Design and Deploy WebSphere Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP

In Oracle Database RAC and Active Data Guard environments

ORACLE WHITE PAPER AUGUST 2015

Table of Contents

Introduction

Achieving maximum application uptime without interruptions is a critical business requirement. There are a number of requirements such as outage detection, transparent planned maintenance, and work load balancing that influence application availability and performance. The purpose of this paper is to help Java Web applications deployed with IBM WebSphere, achieve maximum availability and scalability when using Oracle.

Are you looking for best practices to hide your web applications from database outages? Are you looking at, smooth & stress-free maintenances of your web applications? Are you looking at leveraging Oracle Database's runtime load balancing in your WebSphere applications? This paper covers the configuration of your database and WebSphere Servlets for resiliency to planned, unplanned database outages and dynamic balancing of the workload across database instances, using RAC, ADG, GDS[1], and UCP.

Issues to be addressed

The key issues that impede continuous application availability and performance are:

- **Planned Maintenance:**
  - **Achieve transparent maintenance:** Make the maintenance process fast and transparent to applications for continuous availability.
  - **Session Draining:** When the targeted instance is brought down for maintenance, ensure that all work completes. We will describe how to drain sessions without impacting in-flight work and also avoid logon storms on active instance(s) during the planned maintenance.
- **Unplanned Downtimes:**
  - **Outage detection:** Web application's timeouts are unpredictable and unreliable. This paper describes how to configure WebSphere Servlets to be notified of outages as fast as possible.
  - **Error handling:** Several types of SQL exceptions may be received by your Servlets; how to determine that such errors are indicative of database service failure?
  - **Recovery with Response Time Targets:** Upon outage the Oracle Database RAC system needs a short period of time to recover before becoming fully operational again. How to react quickly and keep such "*brownout*" period under SLA targets?
  - **Outcome of in-flight work:** Have you ever paid twice for books, flights or taxes? Making a reliable determination of the outcome of the in-flight transaction in the face of database outages was a challenge until Oracle Database 12c. We will describe, how to design Servlets and configure Oracle Database 12c for solving this challenge.
  - **Continuation of in-flight work:** How to design Servlets and configure Oracle Database 12c and UCP to allow safe and transparent replay of in-flight transactions in the event of unplanned database outages.
- **Workload Balancing**: In RAC, RAC ONE and ADG environments, connection requests are by default distributed randomly by the Net Listener. How to configure your web applications and configure the database for optimal distribution of the workload when the node/services are added/removed?

1 http://www.oracle.com/technetwork/database/availability/maa-consolidation-2186395.pdf

The paper provides step by step instruction on how to configure JDBC driver, UCP as WebSphere data source and enable high availability properties thereby enabling your applications for planned database maintenance and unplanned database downtimes. Finally the paper discusses the recommended solutions.

Oracle Database 12c High-Availability and Load Balancing Concepts

To support high-availability and load balancing solutions, Oracle Database 12c and prior releases furnish HA configurations (RAC, Data Guard) and features which are leveraged by Oracle Database drivers (e.g., Oracle JDBC) and connection pools (e.g., UCP). This paper will refer to the following features, mechanisms, and concepts described in **Java Programming with Oracle Database 12c RAC and Active Data Guard** [2] white paper:

- Universal Connection Pool (UCP)
- Fast Application Notification (FAN)
- Oracle Notification Service (ONS)
- Fast Connection Failover (FCF)
- Logical Transaction ID (LTXID)
- Database Request
- Recoverable Errors
- Mutable Functions
- Transaction Guard (TG)
- Application Continuity (AC)

Configure WebSphere for UCP

Universal Connection Pool (UCP) has the built in support for planned maintenance, unplanned downtimes and runtime load balancing. UCP along with RAC, RAC One and ADG is a tested and certified combination for handling database failovers. UCP has been successfully used by many customers to handle failovers seamlessly. Configuring UCP in IBM WebSphere is explained in detail, hereafter.

Deploying a servlet which accesses Oracle Database through Oracle JDBC driver and Oracle Universal Connection Pool (UCP) in a WebSphere application container requires the following steps:

- Create a New JDBC Provider
- Create a New Data Source
- Create a JNDI lookup in the servlet
- Create a web.xml for the Servlet

Please note that **WebSphere Application Server version 8.5.5.3** is used in our testing and here are the step by step instructions. Please also, refer to "WebSphere Tips" section of the white paper while using IBM WebSphere.

Create a New JDBC Provider

Define `${ORACLE_JDBC_DRIVER_PATH}` at a location where the Oracle JDBC driver & related libraries are placed. Check Environment → WebSphere variables to define the driver's path as `${ORACLE_JDBC_DRIVER_PATH}`.

2 http://www-content.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

- **Add a New JDBC Provider: (Refer to Fig 1)**

  Navigate to Resources → JDBC → JDBC Providers

  Click New to add a New JDBC Provider

  Step 1: Create a new JDBC provider (Refer to Fig 1.1)

  Scope: Select the required scope from the drop down menu

  Database type: Select 'Oracle' from the drop down menu

  Provider type: Select 'Oracle JDBC Driver UCP' from the drop down menu

  Implementation type: Select 'Connection pool data source' from the drop down menu

  Name: This gets auto filled as 'Oracle JDBC Driver UCP'

  Description: Provide any description

  Step 2: Enter the database class path information (Refer to Fig 1.2)

  classpath: Specify the CLASSPATH for ojdbc7.jar, ucp.jar & ons.jar. **Use jar files from the same database version**

  Eg: ${ORACLE_JDBC_DRIVER_PATH}/ojdbc7.jar. Please note the significance of each library.

  ojdbc7_g.jar or ojdbc7.jar → JDBC driver with or without debug.

  ucp.jar → Required for using UCP

  ons.jar → Required for listening to FAN events.

  Directory location: Mention the path where the above jar files are placed.

  Step 3: Summary (Refer to Fig 1.3)

  Implementation Class Name: Please note that IBM WebSphere correctly chooses and sets the Implementation class as '**oracle.ucp.jdbc.PoolDataSourceImpl'** based on the selections in Step 1. PLEASE DO NOT CHANGE THIS. Changing this to any other value will cause connecting to the database to fail. Click FINISH to confirm all the changes.

