Oracle Reports: Tips and Techniques

An Oracle White Paper
October 2006

Oracle Reports: Tips and Techniques

Introduction

This paper shows you how to use certain features of Oracle Reports to maximize benefits from the report development and publishing experience, keeping in mind the current multi-tier and Web-based deployment requirements of Oracle Reports customers.

Successful businesses know that presenting their data in a timely and meaningful way provides a powerful advantage over the competition. To that end, businesses continue to need ever-more-powerful tools for producing high-quality reports from the masses of disparate data sources kept in every major corporation today. Oracle Reports is a powerful high fidelity reporting tool that is used for dynamic, enterprise-level reporting. It enables businesses to develop and deploy information to all levels within and outside of the organization.

For many years, Oracle Reports has maintained its position as a premier enterprise-reporting tool by keeping pace with rapid shifts in technology. As technology has changed, Oracle Reports has moved from character-based, to graphical-based, to client/server, to multi-tier Web-based development and deployment. The latest release, Oracle Reports 10*g* Release 2 (10.1.2), provides a highly stable and scalable enterprise-reporting solution, with several new features and enhancements to enrich your paper and Web reports.

This paper shows you how to use certain features of Oracle Reports to maximize benefits from the report development and publishing experience, keeping in mind the current multi-tier and Web-based deployment requirements of Oracle Reports customers. It provides tips on which Oracle Reports features you can use for creating professional reports, and what new techniques Oracle Reports offers to create sophisticated reporting systems.

This paper covers the following topics:

- Topic 1: Using OracleAS High Availability with Oracle Reports
- Topic 2: Using Special Features of Graphs
- Topic 3: Using SecurePDF pluggable Destination to Create Password-Protected PDF
- Topic 4: Adding Special Effects to HTMLCSS Reports Using CSS Pseudo-Elements and Pseudo-Classes
- Topic 5: Handling Error Conditions in JavaServer Page (JSP) Reports
- Summary

Accessing the Example Reports

You can access the example reports and other files needed to re-create the demonstrations at the following location on the Oracle Technology Network (OTN):

http://www.oracle.com/technology/products/reports/OOW/2006

NOTE: You must unzip the example zip file into a folder (for example, **`C:\Temp\oow2006)`**. This paper refers to this folder as **`OOW_FOLDER`**.

Release Information

Oracle Reports 10*g* Release 2 (10.1.2) is available in:

- Oracle Application Server 10*g* Release 2 (10.1.2), Enterprise Edition
- Oracle Business Intelligence 10*g* Release 2 (10.1.2), Standard Edition
- Oracle Application Server Forms and Reports Services 10*g* Release 2 (10.1.2)

Topic 1: Using OracleAS High Availability with Oracle Reports

Oracle Reports' integration with OracleAS High Availability ensures that your enterprise-reporting environment is extremely reliable and fault-tolerant.

Oracle Application Server (OracleAS) High Availability provides the industry's most reliable, resilient, and fault-tolerant application server platform. Oracle Reports' integration with OracleAS High Availability ensures that your enterprise-reporting environment is extremely reliable and fault-tolerant. With Reports Server clustering deprecated in 10*g* Release 2 (10.1.2), it is recommended that you switch to OracleAS High Availability if you are using Reports Server clustering.

This demonstration shows you how to set up an OracleAS High Availability environment, and how OracleAS Reports Services leverages this environment.

This topic covers the following sections:

- Prerequisites
- Demonstration
- Summary

NOTE: The setup used in this demonstration is for illustration only, and should not be used in a production environment. To set up OracleAS High Availability in your production environment, follow the instructions in *Oracle Application Server High Availability Guide 10g Release 2 (10.1.2),* available on the Oracle Technology Network (OTN).

Prerequisites

You should have access to:

- 3 host machines for an installation of Oracle Application Server on each machine.
- Oracle Metalink (http://metalink.oracle.com) for downloading Automated Release Updates.
- Human Resources (HR) sample schema of Oracle Database 10*g*.

Additionally, refer to "Accessing the Example Reports" on page 4.

Demonstration

This section describes how to perform the following steps:

- Install the Software
- Configure OracleAS Web Cache to Act Solely as a Software Load Balancer

- Configure Origin Servers
- Create the Site Definition
- Map the Site Definition to the Origin Servers
- Apply All Changes
- Direct All Requests Through the Load Balancer
- Make Report Definition Files Available to the Origin Servers
- Submit Request for a Simple Report
- Submit Request for a Simple Report with a Parameter Form
- Perform Failover Test

Install the Software

1. Install two instances of Oracle Application Server Forms and Reports Services 10*g* Release 2 (10.1.2), preferably each on a different host machine.
2. Install Oracle Application Server Web Cache Standalone 10*g* Release 2 (10.1.2), preferably on a third host machine.

Figure 1 shows the architecture that will be set up in this demonstration. In the figure, the OracleAS Web Cache installation is performed on the host **`apps.mycompany.com`**, while the OracleAS Forms and Reports Services installations are performed on the hosts **`host1.mycompany.com`** and **`host2.mycompany.com`**.

