>** with your fully-qualified catalog server host name.



__ e. After reviewing your input, click **OK**.

__ f. Click **Review** in the Messages box, check **Synchronize changes with nodes**, and click **Save**.

__ g. Click **OK** when the sync operation completes. The listing of your new custom property should be similar to this:



The order of the properties is very important in the Value field. The order of specification is:

CatalogName:FullyQualifiedHostName:clientPort:peerPort:listenerPort

The listenerPort must match the listenerPort used for starting the catalog server. The client and peer ports values must be identical to the ports specified in the catalog server start.

Often in actual production configurations, you will have more than one catalog server. To specify an additional catalog server, add a comma at the end of the string and specify the next set of catalog server values

---

By default, eXtreme Scale will start a catalog server instance in the WebSphere deployment manager. If you do not explicitly configure the catalog.services.cluster custom property, any eXtreme Scale client code in a WebSphere managed server (including the HTTP session filter) will connect to the catalog server in the deployment manager.

Though this lab uses an external, stand-alone catalog server, you can also configure the catalog server to run in any WebSphere process.

Part 3: Update the Application Servlet to use WebSphere eXtreme Scale to persist session data

In this part of the lab you will update the application servlet to persist session data to the eXtreme Scale grid instead of in memory. You will use the WebSphere Application Server EARExpander utility to add required eXtreme Scale grid configuration files to the application. You will then re-run the application and show that it is persisting session data in the grid container.

If you prefer to use Eclipse or some other development environment, refer to "Appendix A – Using Eclipse to modify application" on page 21.

____ 1. Open a command window and navigate to <*WAS_HOME*>*/bin*.

____ 2. You will now expand the ear file into a temporary directory. Issue this command:

```
EARExpander.bat -ear <LAB_FILES>\SessPersistenceLab\HttpSessPersistence.ear
-operationDir <LAB_FILES>\SessPersistenceLab\temp
-operation expand expansionflags all
```

____ 3. Copy the two session-related sample configuration files into the expanded ear file – specifically into the META-INF directory within the HttpSess.war expansion

```
copy <WAS_HOME>\optionalLibraries\ObjectGrid\session\samples\objectGrid.xml
<LAB_FILES>\SessPersistenceLab\temp\HttpSess.war\META-INF

copy
<WAS_HOME>\optionalLibraries\ObjectGrid\session\samples\objectGridDeployment.xml
<LAB_FILES>\SessPersistenceLab\temp\HttpSess.war\META-INF
```

Note: after the copy operations are complete, the .../HttpSess.war/META-INF directory should contain *only* these files:

MANIFEST.MF
objectGrid.xml
objectGridDeployment.xml

____ 4. Issue this command to collapse the modified application files and replace the original ear file with the revised HttpSessPersistence.ear file:

```
EARExpander.bat -ear <\LAB_FILES>\SessPersistenceLab\HttpSessPersistence.ear
-operationDir <LAB_FILES>\SessPersistenceLab\temp
-operation collapse expansionflags all
```

____ 5. Splice the servlet application to allow it to use the eXtreme Scale session filter to persist session data.

__ a. Navigate to <*WAS_HOME*>*optionalLibraries**ObjectGrid**session**samples*

__ b. Edit the **splicer.properties** file for the current environment.

Set the catalogHostPort property value to match the value used to start the "**cat1**" catalog server, specifying a fully qualified name for the host name to replace **<hostA>**

catalogHostPort = <hostA>:28201

__ c. Open a command window and "cd" to <*WAS_HOME*>*optionalLibraries**ObjectGrid**session**bin* directory.

__ d. Run the splicer application by using the **addObjectGridFilter** script.

Run the command:

```
addObjectGridFilter.bat
<LAB_FILES>\SessPersistenceLab\HttpSessPersistence.ear
../samples/splicer.properties
```

__ e. Examine the script and confirm that it takes the **HttpSessPersistence.ear** file and the **splicer.properties** file as parameters and updates the **HttpSessPersistence.ear** file.