  Refer to Fig 1.4 to check the settings after completing all 3 steps above

Fig 1: Add a New JDBC Provider

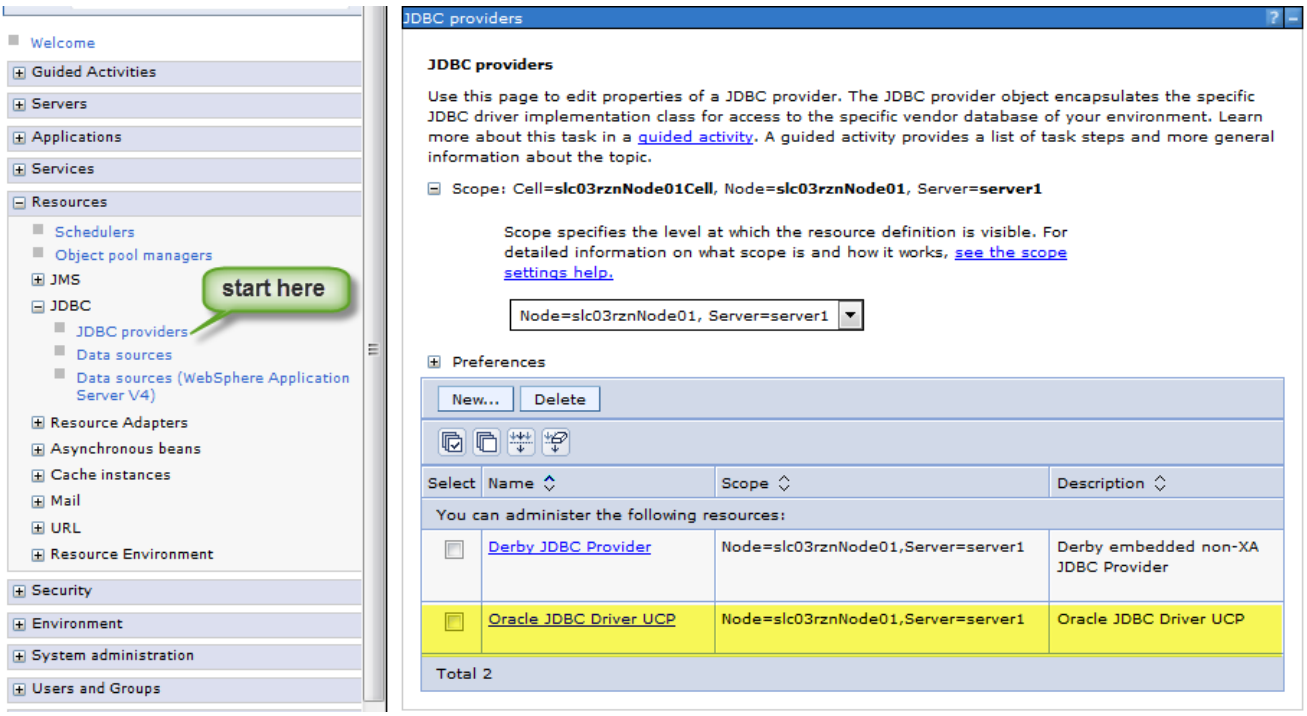


Fig 1.1 : Create new JDBC provider

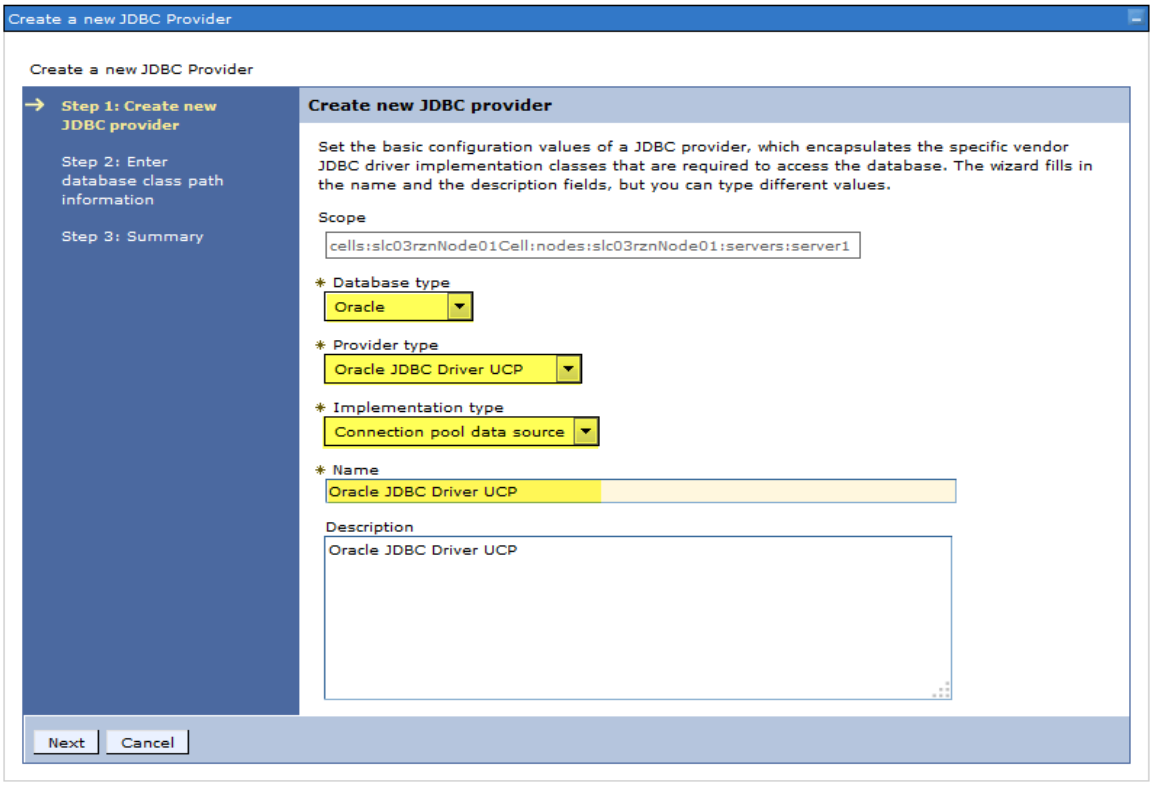


Fig 1.2: Enter database class path information

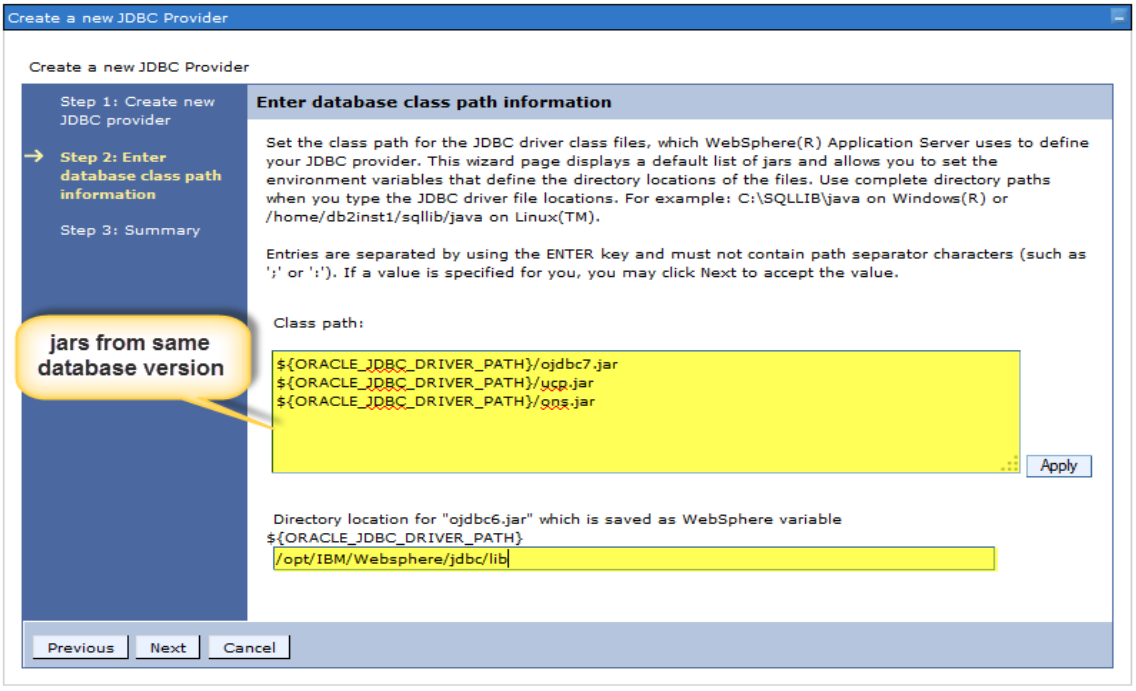


Fig 1.3: Summary

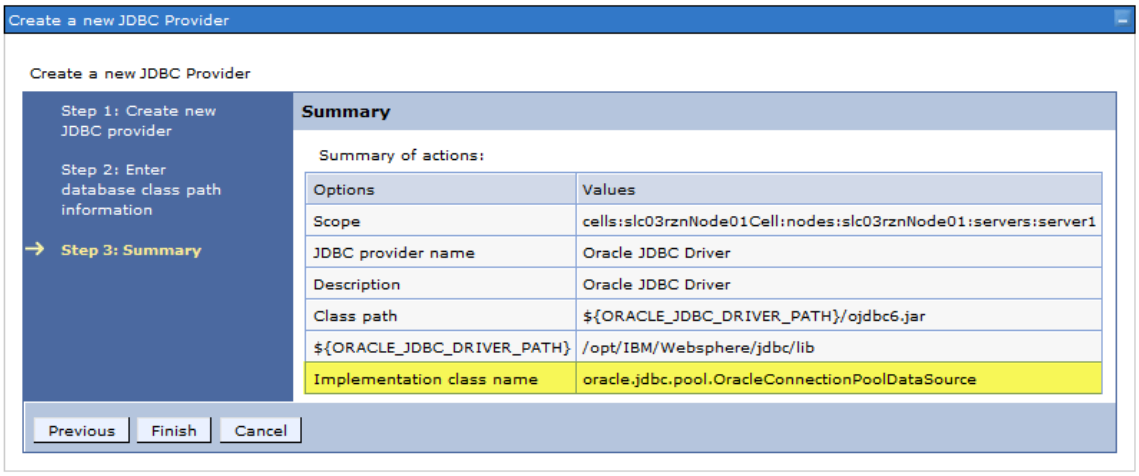
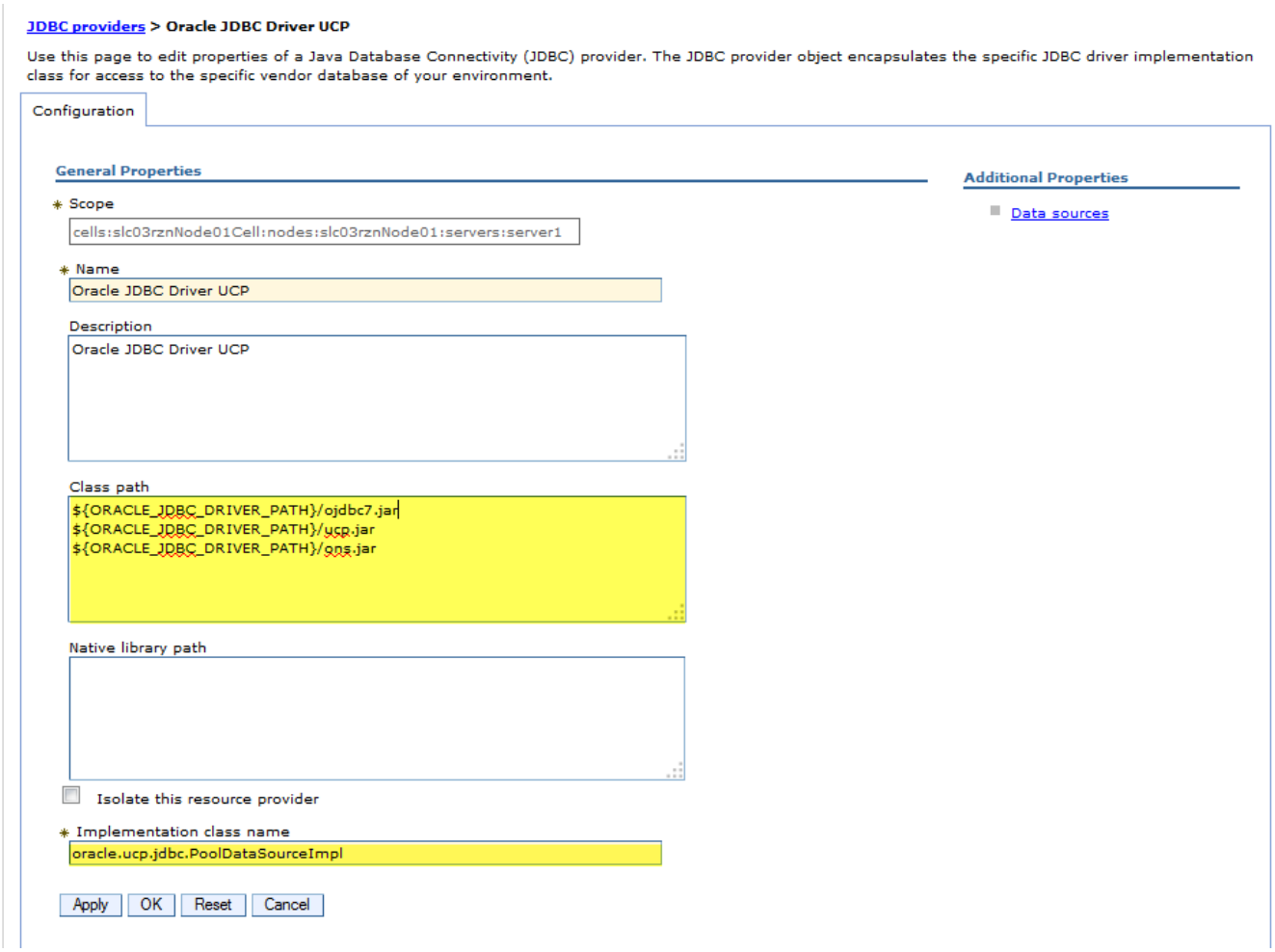


Fig 1.4: Newly added JDBC provider



Create a New DataSource

A new data source is required for connecting to the Oracle Database. The steps are as highlighted below.

- Create a New JAAS-J2C Authentication Data
- Create a New Data Source
- Verify if WebSphere connection pool is disabled
- Set Custom Connection Pool Properties i.e., UCP properties
- Restart the Server after adding a new datasource
- Test Connection

Each one of these steps is explained in detail with screenshots, hereafter.

- **Create a New JAAS-J2C Authentication Data (Refer to Fig 2.1 & Fig 2.1.1)**
  Navigate to *Security → Global Security → Java Authentication and Authorization Service → J2C Authentication data*
  Click New to add a new *JAAS-J2C Authentication Data and fill in the following details.*
  Alias: Choose any appropriate Alias. Such as RAC12c, OracleDB etc.,
  User ID: Enter the username of the Oracle Database
  Password: Enter the password of the Oracle Database
  Refer to Fig 2.1.1 which displays the DB username & password

- **Create a New Data Source (Refer to Fig 2.2)**
  Navigate to *Resources → JDBC → Data Sources*
  Click New to add a new Datasource
  Step 1: Enter basic data source information (Refer to Fig 2.2.1)
  Data source name: Select the appropriate Data source name. E.g., orclDataSource
  JNDI Name: Please make sure that JNDI name is as mentioned "/jdbc/<datasourcename>" Eg., /jdbc/orclDataSource
  Step 2: Select JDBC Provider (Refer to Fig 2.2.2)
  Select an existing JDBC provider **:** Choose the already created JDBC Provider as shown in the screenshot.
  Step 3: Enter database specific properties for the data source (Refer to Fig 2.2.3)
  URL **: Enter the Connect string URL used to connect to the Oracle RAC database.**
  Example:
  jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=proddbcluster-scan)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=proddb)))
  Data store helper class name: Select 'Oracle11g data store helper' from the dropdown menu.

  Step 4: Setup security aliases (Refer to Fig 2.2.4)
  Component-managed authentication alias: Select the J2C Authentication created as per Fig 2.1 from the dropdown menu.
  Mapping-configuration alias: Do not select anything
  Container-manager authentication alias: Select the J2C Authentication created as per Fig 2.1 from the dropdown menu
  Step 5: Summary (Refer to Fig 2.2.5)
  Check all the details to make sure everything is entered correctly and click FINISH

  Refer to Fig 2.2.6 to verify the summary of the dataSource anytime.

- **Verify if WebSphere connection pool is disabled. WEBSPHERE AUTOMATICALLY TAKES CARE OF THIS STEP.**

  The data source will be configured to use UCP with the default settings. The following properties are automatically set on the data source. Do not alter any of these properties. Changing any of these could cause the data source to no longer work properly.

  Step 1: WebSphere connection pooling is turned off. (Refer to Fig 2.3.1)

  To verify this, select data source created. Example: `orclDataSource`

  Click on Connection pools → Maximum connections to see if it is set to 0.

  Note: Maxconnections =0, indicates that WebSphere connection pooling is turned off.

  Changing to a value other than zero will cause WebSphere to track the number of connections attempted which conflicts with the number that Oracle UCP is tracking. It is not advisable to change this setting.