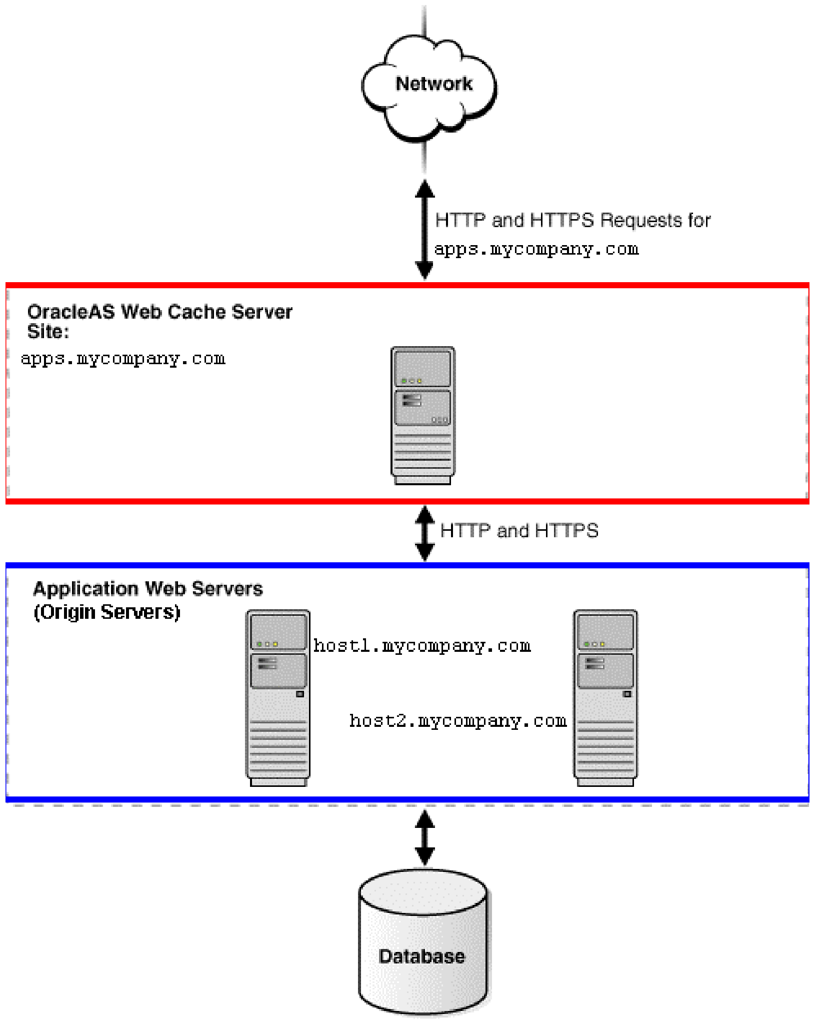
In the sections that follow, we will set up **`apps.mycompany.com`** to act as the load balancer. It will direct all incoming requests to the origin servers **`host1.mycompany.com`** and **`host2.mycompany.com`** in a round-robin fashion. The Reports Servers running on each of these origin servers (**`host1.mycompany.com`** and **`host2.mycompany.com`**) will process the report requests.

NOTE: In this demonstration, we will refer to each of the 3 host machines using the following convention:

- **`apps.mycompany.com`**: Host machine where you have installed OracleAS Web Cache.
- **`host1.mycompany.com`**: Host machine where you have installed OracleAS Forms and Reports Services instance 1.
- **`host2.mycompany.com`**: Host machine where you have installed OracleAS Forms and Reports Services instance 2.

You should replace these names with your actual host names in all the steps in this demonstration.

Figure 1: Architecture used in this demonstration



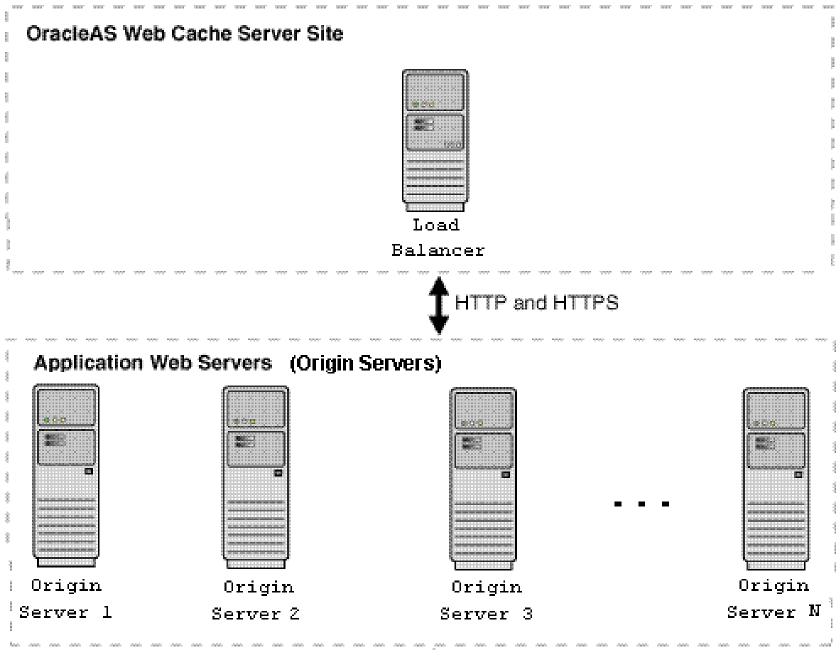
Here is an explanation of the terms used in this topic:

Load balancer: A server that distributes HTTP and HTTPS requests among origin servers so that no single origin server is overloaded. In this demonstration, the host with OracleAS Web Cache installation (**`apps.mycompany.com`**) will act as load balancer.

Origin server: One of potentially multiple servers that process the requests distributed by the load balancer. Any server that is an application Web server can act as an origin server. In this demonstration, the hosts with OracleAS Forms and Reports Services installations (**`host1.mycompany.com`** and **`host2.mycompany.com`**) will act as origin servers.

Figure 2 shows the load balancer and the origin servers.

Figure 2: Load balancer and Origin Servers



Configure OracleAS Web Cache to Act Solely as a Software Load Balancer

All the steps in this section must be performed on the host where OracleAS Web Cache is installed (**`apps.mycompany.com`**). To configure a single OracleAS Web Cache server as a software load balancer:

1. Download an Automated Release Update (ARU) for bug 4569559.

   You can download ARUs from Oracle Metalink:

   http://metalink.oracle.com

   NOTE: This patch resolves some stability issues with OracleAS Web Cache 10*g* Release 2 (10.1.2) when configured as a software load balancer.

2. Follow the instructions provided in the readme file for the patch.

3. Create a backup copy of the **`internal.xml`** file. This file is located in the **`$ORACLE_HOME/webcache`** directory on UNIX and **`%ORACLE_HOME%\webcache`** directory on Windows.

4. Use a text editor to open the **`internal.xml`** file.

5. Locate the **`CALYPSOINTERNALPARAMS`** element:

```
 ...
   <CALYPSOINTERNALPARAMS>
     <HEURISTICS CATELMFACTOR="0.0"/>
     <CACHE/>
     <SEARCHKEY/>
     <INVALIDATION/>
     <MEMORYMANAGER/>
     <PPC/>
     <MISCELLANEOUS/>
     <OEMPERFTOOL/>
```

```
  </CALYPSOINTERNALPARAMS>
...
```

6. Add the **`LOADBALANCE`** sub-element directly after the **`OEMPERFTOOL`** sub-element, as follows:

   ```
   ...
     <CALYPSOINTERNALPARAMS>
       <HEURISTICS CATELMFACTOR="0.0"/>
       <CACHE/>
       <SEARCHKEY/>
       <INVALIDATION/>
       <MEMORYMANAGER/>
       <PPC/>
       <MISCELLANEOUS/>
       <OEMPERFTOOL/>
       <LOADBALANCE ON="YES"/>
     </CALYPSOINTERNALPARAMS>
   ...
   ```