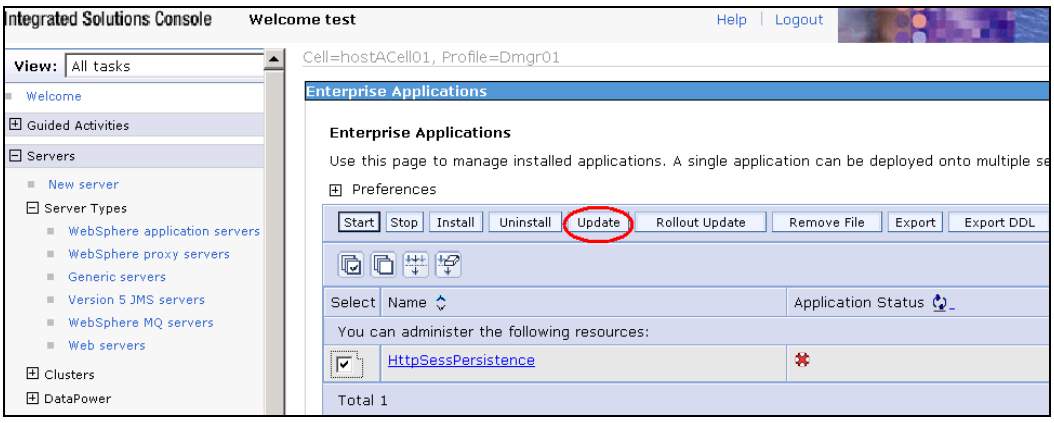
Example output:

```
C:\Program Files\IBM\WebSphere\eXtremeScale\ObjectGrid\session\bin>addObjectGridFilter
   C:\LabFilesXS\SessPersistenceLab\HttpSessPersistence.ear
   ../samples/splicer.properties
CWWSM0023I: Reading properties file: ../samples/splicer.properties
CWWSM0021I: Reading archive: C:\LabFilesXS\SessPersistenceLab\HttpSessPersistence.ear
CWWSM0027I: Processing .war file: HttpSessPersistence
CWWSM0028I: Context parameters are:
CWWSM0029I: Context name: shareSessionsAcrossWebApps Value: false
CWWSM0029I: Context name: sessionIDLength Value: 23
CWWSM0029I: Context name: sessionTableSize Value: 1000
CWWSM0029I: Context name: replicationType Value: synchronous
CWWSM0029I: Context name: defaultSessionTimeout Value: 30
CWWSM0029I: Context name: useURLEncoding Value: false
CWWSM0029I: Context name: affinityManager Value:
                                        com.ibm.ws.httpsession.NoAffinityManager
CWWSM0029I: Context name: catalogHostPort Value: wsbeta160.austin.ibm.com:28201
CWWSM0029I: Context name: objectGridName Value: session
CWWSM0029I: Context name: persistenceMechanism Value: ObjectGridStore
CWWSM0030I: Application splicing completed successfully.
```

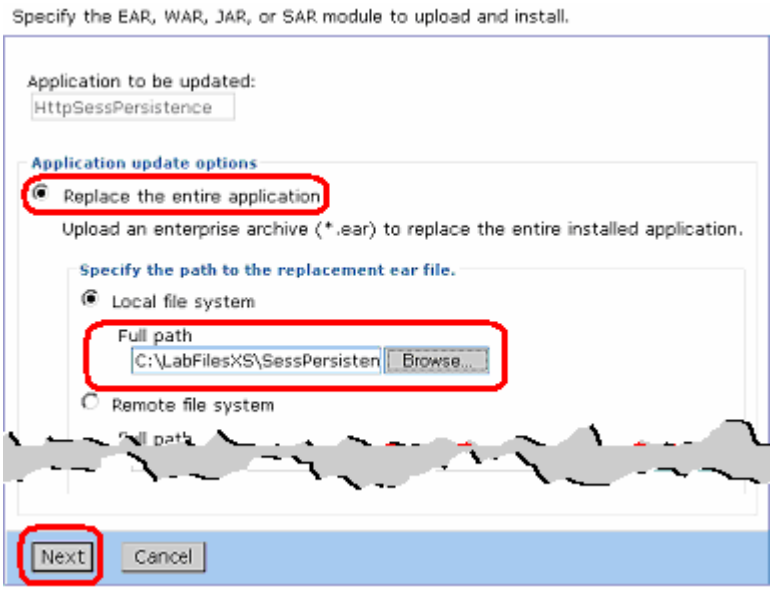
____ 6. Update the servlet application in the WebSphere environment.