  Step 2: WebSphere prepared statement caching is turned off (Refer to Fig.2.3.2)

  To verify this, select the data source created. Example:orclDataSource

  Click on Websphere Application Server data source properties → Statement cache size to see if it is set to 0.

  Note: WebSphere prepared statement caching can only be used when WebSphere connection pooling is turned on. Since, we are using UCP, this should be turned OFF.

  Step 3: Verify the correct connectionFactoryClassName (Refer to Fig 2.3.3)

  To check this, select the UCP datasource; e.g., orclDataSource

  Click on Custom Properties →connectionFactoryClassName, check that it is set to **oracle.jdbc.pool.OracleDataSource** when you select UCP. Or set it to **oracle.jdbc.replay.**OracleDataSourceImpl if you want to use use Application Continuity (AC).

  Note: Setting the connectionFactoryClassName to any other value will throw an exception.

  Step 4: Custom Property to disable WebSphere connection Pool (Refer to Fig 2.4)

  disableWASConnectionPooling is set to true, by default. Otherwise, you must explicitly set it to true.as follows:

  Select the datasource in question; e.g., `orclDataSource`

  Click on Custom Properties and create a new property disableWASConnectionPooling; then set it to true

- **Set Custom UCP Properties such as FCF (Refer to Fig 2.4)**

  FCF (fastConnectionFailoverEnabled) is an important property which handles failover of instances during both planned and unplanned downtimes. It is mandatory to have this property turned on. For more details on how to form `ONSConfiguration` string, refer to the Oracle Notification Service (ONS) section of the white paper "**Java Programming with Oracle Database 12c RAC and Active Data Guard** [3]"

  Select the datasource in question e.g.,`orclDataSource`

  Click on Custom Properties then New and add the desired UCP properties shown below.

Property Name	Property Type	Property Details
minPoolSize	java.lang.String	Set the appropriate minimum pool size
maxPoolSize	java.lang.String	Set the appropriate maximum pool size
initialPoolSize	java.lang.String	Should be closer to minPoolSize
fastConnectionFailoverEnabled	java.lang.Boolean	Required. Set it to TRUE
disableWASConnectionPooling	java.lang.Boolean	Required. Set it to TRUE
ONSConfiguration	java.lang.String	Optional. Required for pre 12c Oracle Database version

- **Restart the Server**

  Refer to 'WebSphere Tips' for more details on restarting the servers.