7. Save **`internal.xml`**.
8. Restart OracleAS Web Cache with the following command:

   ```
   webcachectl restart
   ```

9. Confirm that OracleAS Web Cache is running in the load balancer mode from the OracleAS Web Cache Manager by verifying the following status message displayed beneath the Apply Changes and Cancel Changes buttons:

   ```
   Web Cache running in Load Balancer Mode with current
   configuration
   ```

   NOTE: You can access the OracleAS Web Cache Manager by typing the following URL in the browser, and providing your login credentials:

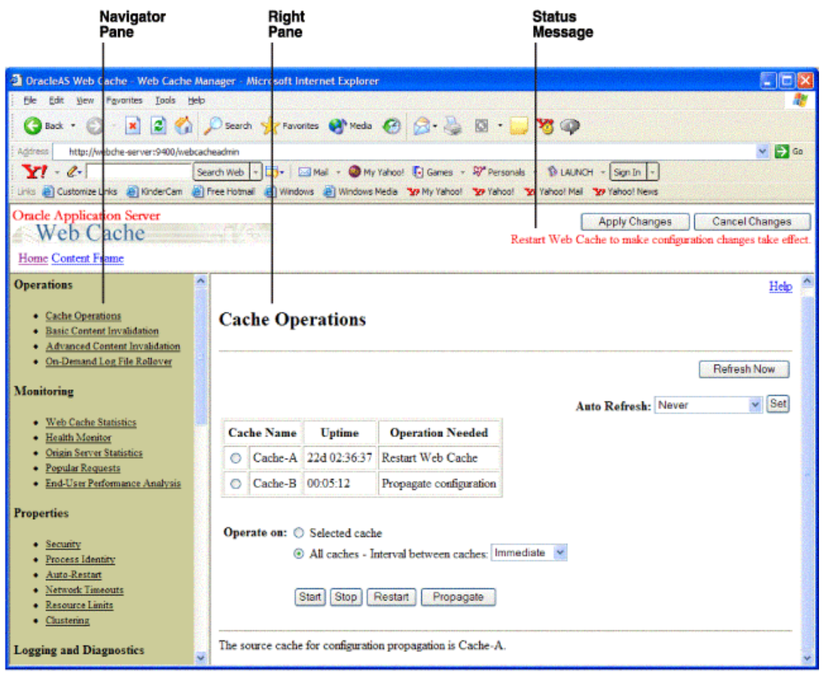
   `http://`*`web_cache_hostname`*`:`*`web_cache_port`*`/webcacheadmin`

   By default, OracleAS Web Cache listens for administrative requests on port 9400. If this port is in use, the installation procedure attempts to assign other port numbers from a range from 9400 to 9499.

   To log in to the OracleAS Web Cache Manager, enter the OracleAS Web Cache administrator user name (**`administrator)`** and password. The password is the one you supplied during the installation.

   Figure 3 shows the OracleAS Web Cache Manager interface.

Figure 3: OracleAS Web Cache Manager Interface



Restart Web Cache to make configuration changes take effect.

Cache Operations

Auto Refresh: Never

	Cache Name	Uptime	Operation Needed
○	Cache-A	22d 02:36:37	Restart Web Cache
○	Cache-B	00:05:12	Propagate configuration

Operate on: ○ Selected cache [✓ All caches - Interval between caches: Immediate]

The source cache for configuration propagation is Cache-A.

Configure Origin Servers

In this section, we will configure **`host1.mycompany.com`** and **`host2.mycompany.com`** (the host machines shown inside the blue box in Figure 1) as origin servers for the load balancer. Any server that is an application Web server for internal sites can act as an origin server. The load balancer can then forward all incoming requests to the origin servers.

All the steps in this section must be performed using the OracleAS Web Cache Manager, which is running on the host where OracleAS Web Cache is installed (**`apps.mycompany.com`**).

To configure OracleAS Web Cache with origin server information from OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Origin Servers**.

   The Origin Servers page displays.

2. Click **Add** in the **Application Web Servers** section.

   The Add Application Web Server dialog box displays.

3. In the **Hostname** field, enter the host name of the origin server (**`host1.mycompany.com`**).

4. In the **Port** field, enter the listening port from which the origin server will receive OracleAS Web Cache requests. In a default OracleAS installation, this port number is 80.

5. In the **Routing** field, select ENABLE to permit OracleAS Web Cache to route requests to the origin server.

6. In the **Capacity** field, enter the maximum number of concurrent connections that the origin server can accept. You determine this number by load testing the origin server until it runs out of CPU, responds slowly, or until a backend database reaches full capacity. For this demonstration, enter 10.

7. In the **Failover Threshold** field, enter the number of allowed continuous read/write failures with an origin server on established connections. For this demonstration, enter 5. After the failover threshold is met, OracleAS Web Cache considers the server down and uses other servers for future requests.

8. In the **Ping URL** field, enter the URL that OracleAS Web Cache will use to poll an origin server that has reached its failover threshold. For this demonstration, enter `"/"` (without the quotes).

9. In the **Ping Interval** (seconds) field, enter the time, in seconds, that OracleAS Web Cache will poll an origin server that has reached its failover threshold. For this demonstration, enter 10.

10. From the **Protocol** list, select HTTP to send HTTP requests on the port.

11. Click **Submit**.

This configures **`host1.mycompany.com`** as one origin server. To configure **`host2.mycompany.com`** as the other origin server, repeat all steps in this section, specifying **`host2.mycompany.com`** as the origin server name in step 3.

Create the Site Definition

OracleAS Web Cache only forwards requests to a configured origin server if the origin server is mapped to a Web site. In this section, we will create a Web site **`apps.mycompany.com`** (the host machine shown inside the red box in Figure 1). In the next section, we will create one site-to-server mapping that maps both our origin servers to the site **`apps.mycompany.com`**.

All the steps in this section must be performed using the OracleAS Web Cache Manager, which is running on the host where OracleAS Web Cache is installed (**`apps.mycompany.com`**).

To create new site definitions from OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**.

   The Site Definitions page displays.

2. Notice that one Web site has already been defined (**`apps.mycompany.com`**). Since the site is already defined, we do not need to perform any configuration.

Map the Site Definition to the Origin Servers

In this section, we will map the Web site definition (**`apps.mycompany.com`**) to the origin servers (**`host1.mycompany.com`** and **`host2.mycompany.com`**) that we configured in the earlier sections. Once this is done, all incoming requests to the Web site will be forwarded to one of the origin servers in a round-robin fashion.