__ a. In the administrative console, expand *Applications*, expand *Application types*, and click **WebSphere enterprise applications**

__ b. Select **HttpSessPersistence** and click **Update**



__ c. Select "**Replace the entire application**", browse to the updated **HttpSessPersistence.ear** file, and click **OK**; then click **Next**



__ d. Click **Step 2: Map modules to servers**. Ensure that the replacement ear file is mapped to **server1**. Click **Next**.

__ e. On the Summary panel, click **Finish**.

__ f. Once the installation completes, click **Review**.

__ g. Make sure that **Synchronize changes with nodes** is selected and click **Save**.

__ h. . Click **OK** when the sync operation completes.

____ 7. In the administrative console, expand *Applications*, expand *Application types*, and click **WebSphere enterprise application**. Select **HttpSessPersistenc**e and click **Start**.

____ 8. Run the updated application servlet from the browser and confirm that it is storing session data in the object grid

__ a. Open the Web browser and go to **http://localhost:9080/HttpSess/Application**

Type test data and click **Submit** several times.

__ b. Review the server1 *SystemOut.log* file to confirm that the servlet is indeed persisting session data to the eXtreme Scale grid.

The log file to review is: <*WAS_HOME*>*profiles**hostANode01**logs**server1**SystemOut.log*. Near the end of the file, you should see the message:

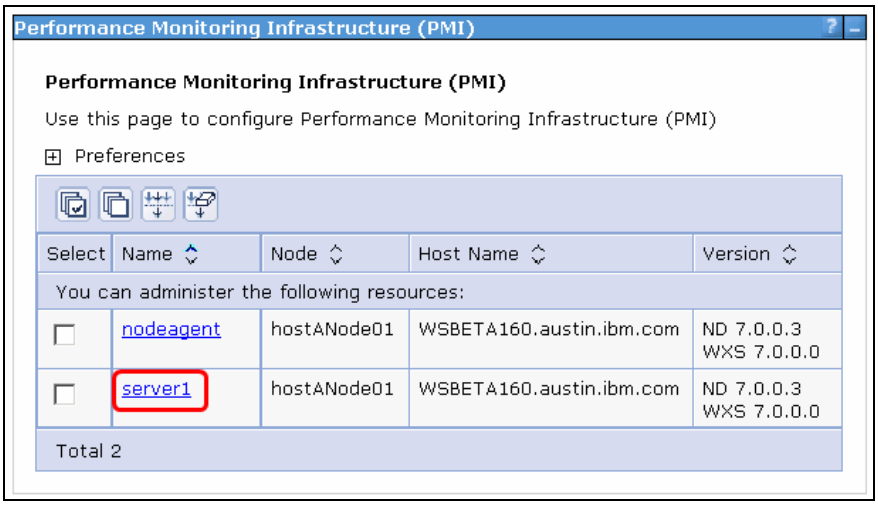
```
"Using the ObjectGrid based Session Manager"
```

Part 4: Using Tivoli PMI Viewer to show eXtreme Scale grid usage to persist session data

In this part of the lab you will use the Performance Monitoring Infrastructure (PMI) of the WebSphere administrative console to show that the session data from the application servlet is being persisted in the ObjectGrid backing maps by monitoring the backing map meta-data for the ObjectGrid. The meta-data shows the number of map entries increase as the application is used.