3 http://www-content.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

- **Test Connection (Refer to Fig 2.5)**
  Select Datasource → Test Connection

Fig 2.1: New J2C Authentication Data

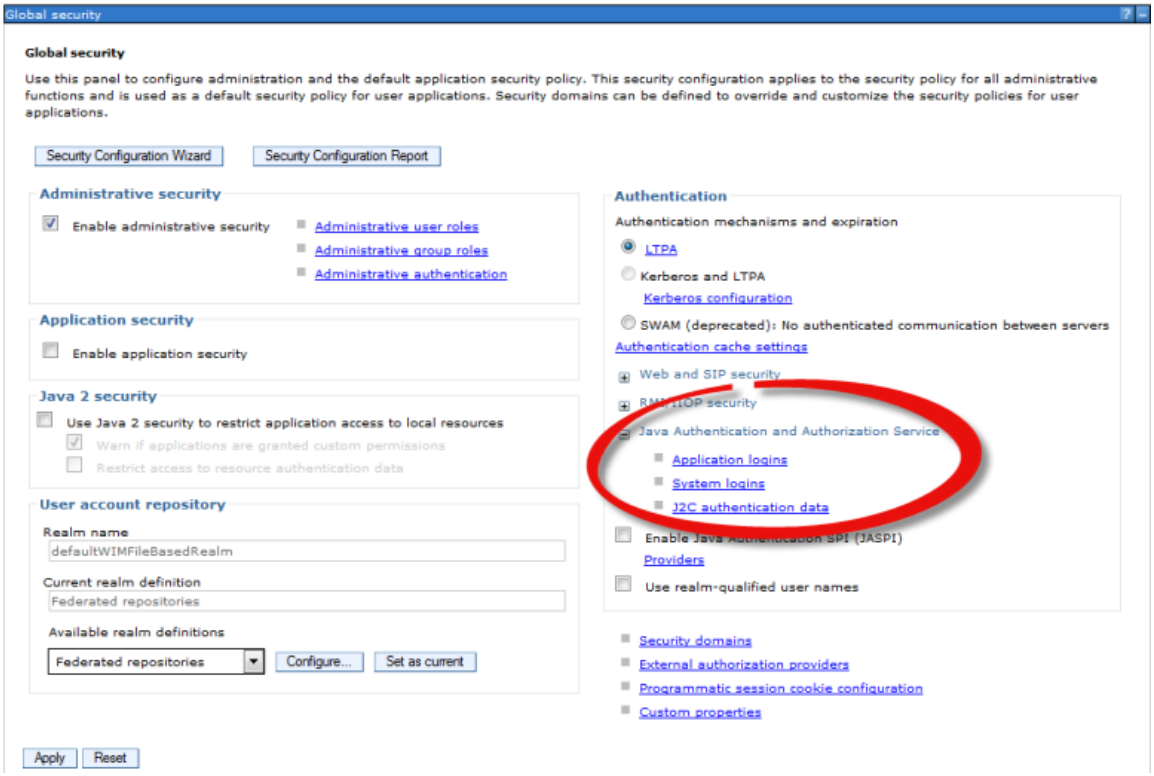


Fig 2.1.1: Set the Database Username/Password

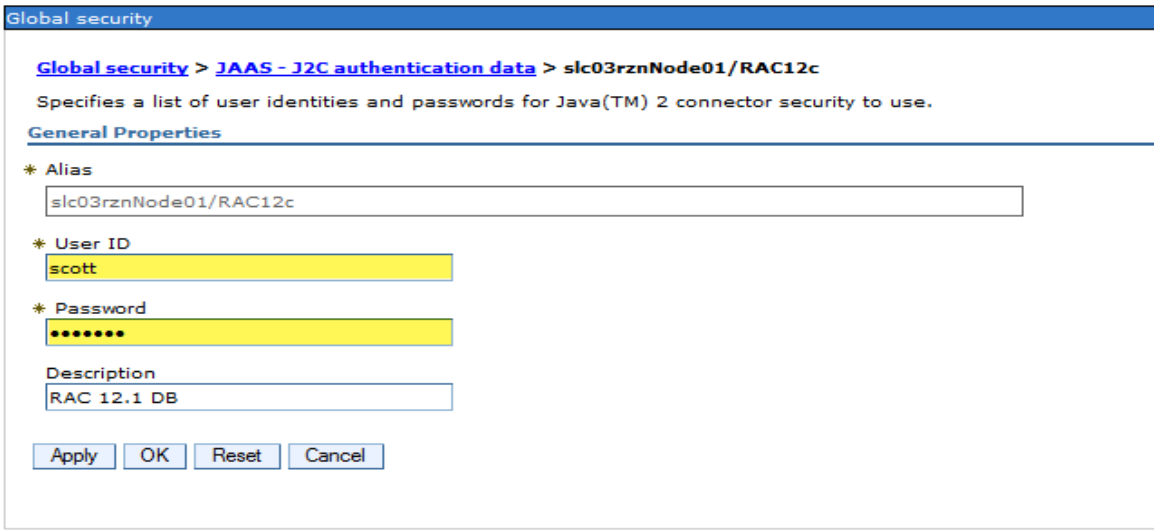


Fig 2.2: Adding a new DataSource

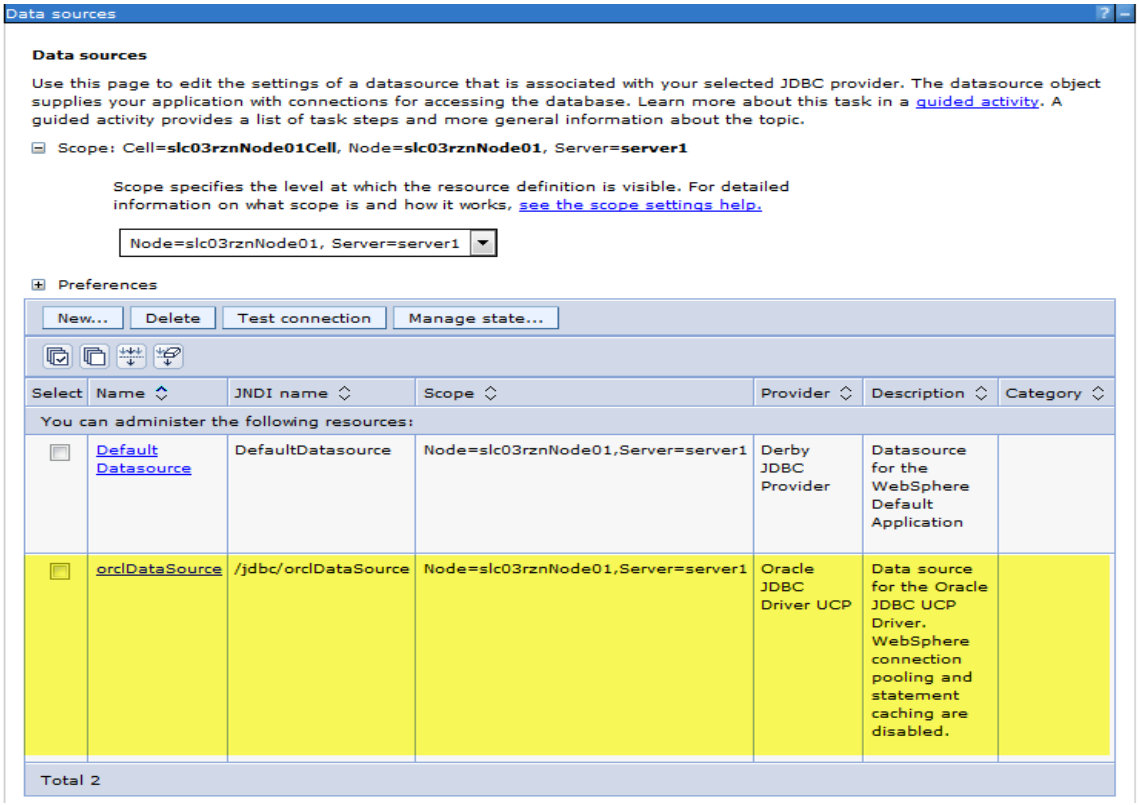


Fig 2.2.1: Enter some basic data source information

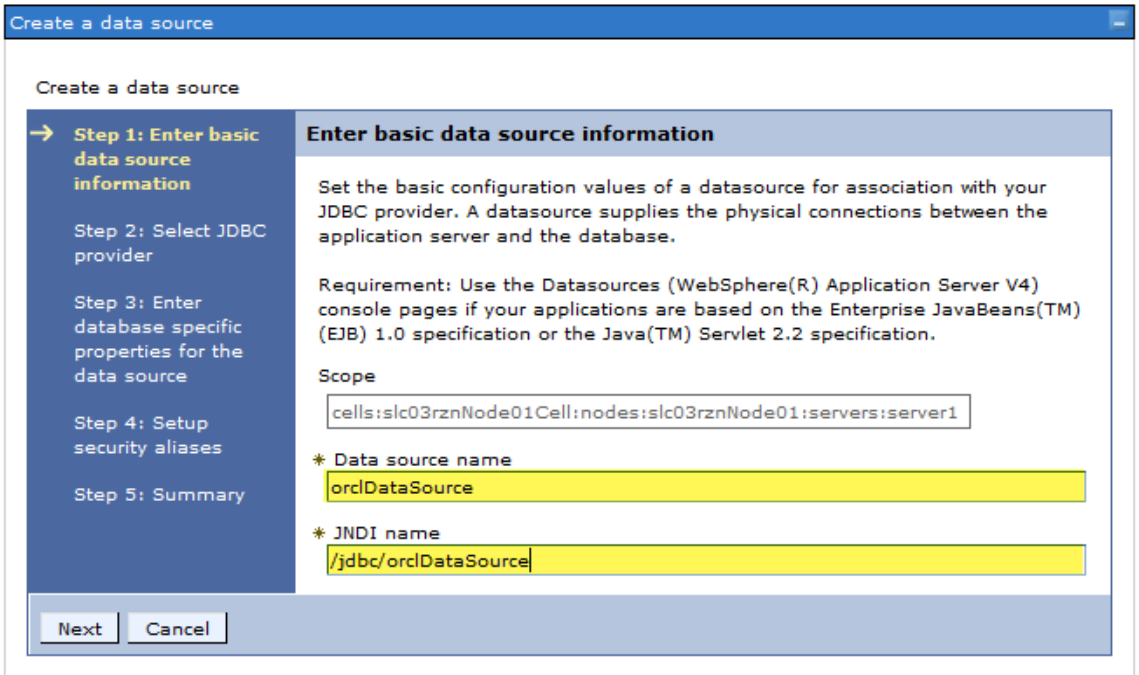


Fig 2.2.2: Select JDBC provider

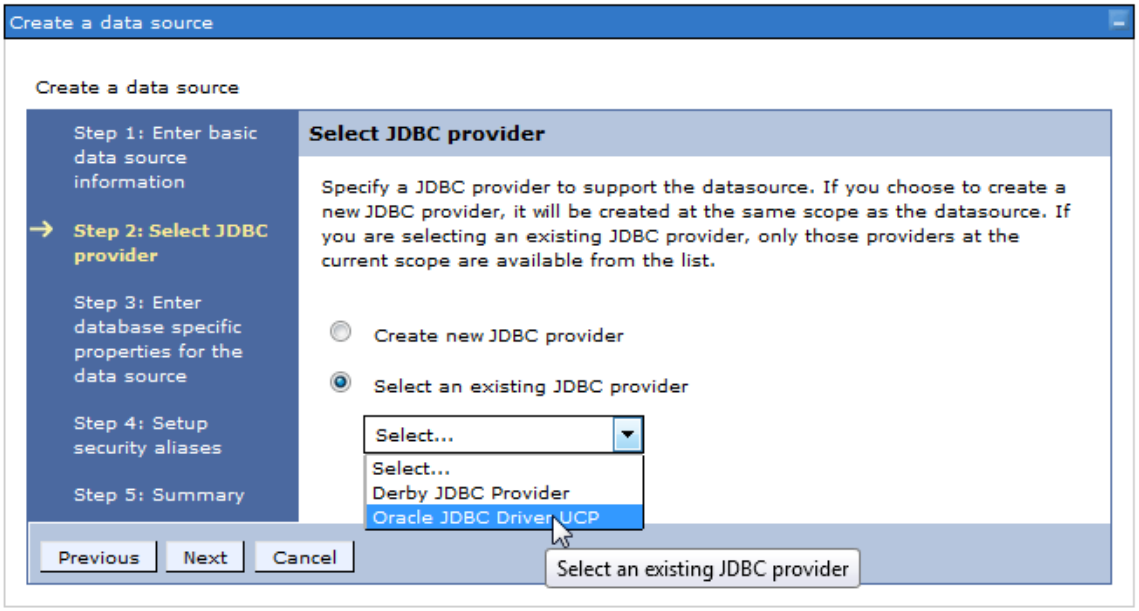


Fig 2.2.3: Enter the database specific properties for the data source

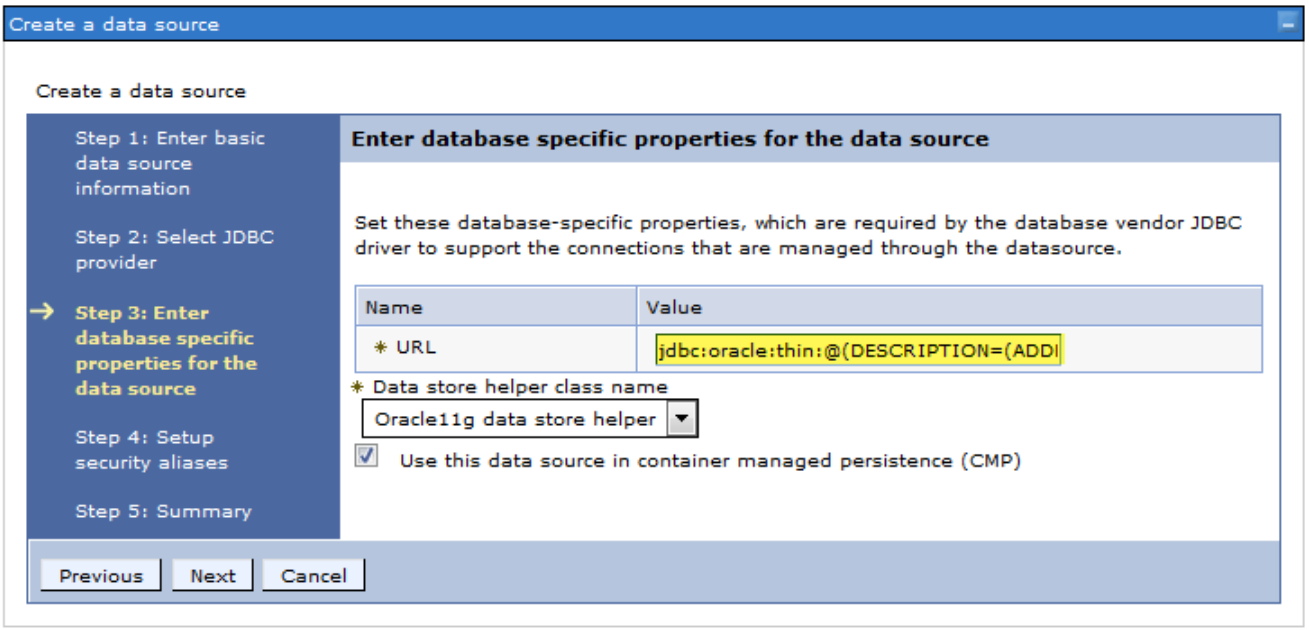


Fig 2.2.4: Set the security aliases

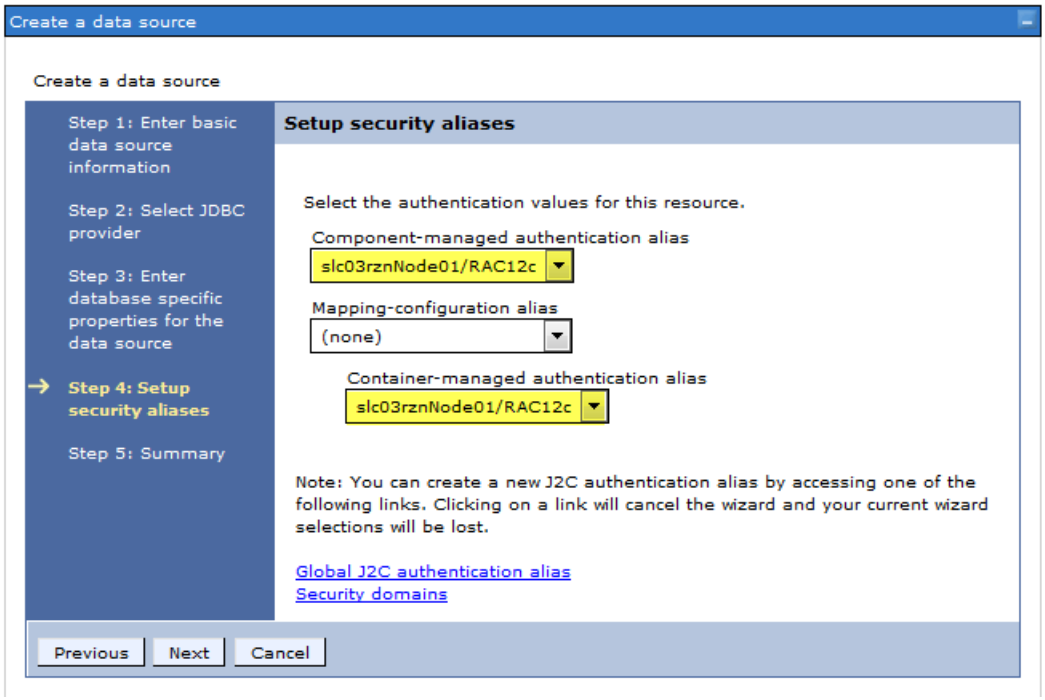


Fig 2.2.5: Summary

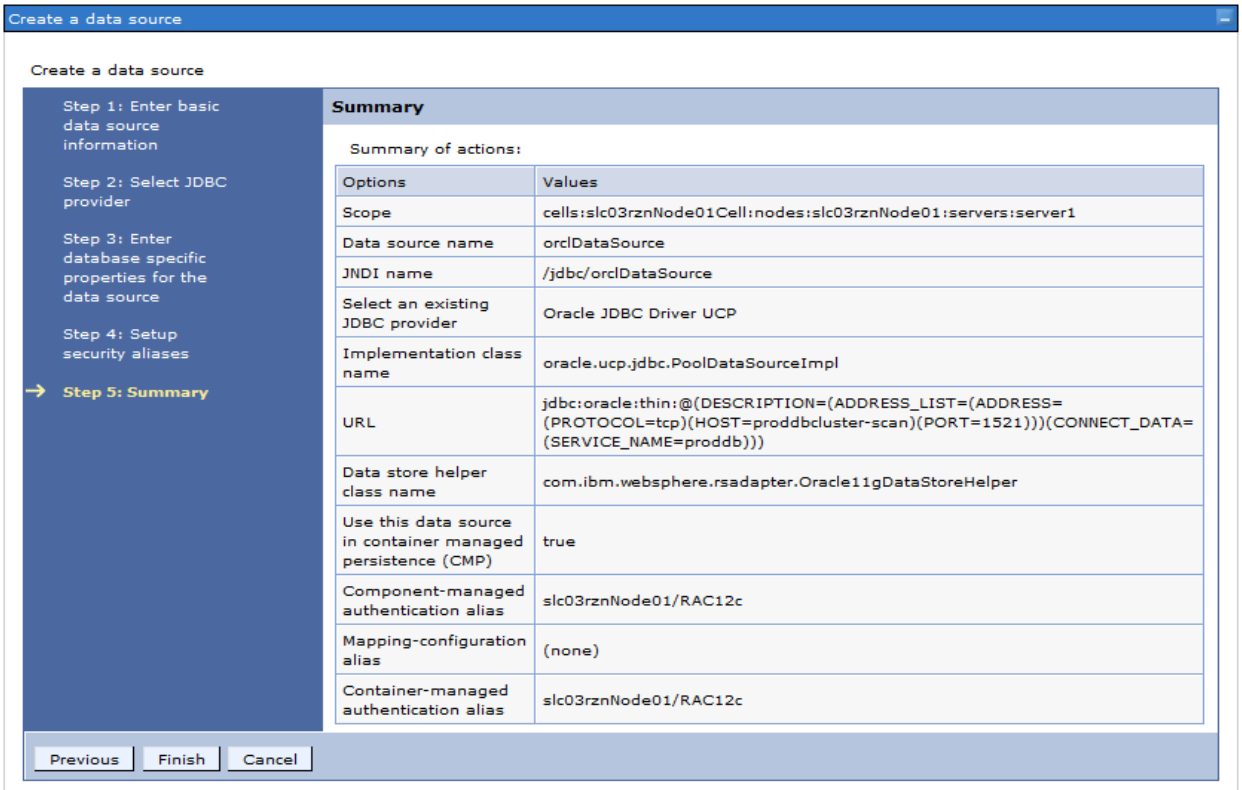


Fig 2.2.6: Details of the JDBC Datasource

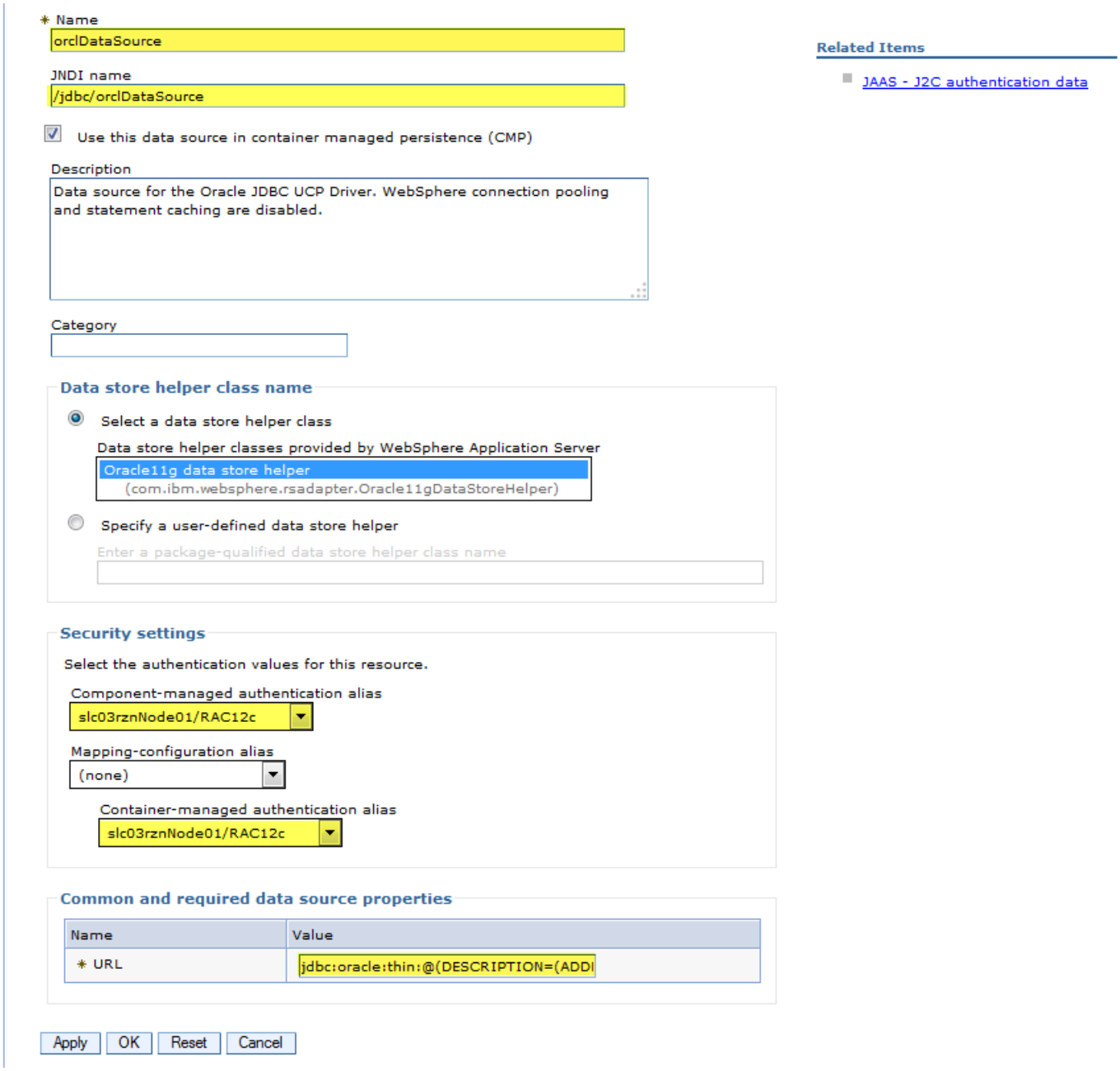


Fig 2.3.1: WebSphere connection pooling is turned off

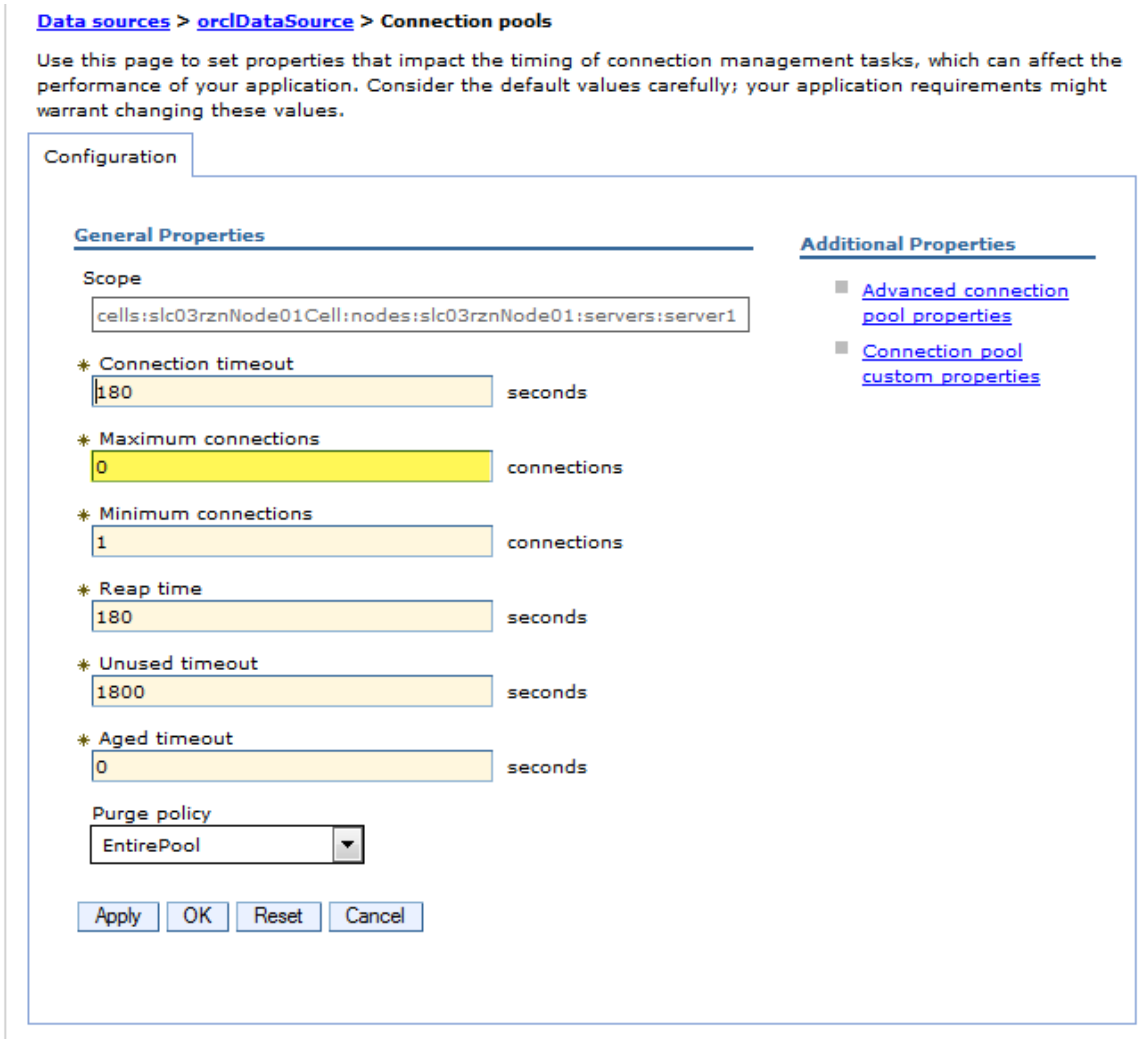


Fig 2.3.2: WebSphere prepared statement caching is turned off

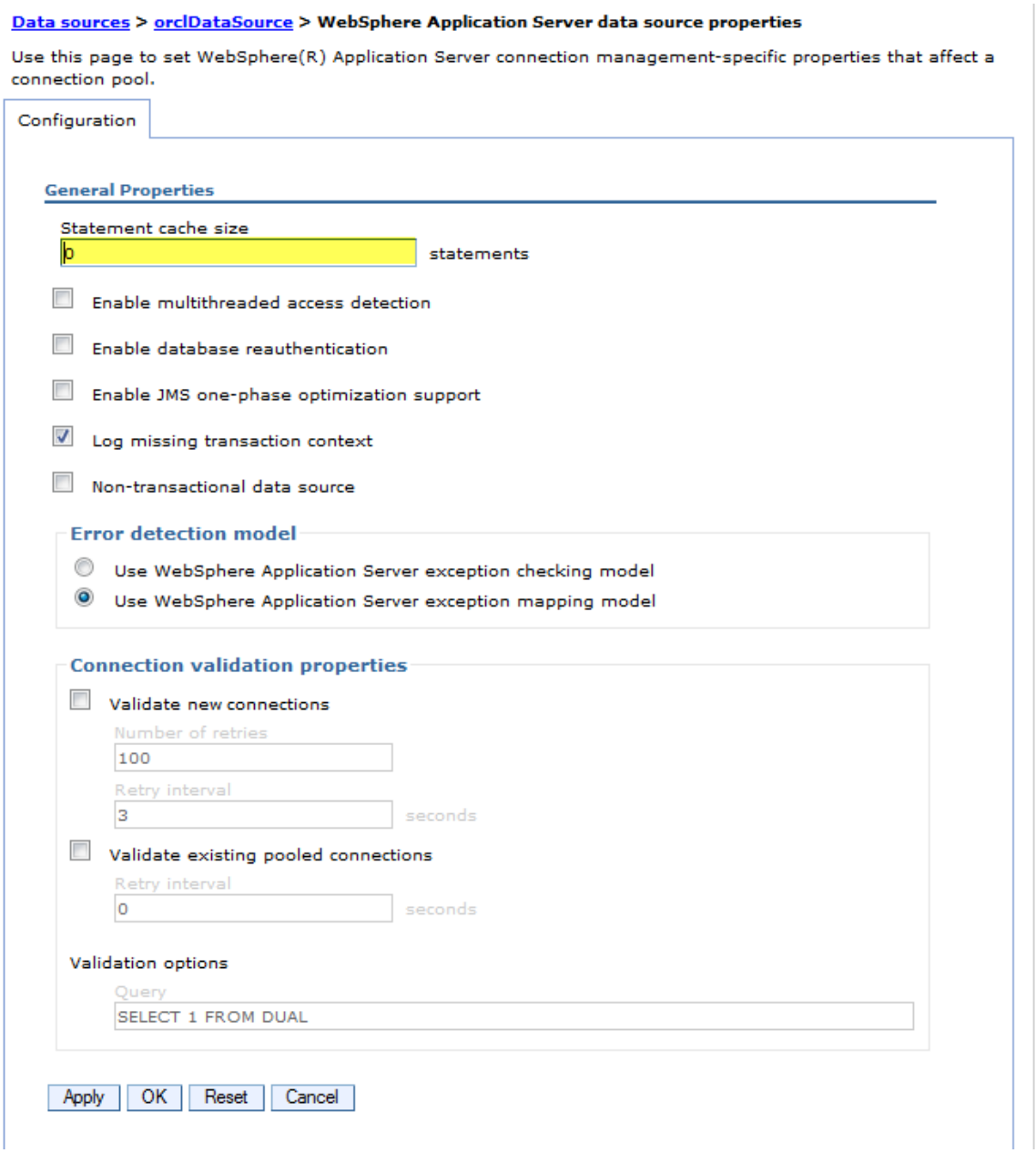


Fig 2.3.3: Verify the connectionFactoryClassName

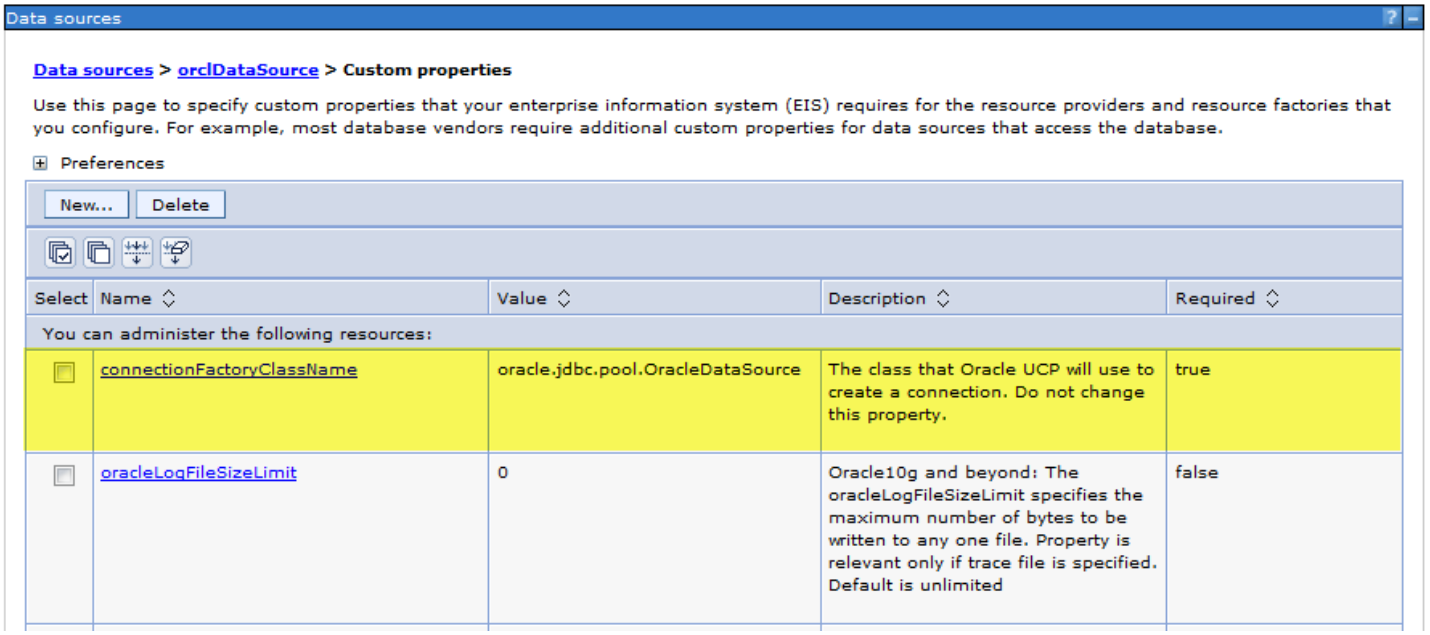


Fig 2.4: Enabling FCF

Data sources > orclDataSource > **Custom properties**

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

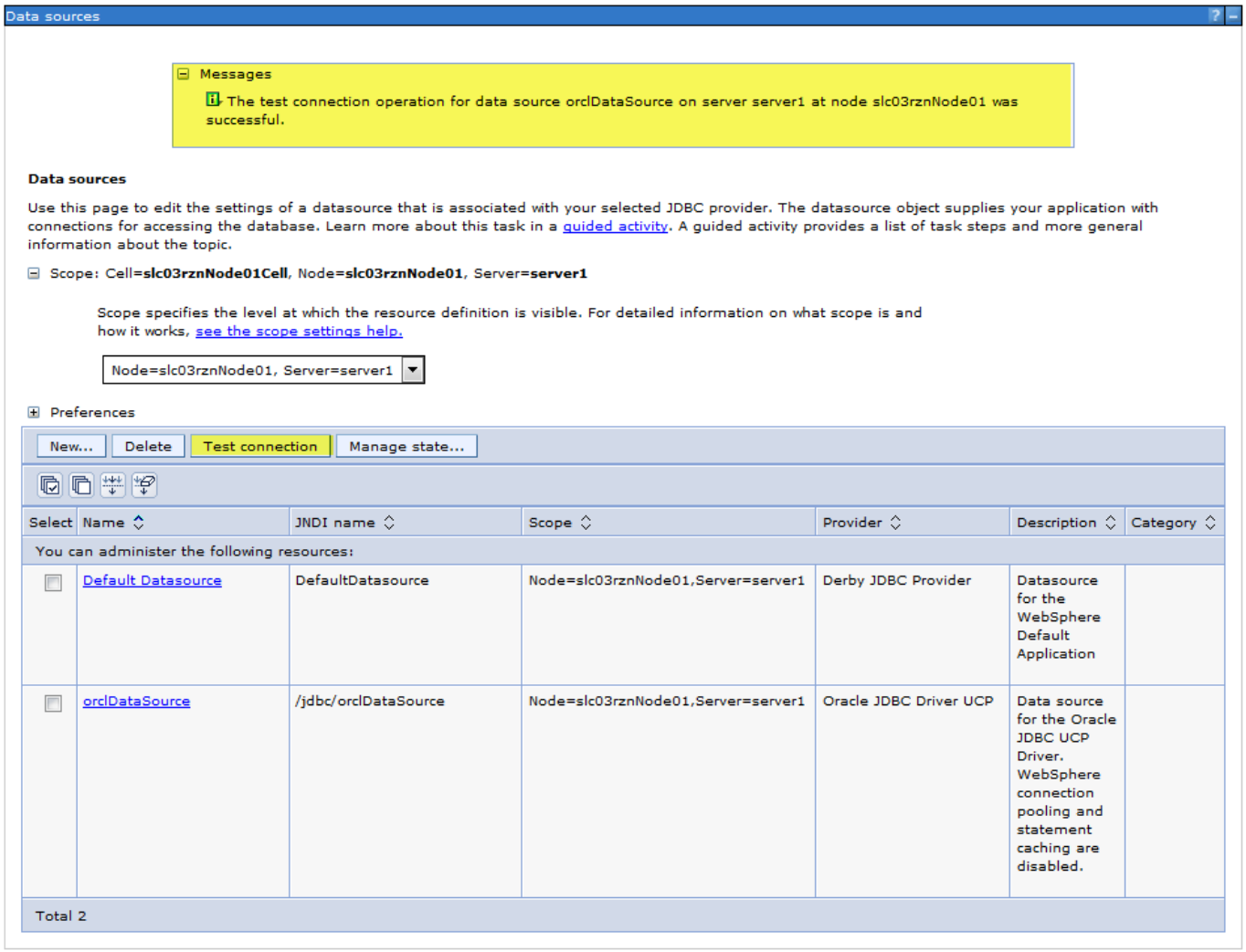
Preferences

New... Delete

Select	Name	Value	Description	Required
You can administer the following resources:				
	webSphereDefaultIsolationLevel		Specifies a default transaction isolation level for new connections. Resource References and Access Intents override this value. To configure a default transaction isolation level, use the constants defined by JDBC: 1 (READ UNCOMMITTED), 2 (READ COMMITTED), 4 (REPEATABLE READ), 8 (SERIALIZABLE).	false
	webSphereDefaultQueryTimeout		Sets a default query timeout, which is the number of seconds (0 means infinite) that a SQL statement may execute before timing out. This default value is overridden during a JTA transaction if custom property syncQueryTimeoutWithTransactionTimeout is enabled.	false