All the steps in this section must be performed using the OracleAS Web Cache Manager, which is running on the host where OracleAS Web Cache is installed (**`apps.mycompany.com`**).

To map sites to origin servers from OracleAS Web Cache Manager, perform the following steps:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mapping**.

   The Site-to-Server Mapping page displays.

2. Notice that some mappings already exist. Select the first mapping, and then click **Insert Above**.

   The Create Site-to-Server Mapping dialog box displays.

3. In the **Edit Site Name** section, select the **Select from Site definitions** option. Make sure that **`apps.mycompany.com:80`** is selected in the **Host Name : Port Number** drop-down list.

4. In the **Select either application Web servers or proxy servers to which this Site is mapped** section, select the **Select Application Web Servers** option. Select both **`host1.mycompany.com`** and **`host2.mycompany.com`**.

5. Click **Submit**.

6. The site-to-server mapping you created is shown in the mappings list.

7. Delete all the remaining mappings by selecting them individually, and clicking **Delete Selected**.

Apply All Changes

All the steps in this section must be performed using the OracleAS Web Cache Manager, which is running on the host where OracleAS Web Cache is installed (`apps.mycompany.com`).

To apply all configuration changes you have made so far from OracleAS Web Cache Manager, perform the following steps:

1. Click **Apply Changes** at the top of the OracleAS Web Cache Manager.

   The Cache Operations page displays.

2. Click **Restart**.

   The Success dialog box displays.

3. Click **OK**.

Direct All Requests Through the Load Balancer

We have configured the load balancing environment; however, the users can still access the origin servers directly. Even if the origin servers are protected behind a firewall, some of the report requests may access the origin servers directly. For example, when you submit the default paper Parameter Form for a report, it always accesses the origin server that displayed the Parameter Form. Ideally, we would like to avoid direct access to the origin servers, and accept requests only through the load balancer.

There are multiple ways to make sure that all requests go through the load balancer:

- Protect the origin servers with a firewall.
- Code your applications (for example, the Parameter Forms) to access the load balancer rather than accessing the origin server.
- If any request lands up directly at the origin server, redirect it to the load balancer.

In production scenarios, the solution will perhaps be a combination of the above options. In this demonstration, we will redirect all requests from the origin servers to the load balancer. To do this, follow these steps:

1. Log in to the Enterprise Manager OracleAS Control for `host1.mycompany.com`.
2. Navigate to **HTTP Server** > **Administration** tab > **Advanced Server Properties** > **httpd.conf**.
3. Under **Edit httpd.conf,** you will see the contents of the `httpd.conf` file.
4. Locate the following code:

   ```
   ServerName host1.mycompany.com
   ```

5. Change it to the following:

   ```
   ServerName apps.mycompany.com
   ```

6. Click **Apply**.

7. The Confirmation page displays. Click **Yes** to restart the HTTP Server.

Repeat the above steps using the Enterprise Manager OracleAS Control for **`host2.mycompany.com`**.

Make Report Definition Files Available to the Origin Servers

To make the report definition files available to the origin servers, perform the following steps:

1. Copy the files under the folder **`OOW_FOLDER\HA\source\host1`** to the following folder on **`host1.mycompany.com:`**

   `ORACLE_HOME`**`\reports\samples\demo`**

   where ***`ORACLE_HOME`*** is the folder where OracleAS Forms and Reports Services is installed.

2. Copy the files under the folder **`OOW_FOLDER\HA\source\host2`** to the following folder on **`host2.mycompany.com:`**

   `ORACLE_HOME`**`\reports\samples\demo`**

   where ***`ORACLE_HOME`*** is the folder where OracleAS Forms and Reports Services is installed.

Submit Request for a Simple Report

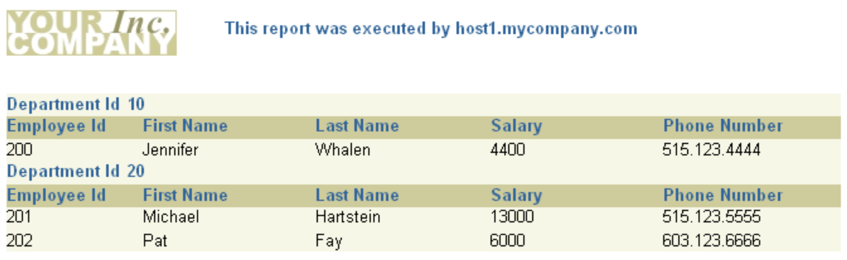
Now you are ready to test this HA environment by submitting report requests to the load balancer. Follow these steps:

1. Submit the following URL in the browser:

   `http://apps.mycompany.com:80/reports/rwservlet?report=Employees.rdf&userid=hr/hr@mydb&destype=cache&desformat=htmlcss`

2. The report output displays in the browser, as shown in Figure 4:

Figure 4: Output of report executed via the load balancer. The report is internally executed by `host1.mycompany.com`



This report was executed by host1.mycompany.com

Department Id 10

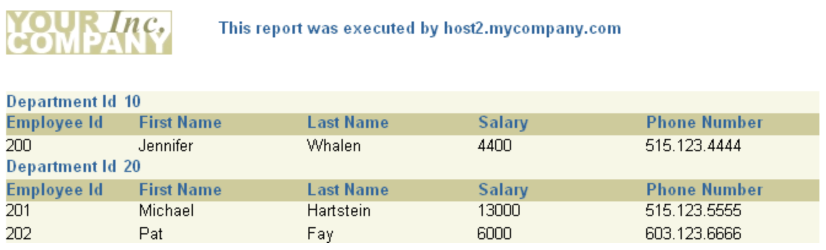
Employee Id	First Name	Last Name	Salary	Phone Number
200	Jennifer	Whalen	4400	515.123.4444

Department Id 20

Employee Id	First Name	Last Name	Salary	Phone Number
201	Michael	Hartstein	13000	515.123.5555
202	Pat	Fay	6000	603.123.6666

3. Refresh the browser contents. The report output displays as shown in Figure 5:

Figure 5: Output of report executed via the load balancer. The report is internally executed by `host2.mycompany.com`



This report was executed by host2.mycompany.com

Department Id 10

Employee Id	First Name	Last Name	Salary	Phone Number
200	Jennifer	Whalen	4400	515.123.4444

Department Id 20

Employee Id	First Name	Last Name	Salary	Phone Number
201	Michael	Hartstein	13000	515.123.5555
202	Pat	Fay	6000	603.123.6666

4. Notice the following:

   a. Report requests to **`apps.mycompany.com`** get forwarded to the origin servers.

   b. Different origin servers execute each of the two consecutive requests. This is because the load balancer forwards the requests to each of the origin servers in a round-robin fashion.