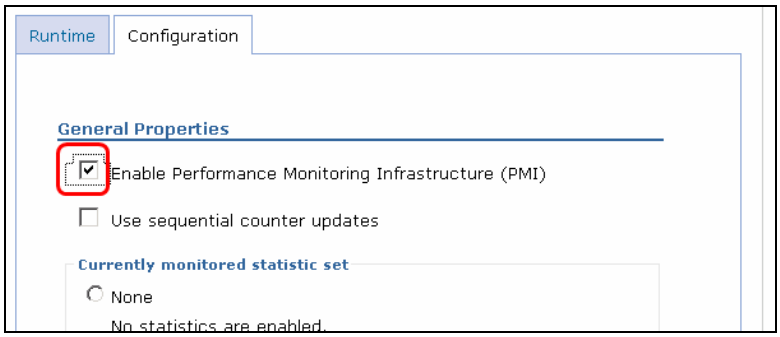
____ 1. Open the WebSphere administrative console and set up monitoring parameters.

__ a. In the administrative console, navigate to *Monitoring and Tuning* → *Performance Monitoring Infrastructure*, and click **server1**



____ 2. Note the two check boxes at the top; and farther down, under "*Currently monitored statistic set*" the radio buttons for statistical monitoring.

__ a. At the top, verify that *Enable Performance Monitoring Infrastructure (PMI)* is selected.



This setting is enabled by default. If the setting is not enabled:

1) Select the check box

2) Click **OK**,

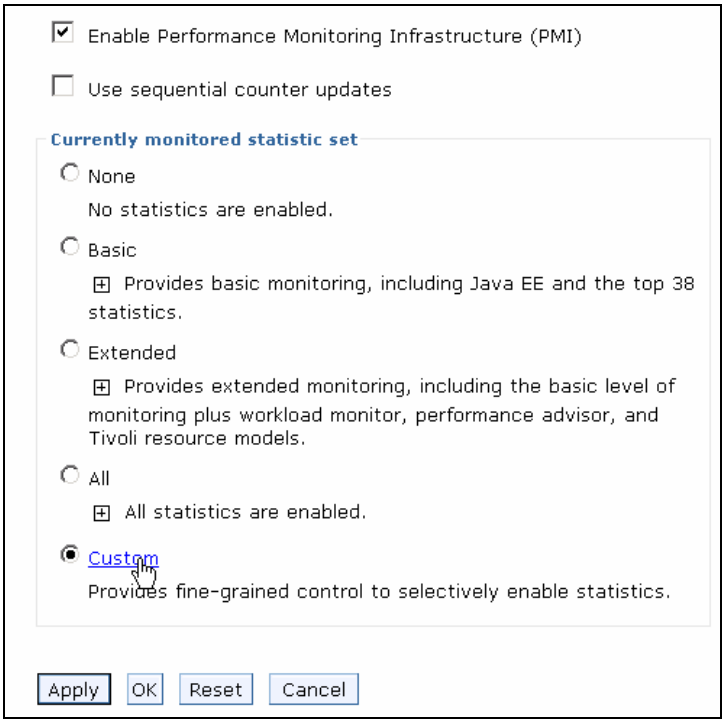
3) Save the setting to the configuration,

4) Restart server1.

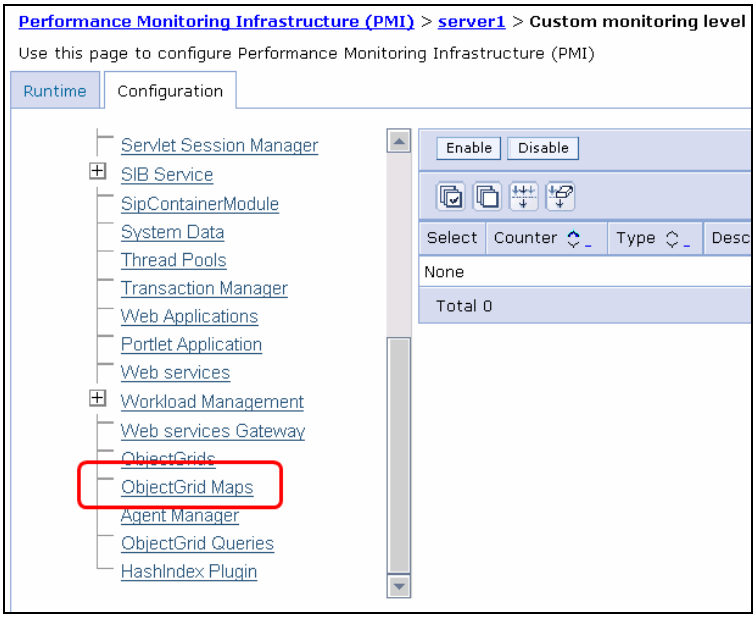
5) Navigate to this screen again.