	enableClientInformation	false	Enables the implicit passing of client information on database connections. The client information provided on each connection is the same as for the WAS.clientinfo trace group. Whereas the WAS.clientinfo trace is configured on an application server, the enableClientInformation property applies to a data source configuration.	false
	onsConfiguration	nodes=slc06bmu:25227,slc06bmv:25227,slc06bmw:25227	Required only for 11g and Not required in 12c which has auto-ONS	false
	fastConnectionFailoverEnabled	true	Required setting	false
	minPoolSize	2		false
	maxPoolSize	50		false
	disableWASConnectionPooling	true		false
	initialPoolSize	15		false

Page: 2 of 2 Total 29

Fig 2.5: Test the Connection with the Oracle Database



Create a JNDI context in the servlet

The following code snippet shows how to get a database connection by referring to the JNDI datasource created in Websphere.

```
PoolDataSource pds = getPoolInstance();

conn = pds.getConnection();
```

```
private PoolDataSource getPoolInstance() throws SQLException {
    javax.naming.InitialContext ctx = null;
    javax.sql.DataSource pds = null;
    System.out.println ("Attempting connection..." + DateUtil.now());
    ctx = new javax.naming.InitialContext();
    javax.sql.DataSource ds = (javax.sql.DataSource) ctx
        .lookup("java:comp/env/jdbc/orclDataSource");
    PoolDataSource pds = (PoolDataSource) ds;
    return pds;
  }
```

Create a web.xml for the Servlet

The data source resource reference should also be present in web.xml as illustrated hereafter.

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">
        <display-name>test1</display-name>
        <servlet-mapping>
                <servlet-name>com.test1.DemoServlet</servlet-name>
                <url-pattern>/DemoServlet</url-pattern>
        </servlet-mapping>

        <resource-ref>
                <description> Datasource to connect to DB </description>
                <res-ref-name>jdbc/orclDataSource</res-ref-name>
                <res-type>javax.sql.DataSource</res-type>
                <res-auth>Container</res-auth>
        </resource-ref>
</web-app>
```

WebSphere Tips

Refer to this section when you require more details on how to access WebSphere console, start/stop an application server, how to set java system property in the console etc., These tips come handy during application deployment.