      NOTE: OracleAS Web Cache actually forwards the requests based on the available free capacity on each of the origin servers. However, for the purposes of this demonstration the available free capacity is equal for both the origin servers. This results in the load balancer forwarding the requests in a round-robin fashion.

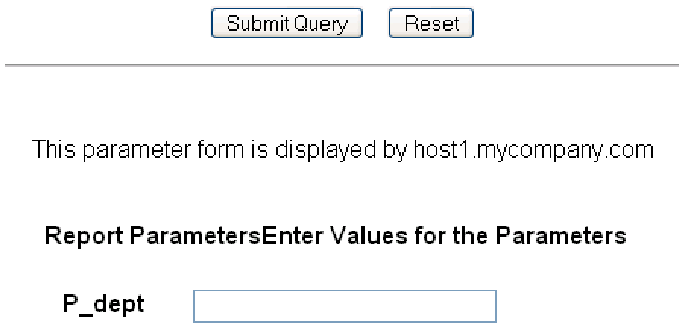
Submit Request for a Simple Report with a Parameter Form

1. Submit the following URL in the browser:

   `http://apps.mycompany.com:80/reports/rwservlet?report=Employees_pform=hr/hr@`*`mydb`*`&destype=cache&desformat=htmlcss&paramform=yes`

2. The Parameter Form displays in the browser, as shown in Figure 6:

   Figure 6: The Parameter Form, which is internally executed by `host1.mycompany.com`



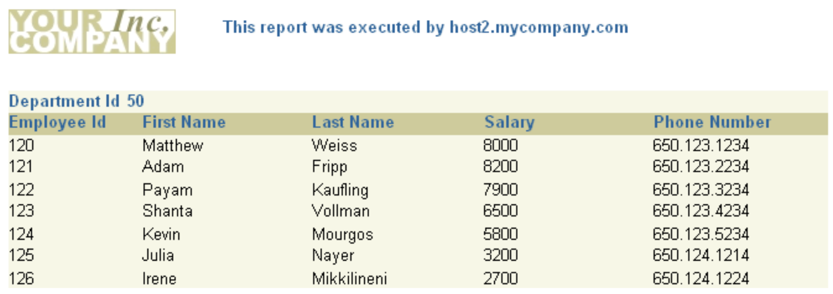
This parameter form is displayed by host1.mycompany.com

Report ParametersEnter Values for the Parameters

P_dept	

3. Enter the value of the parameter **`P_dept`** (for example, 50) and click **Submit Query**. The report output displays as shown in Figure 7:

   Figure 7: Output of the report after submitting the Parameter Form. The report is internally executed by `host2.mycompany.com`



This report was executed by host2.mycompany.com

Department Id 50

Employee Id	First Name	Last Name	Salary	Phone Number
120	Matthew	Weiss	8000	650.123.1234
121	Adam	Fripp	8200	650.123.2234
122	Payam	Kaufling	7900	650.123.3234
123	Shanta	Vollman	6500	650.123.4234
124	Kevin	Mourgos	5800	650.123.5234
125	Julia	Nayer	3200	650.124.1214
126	Irene	Mikkilineni	2700	650.124.1224

4. Notice that the report is executed by **`host2.mycompany.com,`** though the Parameter Form was displayed by **`host1.mycompany.com`**. Ideally, we would like multiple sub-requests for a report to be executed by the same Reports Server; for example, both the Parameter Form and the actual report should be executed by the same Reports Server. To enable this, you need to configure session binding (also called session stickiness and session persistence) on the load balancer. Though OracleAS Web Cache offers this feature, we did not use it in this demonstration for the sake of simplicity. Refer to the *Oracle Application Server Web Cache Administrator's Guide* available on the Oracle Technology Network (OTN) for steps on how to enable session binding in OracleAS Web Cache.

Perform Failover Test

We will now shut down one of the OracleAS instances, and see whether the load balancer is able to fail over all requests to the single available OracleAS instance.

1. Log in to the Enterprise Manager OracleAS Control for **`host1.mycompany.com`**.

2. Select **Web Cache**, and click **Stop**.

3. On the Confirmation page, click **Yes**.

4. Submit the following request in the browser:

   `http://apps.mycompany.com:80/reports/rwservlet?report=Employees.rdf&userid=hr/hr@`*`mydb`*`&destype=cache&desformat=htmlcss`

5. Notice that the request is executed by **`host2.mycompany.com`**.

6. Submit the above report request multiple times.

7. Notice that the request does not get forwarded to **`host1.mycompany.com`**. This host is perceived to be unavailable, because the Ping URL that we specified in the section "Configure Origin Servers" does not return a valid result. This means the load balancer is failing over all requests to the only available OracleAS instance **`host2.mycompany.com`**.

Summary

In this demonstration we saw:

- Configuring OracleAS Web Cache as a load balancer with two OracleAS Forms and Reports Services installations as the origin servers.
- Redirecting all requests through the load balancer by editing the **`httpd.conf`** file on each of the origin servers.
- The following features of the load balancer environment:
  - Distributing the load between the two origin servers.
  - Failing over to the available OracleAS instance in case one of the instances goes down.

For more information, refer to the following documents:

- *Oracle Application Server Best Practices Guide 10g Release 2 (10.1.2)* available on the Oracle Technology Network (OTN). Locate the chapter titled "Oracle Business Intelligence". In this chapter, locate the section titled "Oracle Application Server Reports Services".
- *Oracle Application Server High Availability Guide 10g Release 2 (10.1.2)* available on the Oracle Technology Network (OTN).
- *Changed Functionality Between Oracle Reports 6i and 10g* available on the Oracle Reports 10*g* Release 2 page on the Oracle Technology Network (OTN).

Topic 2: Using Special Features of Graphs

In this demonstration, we will take a look at some graphing features that you can set via XML to generate impressive graphs.

Graphs provide a very useful way of displaying information succinctly. Creating and editing graphs in Oracle Reports is done using the Graph Wizard, which offers a large selection of graph types as well as a lot of control over the look and feel of the graph. Oracle Reports internally uses another Oracle product called Business Intelligence Beans (BI Beans) for creating graphs.