____ 3. At the bottom portion of the screen, under "*Currently monitored statistic set*", click the underlined word **Custom**.

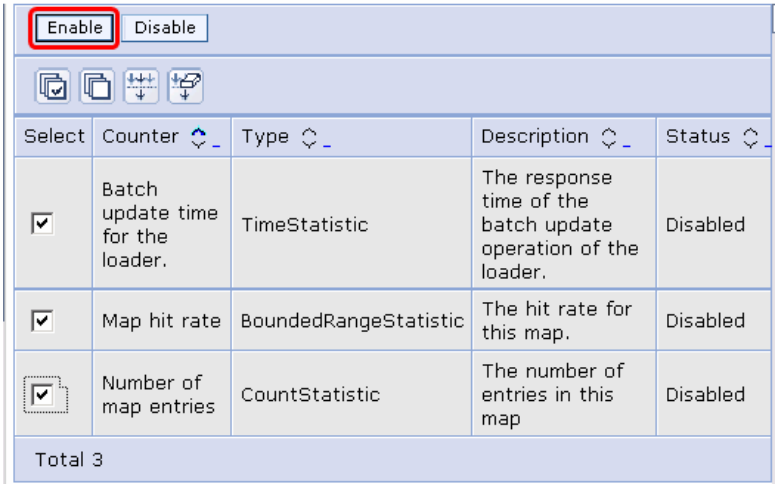
Note: if you just click the radio button at the left rather than the word Custom, you will not see the configuration tree.



____ 4. Within the configuration tree, click **ObjectGrid Maps**.



____ 5. Select the statistics for each counter and click **Enable**.



____ 6. Save the changes.

__ a. Click **Review** in the Messages area.

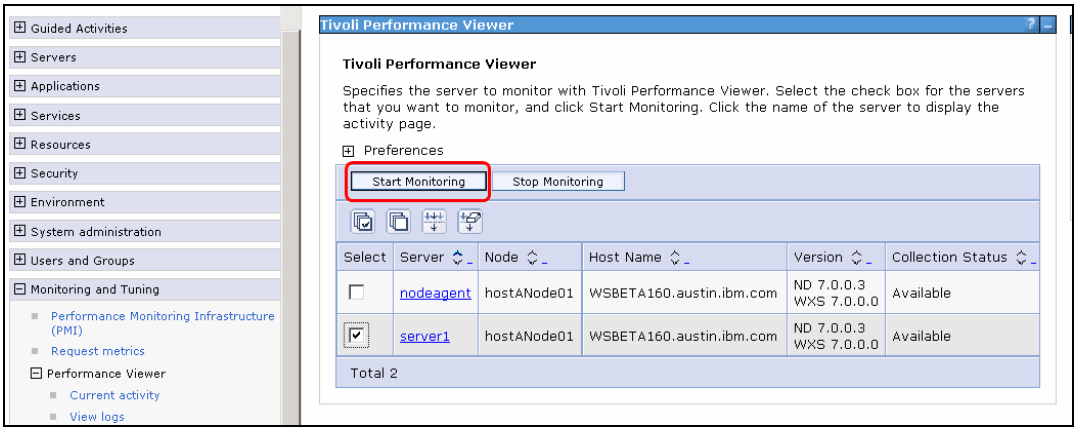
__ b. On the Save panel, make sure that the check box **Synchronize changes with Nodes** is selected.