Description	Details
WebSphere Administrative Console	http://localhost:9060/ibm/console/login.do Usually 9060 is the default port where admin console is accessed.
Startup and shutdown scripts location	{WAS_INSTALL_DIR}/IBM/Websphere/AppServer/profiles/<AppServProfileName >/bin Example:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin/
Start an application server	Start Command : ./startServer.sh <Name of the server> -profileName <AppServerProfileName> Example: ./startServer.sh server1 –profileName AppSrv01
Stop an application server	Stop Command: ./stopServer.sh <Name of the server> Example: ./stopServer.sh server1
Increase the number of threads	The default number of threads in Websphere will be 10. If you want to change this, go to Servers → WebSphere application servers → <server name> → Thread Pools → Default. Change the Maximum size to the required value (Eg. 50)
Setting up a System Property	Servers → WebSphere Application servers → <servername> → “Java & Process Management” (Process Definition) → Java Virtual Machine → Custom Properties Add any JVM system property required.
Check if ONS is running or configured	Make sure to add $ORACLE_CONFIG_PATH to the path where ONS is running. Environment → WebSphere variables → (Add ORACLE_CONFIG_HOME)

Hiding Planned Maintenance from WebSphere Applications

For maintenance purposes (e.g., software upgrades), the Oracle Database instances can be gracefully shutdown one or several at a time without disrupting the operations and availability of the Web applications. Upon FAN **DOWN** event[4], UCP drains sessions away from the instance(s) targeted for maintenance. What is the configuration of Web applications and the database to achieve *session draining at service stop or relocation*? In a nutshell, the procedure consists in stopping non-

4 status=down reason=user

singleton services running on the target database instance or relocating singleton services from the target instance to a new instance.

Developer or Web Applications Steps

To hide the planned database maintenance, Web applications need to:

(i) enable Fast Connection Failover (FCF) as mentioned above. Please refer to *"Fig 2.4: Enabling FCF"* for more details. FCF can also be enabled programmatically as illustrated hereafter;