Once you create a graph in Oracle Reports using the Graph Wizard, it is possible to modify the graph by re-entering the Graph Wizard. For simplicity, however, the Graph Wizard does not expose all the functionality available in the BI Beans. In order to use this extra functionality, you must access and modify the XML definition that stores the visual properties of the graph. This XML is stored in the report definition file as a property of the graph object. In the paper layout, you can access it in the Graph object's Property Inspector, and in the JSP-based Web layout you can access it directly in the JSP code.

In this demonstration, we will take a look at some graphing features that you can set via XML to generate impressive graphs.

This topic covers the following sections:

- Prerequisites
- Demonstration: Descending Bars
- Demonstration: Stock Graph
- Summary

Prerequisites

You should have access to the following:

- A test schema in Oracle Database where you can execute the provided SQL scripts (for example, the SCOTT schema).
- Reports Builder 10*g* Release 2 (10.1.2) to create and edit reports. Reports Builder is the report-building component of Oracle Developer Suite.

Also see the section "Accessing Example Reports".

Demonstration: Descending Bars

1. Open SQL*Plus. Connect to the test schema (for example, SCOTT schema). Execute the following script to create the necessary data:

   ```
   OOW_FOLDER\graphs\scripts\thefts_drop.sql
   ```

2. Open Reports Builder, and in the Welcome dialog box that displays, select **Build a New Report Manually**. If Reports Builder is already open, select **File** > **New** > **Report** > **Build a New Report Manually**.

3. In the Data Model of the report, create the following SQL query:

   ```
   select * from thefts_drop
   ```

4. Go to the **Paper Layout** view, and select **Insert** > **Graph**.

5. In the Graph Wizard that displays, select the following options, and leave the rest as default values:

   Type: Bar Graph

   Category: Month

   Data: Thefts_Drop

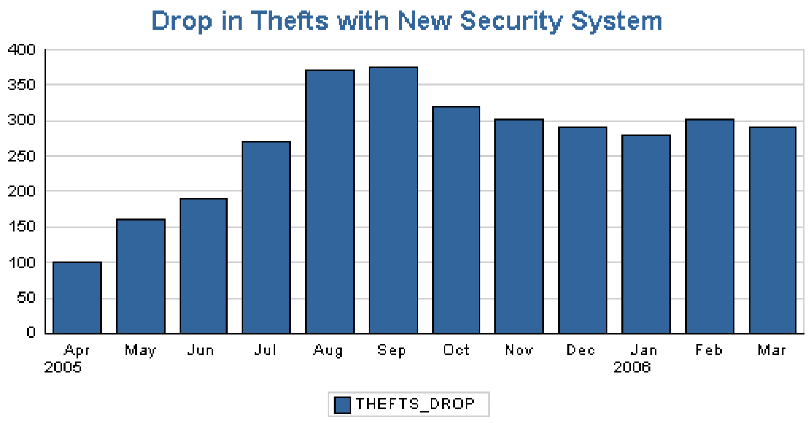
   Title: Drop in Thefts with New Security System

   Legend Location: Bottom

6. Click **Finish**.

7. Run the report to paper layout. The graph displays as shown in Figure 8:

Figure 8: Graph showing drop in thefts



8. As you can see, the graph shows a drop in thefts after the installation of a new security system. Since this is a *drop* in value, it makes sense to show the bars dropping down from the top axis rather than going up from the bottom axis. To do this, follow the steps below.

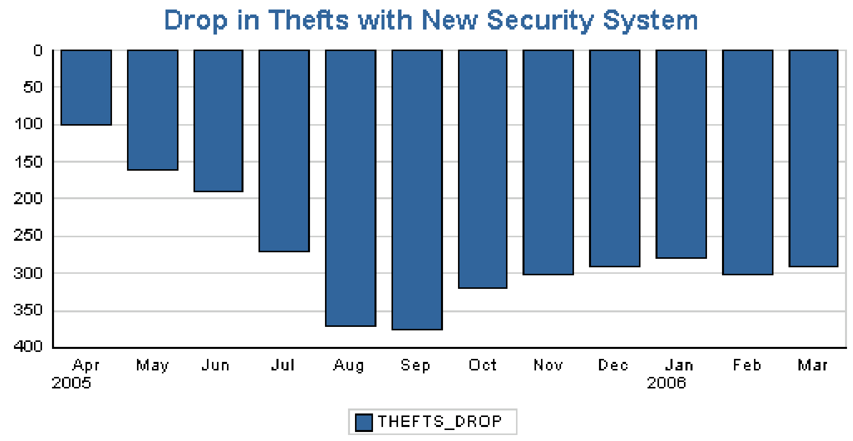
9. Right-click the graph object, and select **Property Inspector**.

10. In the graph Property Inspector, select **Graph Settings**, and click the browse button in this property.
11. In the Graph Settings dialog box that displays, locate the opening and closing `<Graph>` tags. The XML enclosed within these tags stores the visual properties for the selected graph object.
12. Add the following XML code anywhere within the graph XML:

    ```
    <Y1Axis ascending="false"/>
    ```

13. Click **OK**, and close the graph Property Inspector.
14. Run the report again to paper layout. The graph will display as shown below:

Figure 9: Graph showing bars dropping down from the top



15. As you can see, the above graph looks more intuitive for showing the *drop* in thefts.
16. Optionally, you can add some special effects like color gradients to the bars to make the graph look more impressive. To do this, follow the steps below.
17. Add the following code to the XML of the graph object:

    ```
    <SeriesItems>
    <Series id="0" borderColor="#666600">
    <SFX fillType="FT_GRADIENT" gradientNumStops="3">
     <GradientStopStyle stopIndex="0"
     gradientStopColor="#666600"/>
     <GradientStopStyle stopIndex="1"
     gradientStopColor="#FFFFFF"/>
     <GradientStopStyle stopIndex="2"
     gradientStopColor="#666600"/>
    </SFX>
    </Series>
    ```

18. Change the font color of the graph title to match the color of bars. To do this, locate the following code in the graph XML:

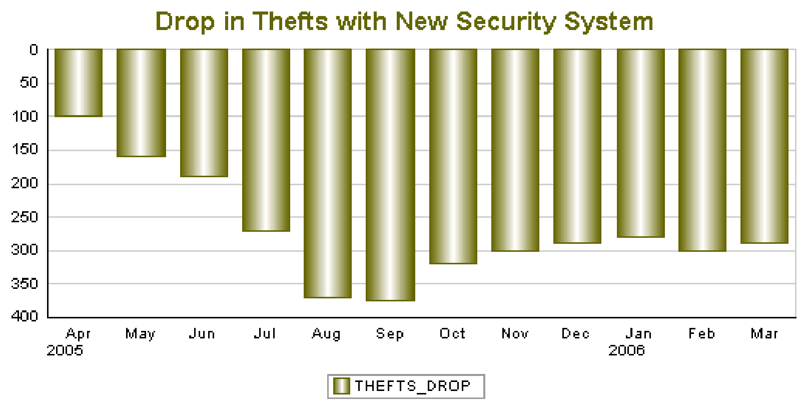
    ```
    <Title text="Drop in Thefts with New Security System"
    visible="true"/>
    ```