__ c. Click **Save**.

__ d. Click **OK** when the sync operation completes.

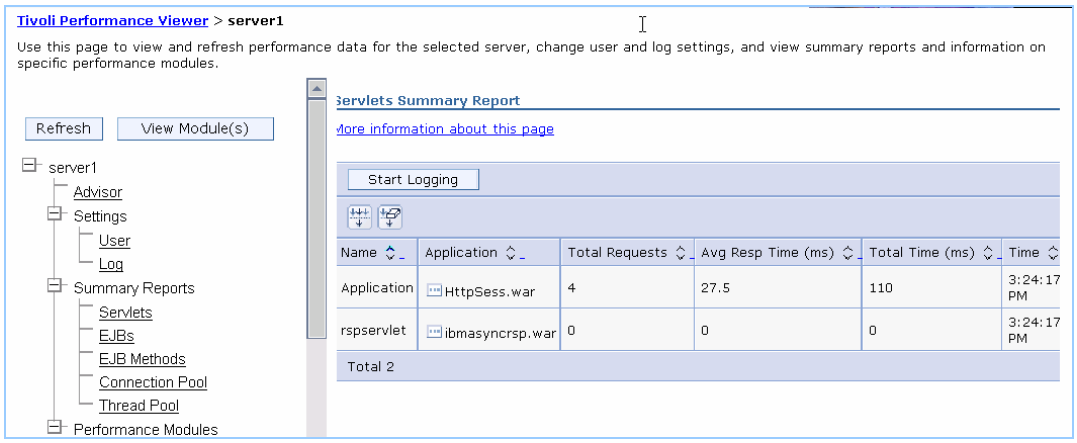
____ 7. Navigate to *Monitoring and Tuning* → *Performance Viewer* → *Current Activity*

Select **Server1** and click the '**Start Monitoring**' button

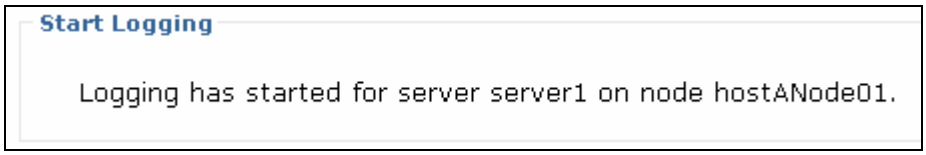


Expect to see this message:

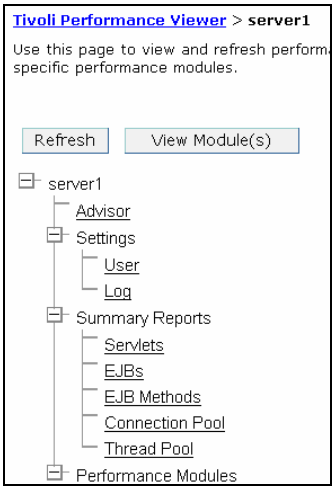
____ 8. Click **Server1** and you will see the '*Tivoli Performance Viewer*' for server1.



____ 9. On this screen, click the **Start Logging** button to start logging performance data. You will receive this message:



__ a. On the left side of this screen, you will see a tree of settings for server1.



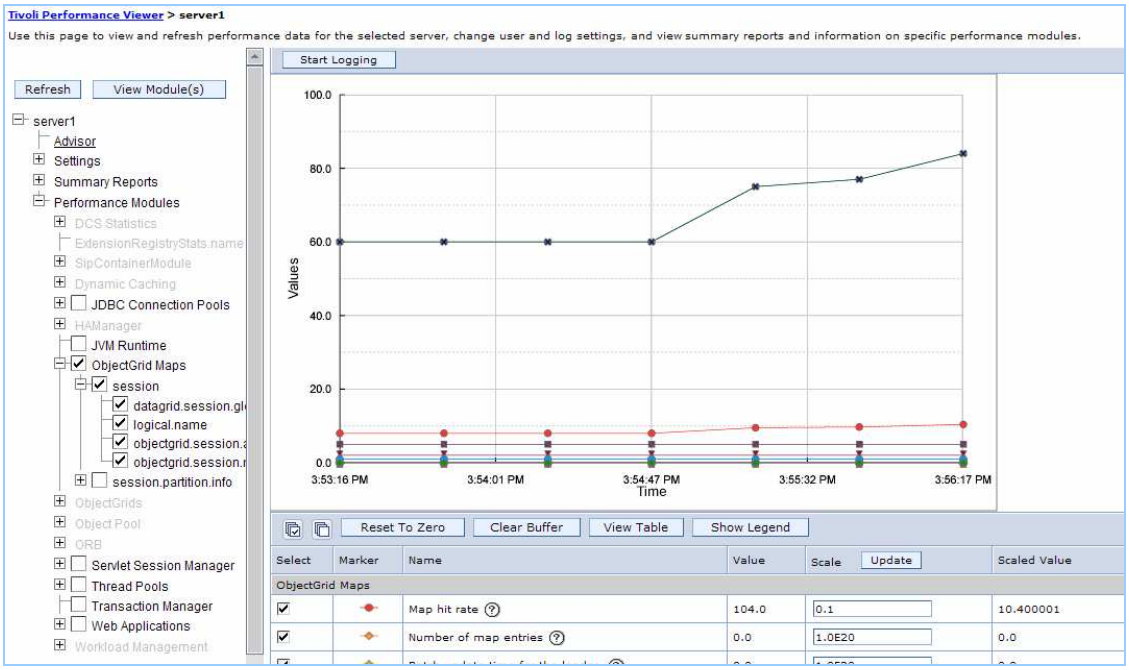
____ 10. Navigate this tree to set the attributes you want to monitor.

Server1 → Performance Modules → ObjectGrid Maps.

Select **session**, and all the four other entries under session including

- ☑ logical.name,
- ☑ objectgrid.session.metadata,
- ☑ datagrid.sessions.global.ids, and
- ☑ objectgrid.session.attribute.

Note: If you are unable to select “ObjectGrid Maps” you must stop and restart the server.



After these parameters are selected, you will see graphs and tables that show the data being stored into the backing maps.

NOTE: The intent is to show that http session data is being persisted into the eXtreme Scale grid and not to understand the backing map schema.

__ a. Open the Web browser and navigate to **http://localhost:9080/HttpSess/Application**

Invoke the application servlet multiple times from the browser and watch the metrics change, showing that the session data is being persisted in the grid.

Appendix A – Using Eclipse to modify application

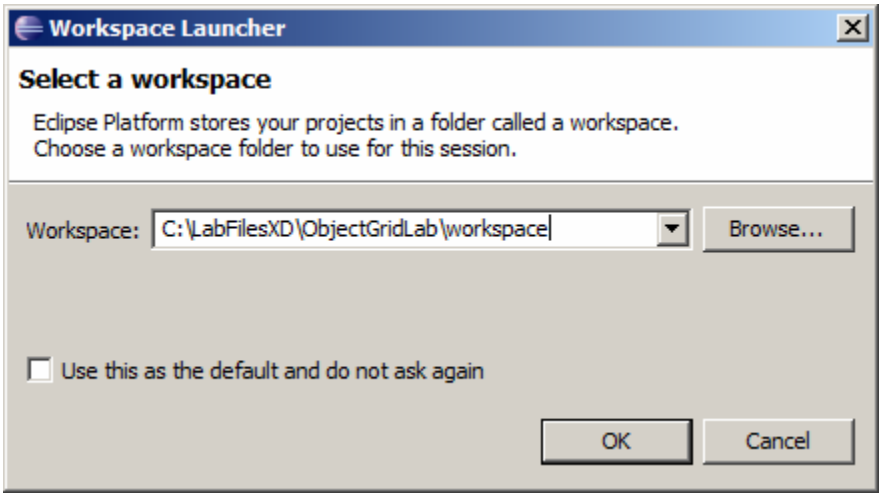
These notes provide an alternative way to insert the two sample session-related XML files into the HttpSessPersistence.ear file. In this section you will import the application into Eclipse, modify it, and re-export from an Eclipse project to the WebSphere environment.