```
PoolDataSource pds = new PoolDataSourceFactory.getPoolDataSource();
// not required with auto-ONS in 12c
pds.setONSConfiguration("nodes=<RACNode1>:<port1>,<RACNode2>:<port2>,<RACNode3>:port3");
pds.setFastConnectionFailoverEnabled(true);
```

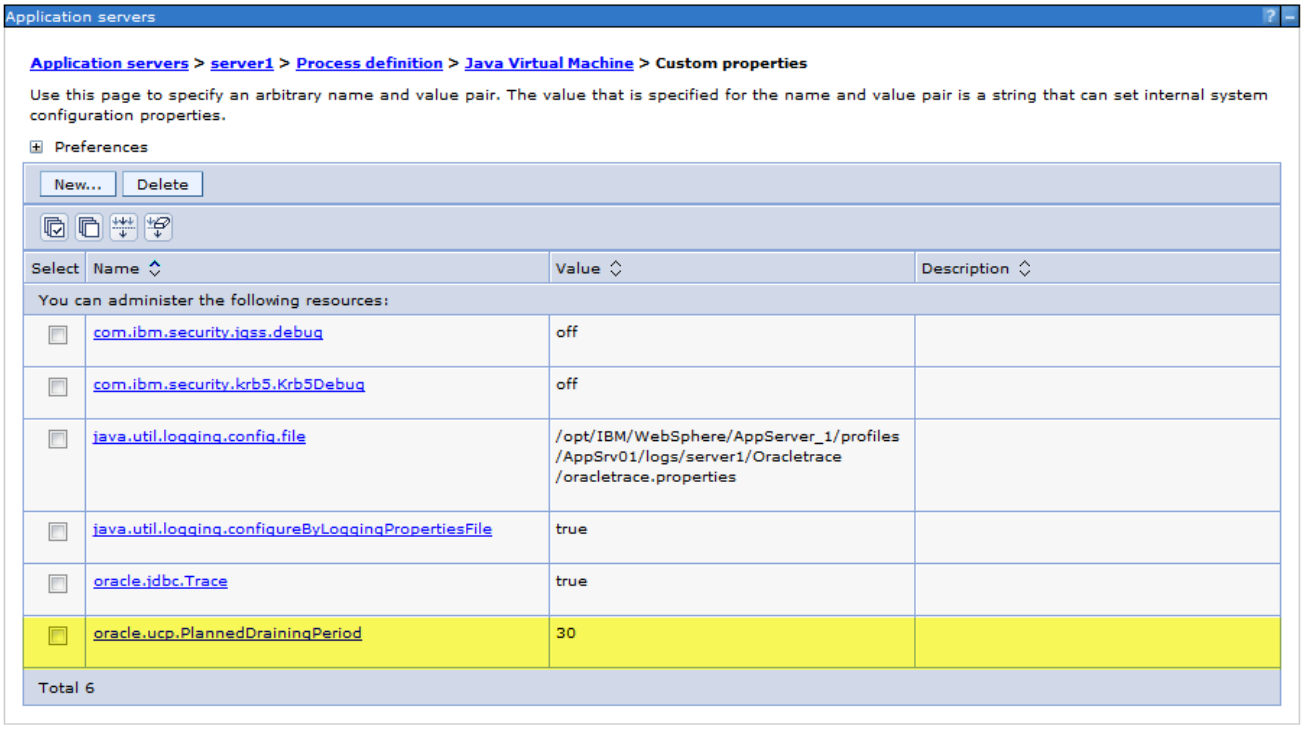
(ii) check that ons.jar is in the classpath.

(iii) In addition, with release 12.1.0.2, UCP introduces `PlannedDrainingPeriod`, a new system property which allows a graceful draining period. It can be specified as a JDK system property (i.e., using -D)

```
-Doracle.ucp.PlannedDrainingPeriod=30
```

In IBM WebSphere, the JVM system property can be set as follows. (Refer to Fig.3)
Servers → WebSphere application servers → <servername> → Java and Process Management (Process Definition) → Java Virtual Machine → Custom properties

Fig 3: Setting PlannedDrainingPeriod as System property



DBA or RDBMS Steps

DBAs should perform the following steps[5] to stop all services on the target machine where the database instance is scheduled for maintenance. . For each service repeat the following actions:

1. Stop the service without using `-force` option or relocate the service. Service relocation is required for singleton service (i.e., runs only on one instance at a time)

```
$srvctl stop service -db <db_name> -service <service_name> -instance <instance_name
or (NOTE: Omitting -service stops all services)
$srvctl relocate service -db <db_name> -service <service_name> -oldinst <oldins> -
newinst <newinst>
```

2. Disable the service and allow sessions some time to drain. E.g., 2-30 minutes. This avoids the logon storm on the other active instance where the workload gets transferred. Disabling service is optional if you choose to disable the instance.

```
$srvctl disable service -db <db_name> -service <service_name> -instance <instance_name>
```

3. Wait to allow sessions to drain Example: 10-30 minutes
4. Check for long-running sessions and terminate these (you may check again afterwards)

```
SQL> select count(*) from ( select 1 from v$session where service_name in
upper('<service_name>') union all
select 1 from v$transaction where status = 'ACTIVE' )
SQL> exec dbms_service.disconnect_session ('<service_name>',
DBMS_SERVICE.POST_TRANSACTION);
```

5. Repeat steps 1-4 for all services targeted for planned maintenance.
6. Stop the database instance immediately.

```
$srvctl stop instance -db <db_name> -instance <instance_name> -stopoption immediate
```

7. Disable instance to prevent restarts during maintenance

```
srvctl disable instance -db <db_name> -instance <instance_name>
```

8. Apply patch or carry out the scheduled maintenance work
9. Enable and then start the instance again

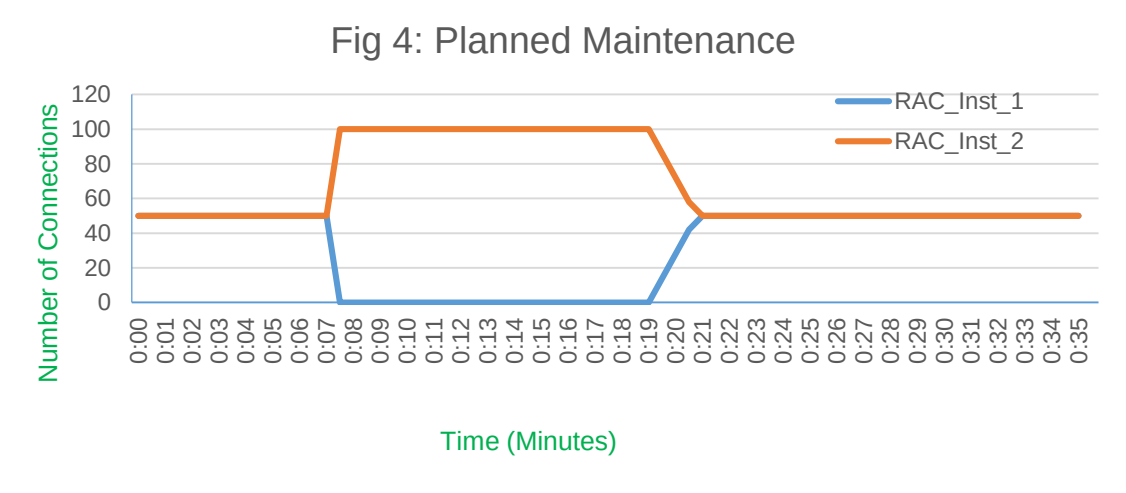
5 See Metalink note 1593712.1 @ https://support.oracle.com/epmos/faces/DocumentDisplay?id=1593712.1 for more details

```
$srvctl enable instance –db <db_name> -instance <instance_name>
$srvctl start instance –db <db_name> -instance <instance_name>
```

10. Enable then start the service back and check if the service is up and running

```
$srvctl enable service –db <db_name> -service <service_name> -instance <instance_name>
$srvctl start service –db <db_name> -service <service_name> -instance <instance_name>
```

Figure 4, shows connections distribution of XYZ service across two RAC instances before and after Planned Downtime. Notice that the connection workload goes from fifty-fifty across both instances to one hundred-zero. In other words, RAC_INST_1 can be taken down for maintenance without any impact on the business operation.



Hiding Unplanned Database Downtime from WebSphere applications

WebSphere Servlets can be configured to handle unplanned database outages using the following features and mechanisms:

- **Fast Connection Failover (FCF)**
- **Transaction Guard (TG)**
- **Application Continuity (AC)**

Please refer to the white paper, **Java Programming with Oracle Database 12c RAC and Active Data Guard** [6] for understanding these concepts in detail.

Developer or Web Application Steps

Need to set FCF to true for handling unplanned outages. FCF enables UCP to detect dead instance and helps in transferring the work load to the surviving active instance as soon as the unplanned down event occurs. Enable

6 http://www-content.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

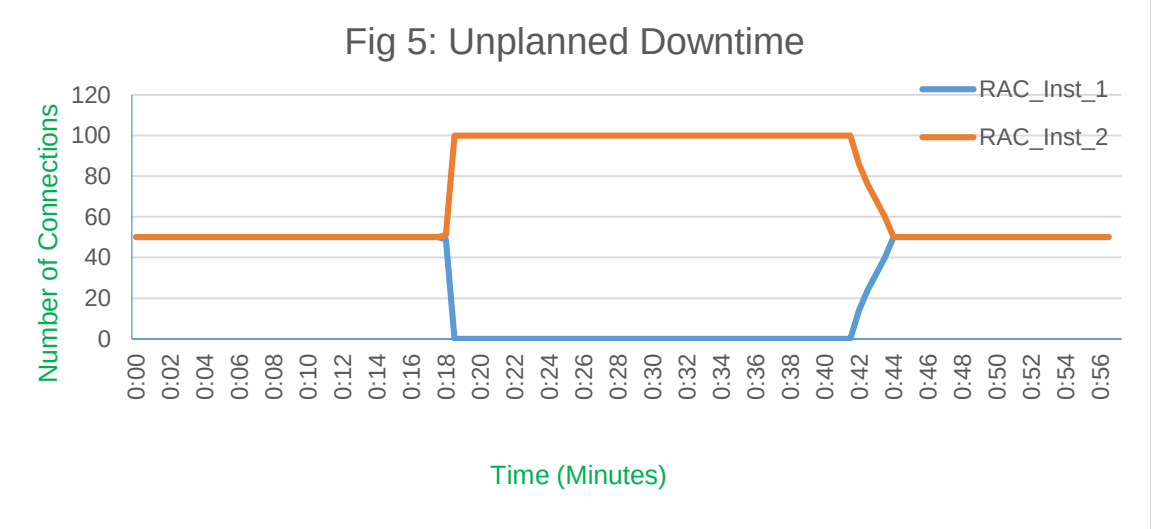
Transaction Guard and Application Continuity to achieve continuous service without any interruption of in-flight work. Please refer to the white paper **Java Programming with Oracle Database 12c RAC and Active Data Guard**[7] for understanding how TG and AC will protect your application from unplanned downtimes.

DBA or RDBMS Steps

To simulate Fast Connection Failover, the DBA may either stop the service on one instance with –force option (as specified hereafter) or, alternatively, kill the Oracle instance SMON background process. An even more drastic approach consists in powering down of one of the nodes supporting the database.