19. Change it to the following:

```
<Title text="Drop in Thefts with New Security System"
visible="true">
<GraphFont fontColor="#666600"/>
</Title>
```

20. Run the report again to paper layout. The graph will display as shown in Figure 10:

Figure 10: Graph output after adding some special effects



Demonstration: Stock Graph

1. Open SQL*Plus. Connect to the test schema, for example, SCOTT schema. Execute the following script to create the necessary data:

   `OOW_FOLDER\graphs\scripts\stock_data.sql`

2. Switch to Reports Builder, save any reports that may be open, and close them.

3. Select **File** > **New** > **Report** > **Build a New Report Manually**.

4. In the Data Model of the report, create the following SQL query:

   `select * from stock_data`

5. Go to the **Paper Layout** view, and select **Insert** > **Graph**.

6. In the Graph Wizard that displays, select the following options, and leave the rest as default values:

   Graph Type: Stock Graph

   Graph Subtype: Open-Hi-Lo-Close Candle

   X-axis Categories: Symbol

   Y-axis Data: Open_Price

   High_Price

   Low_Price

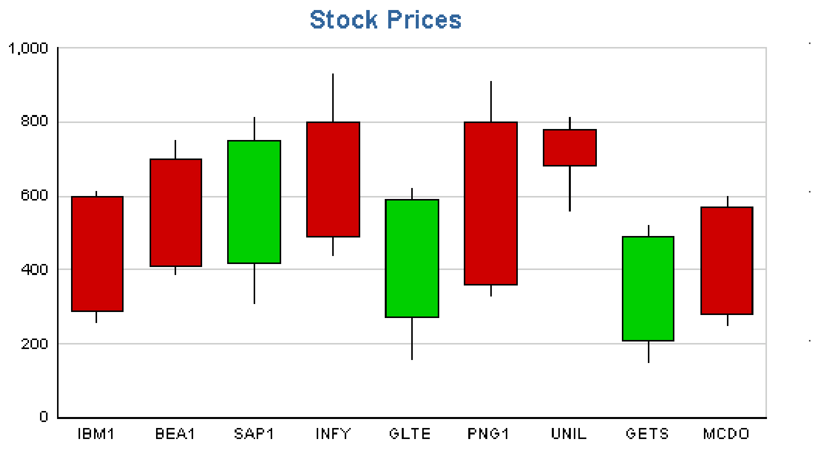
   Close_Price

Data: Match the high, low, open, and close prices with the corresponding columns.

Title: Stock Prices

7. Click **Finish**.
8. Run the report to paper layout. The graph will display as shown below:

Figure 11: Graph showing change in stock prices



9. The graph above shows the change in stock prices for different stocks. Here is how each stock is represented in this graph:
   a. Vertical line: The top and bottom ends of the vertical line represent and highest and lowest price respectively.
   b. Bar: The bar represents the opening and closing values. By default, the stocks that *lost* value are shown in red, while the stocks that *gained* value are shown in green.
10. You can add automatic color gradients to the bars. To do this, follow these steps:
11. Right-click the graph object, and select **Property Inspector**.
12. In the graph Property Inspector, select **Graph Settings**, and click the browse button in this property.
13. In the Graph Settings panel that displays, locate the opening and closing `<Graph>` tags. The XML enclosed within these tags stores the visual properties for the selected graph object.
14. Locate the following code in the graph XML:

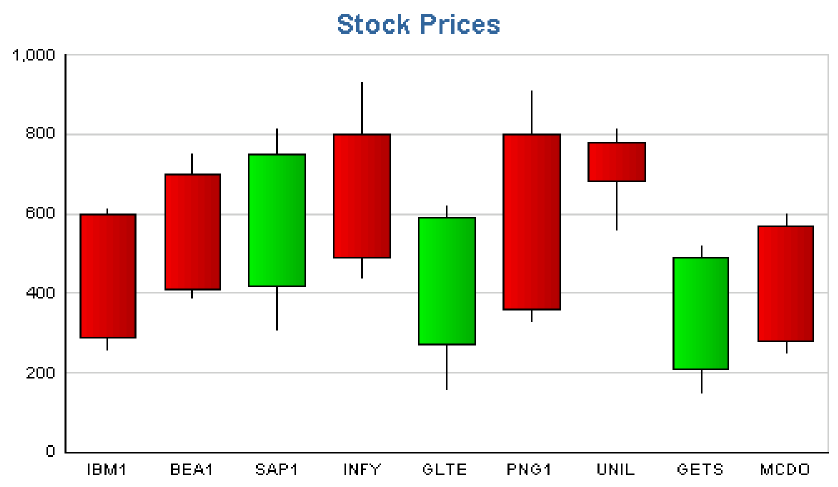
```
<Graph version="3.2.0.22" graphType="STOCK_OHLC_CANDLE">
```

15. Change it to the following:

```
<Graph version="3.2.0.22" graphType="STOCK_OHLC_CANDLE"
seriesEffect="SE_AUTO_GRADIENT">
```

16. Click **OK**, and close the graph Property Inspector.
17. Run the report to paper layout again. The graph will display as shown below. Notice that the bars use a color gradient.