____ 1. Create a new workspace in Eclipse

__ a. Launch Eclipse

1) From a Windows Command line, type "**`C:\eclipse\eclipse.exe`**"

__ b. When prompted to select a workspace, enter **C:\LabFilesXD\ObjectGridLab\workspace**.



__ c. Click **OK**.

__ d. Close the Welcome tab to go to the workbench.

You will now import the ear file, add the two session-related sample xml files to the ear file, and then export the files to re-create the HttpSessPersistence.ear file.

____ 2. Import HttpSessPersistence source code.

__ a. Select *File->Import…*

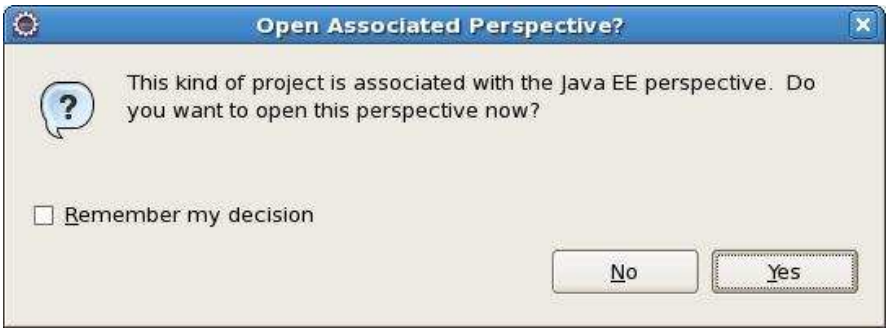
__ b. From the wizard panel, expand the "*Java EE*" section, select "*EAR File*", and click the **Next** button.

__ c. Click "**Browse…**" button next to the EAR File and select the **HttpSessPersistence.ear** file location.

__ d. Click "**Finish**" to import project.

The import will create two new projects. Project HTTPSessPersistence for the enterprise application, and project HTTPSess for the embedded Web application.

____ 3. You might be presented with a window asking if you want to switch to the JEE Perspective. If asked, click **Yes**.



____ 4. Add required libraries to the Web project

__ a. In the Project Explorer, Right click the **HttpSess** project, then click **Build Path → Configure Build Path… .**

__ b. Click the **Libraries** tab

__ c. Click **Add External JARs…**

__ d. In the *JAR Selection* window, browse to <*WAS_HOME*>*lib* and highlight **wsobjectgrid.jar**.

In a stand-alone environment you will browse to <WXS_HOME>\ObjectGrid\lib and select objectgrid.jar.

__ e. Click **Open**.

__ f. Verify that **wsobjectgrid.jar** is listed under the Libraries tab, as shown below.

__ g. Click **OK**.

__ h. You might need to add j2ee.jar file if errors persistent in the project.

____ 5. Import ObjectGrid configuration files.

__ a. Expand *HttpSess → Web Content → META_INF* folders.

__ b. Copy ObjectGrid deployment XML files from the ObjectGrid installation to the **WebContent\Meta-INF** of the HttpSessPersistence project.

```
Copy C:/IBM/ObjectGrid/ObjectGrid/session/samples/objectGrid.xml to the
..WebContent\Meta-INF directory of the HttpSessPersistence project in
Eclipse.

Copy
C:/IBM/ObjectGrid/ObjectGrid/session/samples/objectGridDeployment.xml
to the ..WebContent\Meta-INF directory of the HttpSessPersistence
project in Eclipse..
```

____ 6. Export the HttpSessPersistence file

__ a. Right click the Eclipse **HttpSessPersistence** project and **export EAR** file to the `C:/IBM\ObjectGrid/ObjectGrid/session/bin` directory. This will create a new file called HttpSessPersistence.ear in the `C:/IBM/ObjectGrid/ObjectGrid/session/bin` directory.

____ 7. You can now splice the HTTP session filter into the ear file and install the updated application.

What you did in this exercise

In this lab you installed a simple Web application which stores data in an HTTP session. Next you started a stand-alone eXtreme Scale catalog server and configured WebSphere to connect to this catalog server. You then modified the application to store its session data in the eXtreme Scale grid and used the Tivoli PMI Viewer to show eXtreme Scale grid usage to persist session data.