```
$srvctl stop service –db <db_name> -service <service_name> -instance <instance_name> -
force
```

Figure 5, shows connections distribution of XYZ service across two RAC instances before and after unplanned downtime. Notice that the connection workload goes from fifty-fifty across both instances to hundred-zero. In other words, the remaining instances sustain the workload without disrupting the business operation.



Runtime Load Balancing (RLB) with WebSphere Servlets

Runtime Connection Load Balancing enables routing of work requests across RAC or ADG instances to achieve predictable runtime performance. RAC and GDS post runtime load balancing advisories every 30 seconds. UCP uses the Load Balancing advisory to balance the work across RAC instances, dynamically and thereby achieving best scalability. Runtime Load Balancing comes also into play when new node(s)/instance(s) are added/removed to/from the service; the work load gets balanced in both situations without any manual intervention.

7 http://www-content.oracle.com/technetwork/database/application-development/12c-ha-concepts-2408080.pdf

Developer or Web Application steps

Web applications need to set the UCP property *'setFastConnectionFailover'* to true as already described (refer to *"Fig 2.4:Enabling FCF"* for more details) to allow receiving FAN Load Balancing advisories. UCP dispenses connections from the least loaded database instance (in RAC or GDS environments). Ultimately the workload is uniformly spread across the databases in question (RAC or GDS).

DBA or RDBMS steps

Configure the Oracle RAC Load Balancing Advisory with the following values.

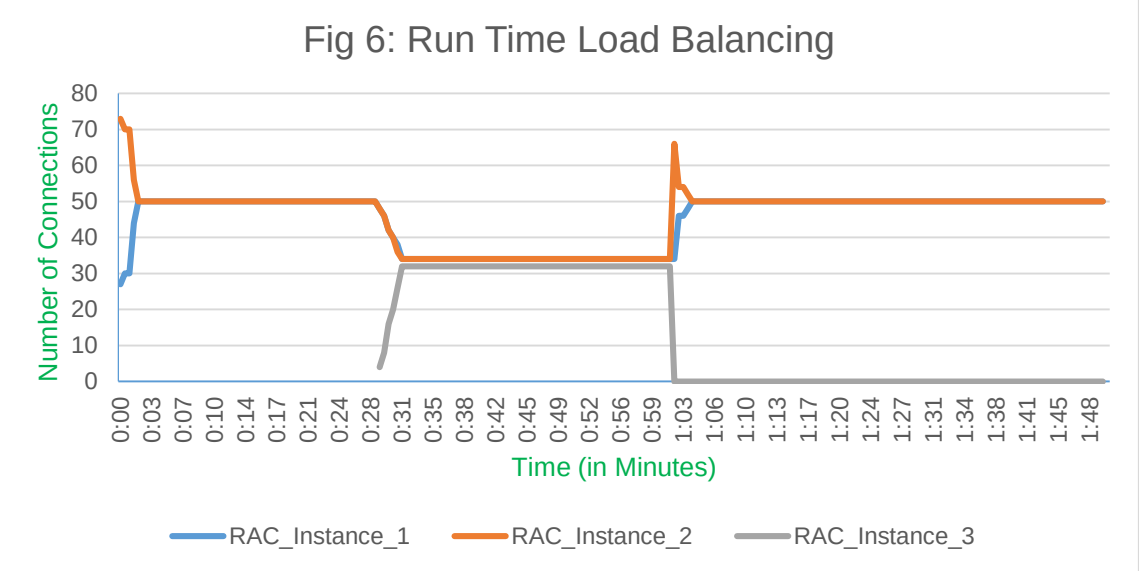
Set 'Runtime Load Balancing Goal' to SERVICE_TIME or THROUGHPUT

```
$srvctl modify service -db <db_name> -service <service_name> -rlbgoal SERVICE_TIME
$gdsctl modify service -db <db_name> -service <service_name> -rlbgoal SERVICE_TIME
```

Set 'Connection Load Balancing Goal' to SHORT

```
$srvctl modify service -db <db_name> -service <service_name> -clbgoal SHORT
$gdsctl modify service -db <db_name> -service <service_name> -clbgoal SHORT
```

Figure 6, shows connections distribution of XYZ service across three RAC instances. Notice that the workload is gradually distributed across the available instances with 50-50 connections each between RAC_Instance_1 and RAC_Instance_2. When a new instance, RAC_Instance_3 is added, the load will be re-distributed evenly to 34-34-32. After some time, RAC_Instance_3 is removed, UCP gradually rebalances the load between the remaining instances and in this case, achieves 50-50 connection workload distribution.



Appendix

Enable JDBC & UCP logging for debugging

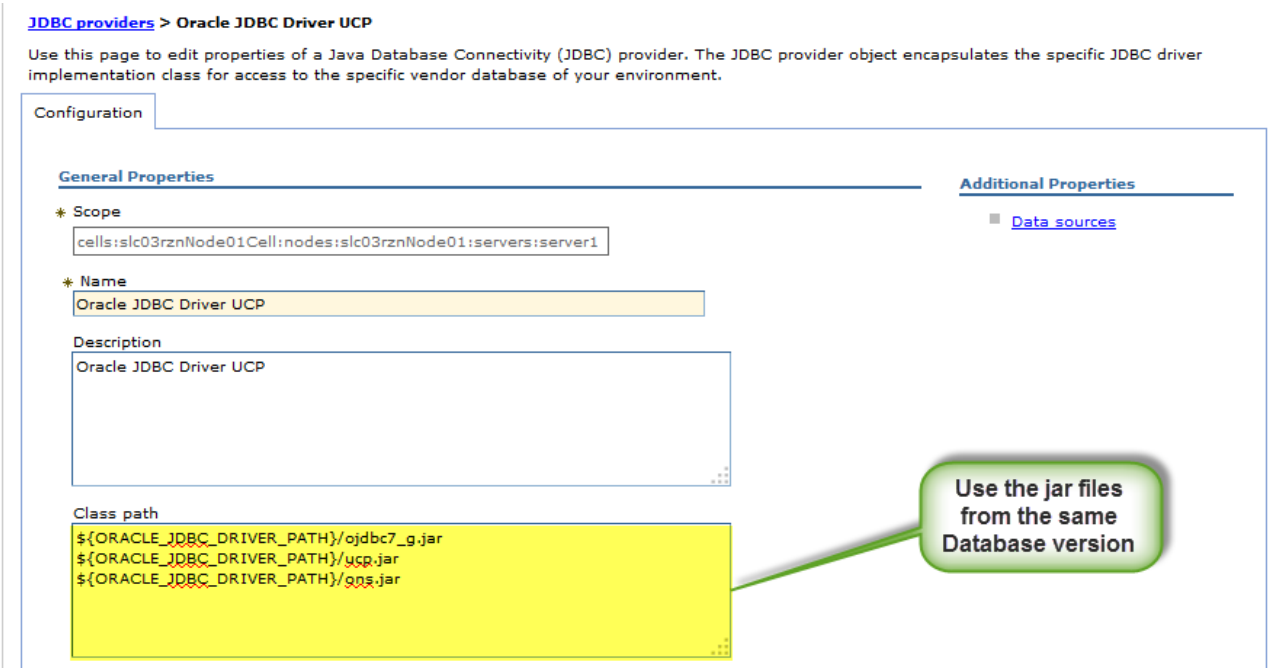
Enable JDBC & UCP logging when there are issues. This helps to debug and find the root cause of the problem.

There are few steps for enabling JDBC & UCP logging.

- Configure debug jar in the classpath
- Enable logging
- Setup a config file for advanced logging

Configure debug jar in the classpath:

Make sure to have **ojdbc7_g.jar** in the classpath under JDBC&UCP provider created as shown below.



Enable logging

In order to get any log output from the Oracle JDBC drivers you must enable logging. Enable logging by setting the system property -D**oracle.jdbc.Trace = TRUE**. This turns logging ON. Refer to *Fig 5. Enable JDBC/UCP Logging in WebSphere.*

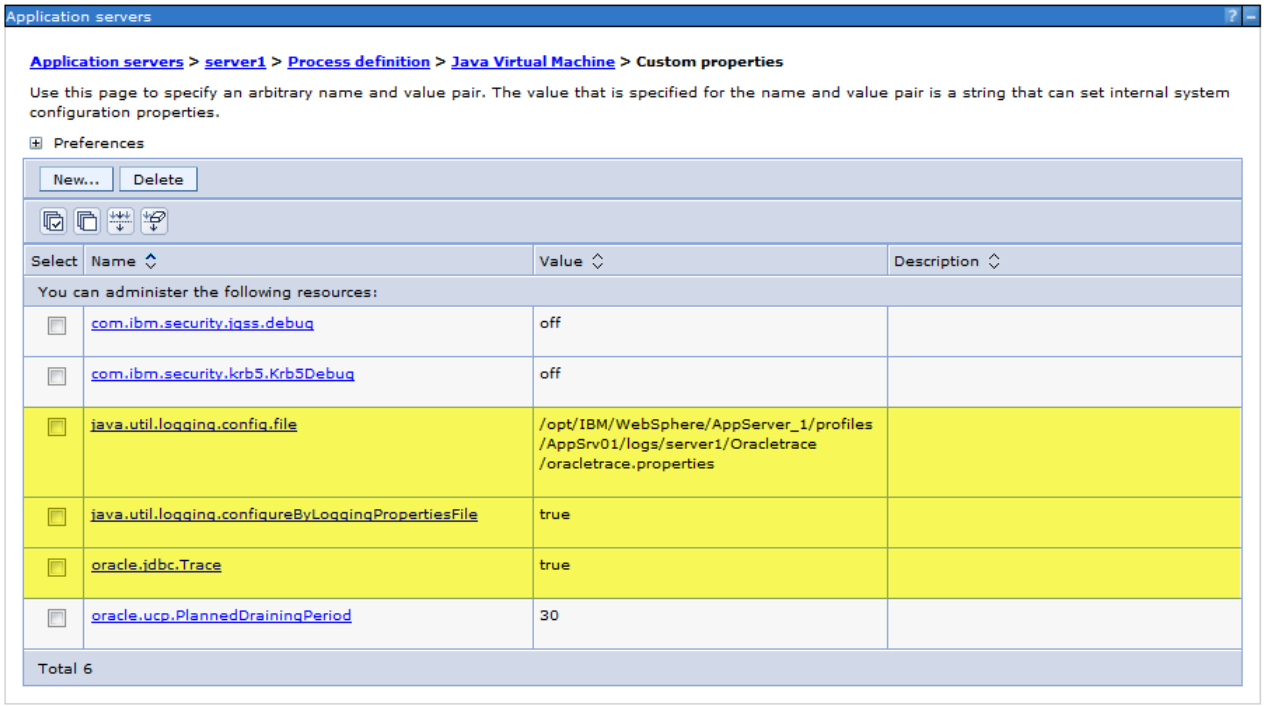
Setup a config file for advanced logging

Create a configuration file, for example oracletrace.properties and insert the following and save the file.
Enable the config file by setting the system properties **–Djava.util.logging.config.file =<localtion of the config file>.** Refer to *Fig 5. Enable JDBC/UCP Logging in WebSphere.*

```
# FOR UCP logs
.level=WARNING
oracle.ucp.jdbc.oracle.level=FINEST
oracle.ucp.jdbc.level=FINEST
```

oracle.ucp.common.level=FINEST
oracle.ucp.jdbc.oracle.rlb.level=FINEST
# For JDBC Driver logs
level=SEVERE
oracle.jdbc.level=ALL
oracle.jdbc.driver.level=FINEST
oracle.jdbc.pool.level=FINEST
oracle.jdbc.util.level=OFF
oracle.jdbc.handlers=java.util.logging.FileHandler
java.util.logging.FileHandler.level=FINE
java.util.logging.FileHandler.pattern=jdbc.log
java.util.logging.FileHandler.count=1
java.util.logging.FileHandler.formatter=java.util.logging.SimpleFormatter

Fig.5: Enable JDBC/UCP Logging in WebSphere



Conclusion

This paper furnishes a comprehensive and practical coverage of high-availability and load balancing in of WebSphere web applications with Oracle Database 12c; more specifically how to design Web applications and configure the RDBMS, UCP and the WebSphere container for resiliency to planned, and unplanned database downtimes and workload balancing. The steps described in this paper are valid for all Oracle Database 12c high availability and scalability configurations including RAC, RAC One and Active Data Guard. The complete UCP WebSphere demo referenced in this paper will be posted on https://github.com/oracle/jdbc-ucp. Java architects, Web application designers and DBAs may now design robust and reliable WebSphere Web applications for better user experience and application continuity.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together

Oracle is committed to developing practices and products that help protect the environment