Figure 12: Graph showing change in stock prices with color gradient



18. If necessary, you can change the colors shown in the graph; for example, to use colors approved in your corporate policy. To do this, open the graph Property Inspector and add the following XML code anywhere within the graph XML.

```
<StockMarker fallingColor="#B40000" risingColor="#009C00"/>
```

19. Change the font color of the graph title. To do this, locate the following code in the graph XML:

```
<Title text="Stock Prices" visible="true"/>
```

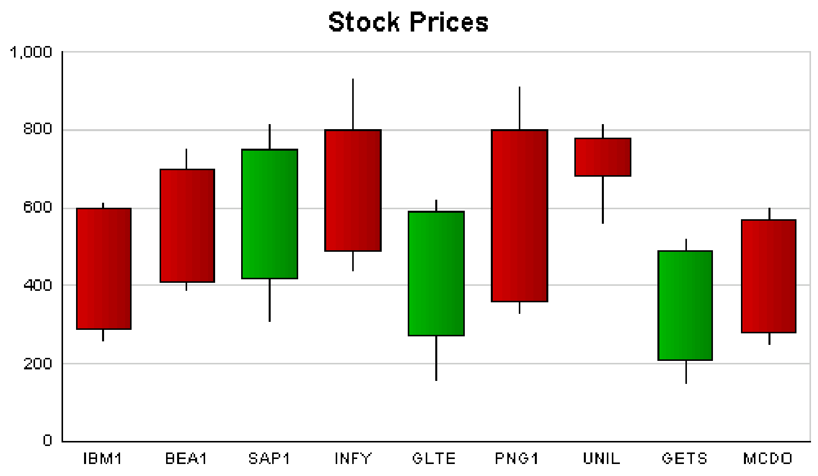
20. Change it to the following:

```
<Title text="Stock Prices" visible="true">
<GraphFont fontColor="#000000"/>
</Title>
```

21. Click **OK**, and close the graph Property Inspector.

22. Run the report to paper layout again. The graph will display as shown below. Notice that the graph uses darker shades of the same colors.

Figure 13: Graph showing change in stock prices with custom colors



Summary

In this demonstration we saw:

- How to create a graph that shows bars dropping down from the top, and how to add special effects to the bars.
- How to create a stock graph that shows change in stock prices, and how to change the colors of the bars.

For more information on graphing in Oracle Reports, refer to the following FAQ on the Oracle Technology Network (OTN):

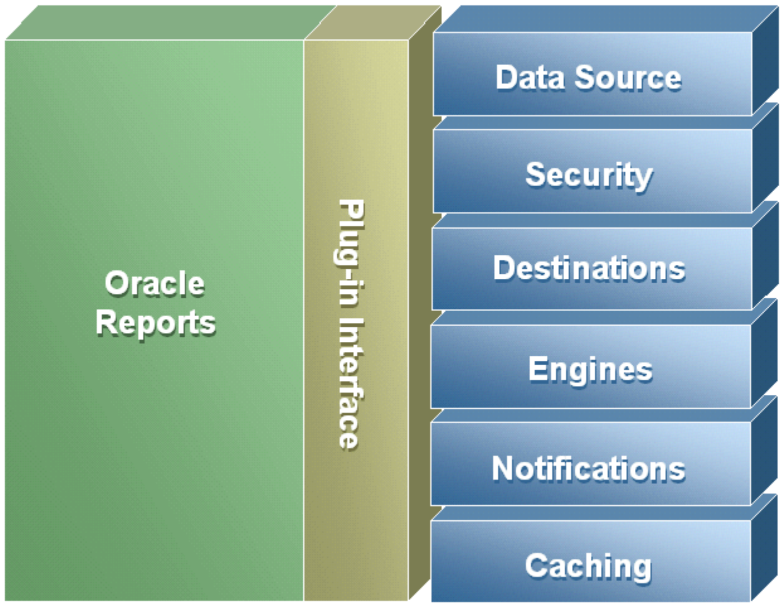
http://www.oracle.com/technology/products/reports/htdocs/faq/Graph_FAQ_with_style.html

Topic 3: Using SecurePDF Pluggable Destination to Create Password-Protected PDF

In this demonstration, we will use the SecurePDF pluggable destination to create password-protected PDF.

Oracle Reports exposes various plug-in interfaces that allow you to plug in your own data source, security mechanism, destination, engine, notification mechanism, and caching algorithm. Each of these plug-in interfaces is exposed via Oracle Reports' Java API.

Figure 14: Oracle Reports Pluggable Architecture



You can access Oracle Reports' Java API documentation and sample implementations at the Reports Software Development Kit (Reports SDK) available on the Oracle Technology Network.

Oracle Reports provides several out-of-the-box destinations such as file, printer, email, and so on. It is easy to extend this list of destinations by creating a pluggable destination of your own, and plugging it in to Oracle Reports using the Java API.

In this demonstration, we will use the SecurePDF pluggable destination to create password-protected PDF.

NOTE: This pluggable destination is not a part of the Oracle Reports production release, so it is only supported through the Oracle Reports discussion forum.

This topic covers the following sections:

- Prerequisites
- Demonstration
- Summary

Prerequisites

You should have access to the following:

- The HR (human resources) schema in Oracle Database 10*g*.
- Oracle Application Server 10*g* Release 2 (10.1.2).
- Third party software "PDF Linearization, Optimization and Privacy" (PLOP). This software is provided by PDFlib GmbH and is available at the following location:

  http://www.pdflib.com/products/plop/download.html

  NOTE: The PLOP command-line tool and library as offered for download from this page can be used as fully functional evaluation versions even without a commercial license. Using PLOP for production purposes requires a valid PLOP license.
- Adobe Acrobat Professional version 6.0 or later.

  NOTE: Earlier versions of Acrobat Professional may not support all the PDF security features available with PLOP.

Also see the section "Accessing Example Reports".

Demonstration

This section describes how to perform the following steps:

- Install and Configure
- Run Reports to the SecurePDF Destination
- Check the Specified Security Permissions
- Understand Limitations With This Sample

Install and Configure

Installing and Configuring PLOP

1. Download and install PLOP for your platform. For example, download and install **`PLOP-2.0.0p3-Windows.msi`** for Windows.
2. On Windows, copy the file **`plop_java.dll`** to **`C:\WinNT\System32`** or to any other folder that is in the system **`PATH`**. This file is available in **`<PLOP_install_folder>\bind\java`**

   For example:

   ```
   C:\Program Files\PDFlib\PLOP 2.0.0p3\bind\java
   ```

On Solaris and Linux, copy the file **`libplop_java.so`** to **`ORACLE_HOME/lib`**.

3. Copy the file **`plop.jar`** to **`ORACLE_HOME\reports\jlib`** folder. This file is available in **`<PLOP_install_folder>\bind\java`**.

Note that if you do not find the folder **`<PLOP_install_folder>\bind\java`**, then you may need to reinstall PLOP or modify the installation with the Additional Language Bindings option.

Installing the SecurePDF Destination

Copy the file **`SecurePdfDestination.jar`** to **`ORACLE_HOME\reports\jlib`** folder.

Make the PLOP jar (**`plop.jar`**) and the SecurePDF destination jar (**`SecurePdfDestination.jar`**) available to Oracle Reports classpath, as follows:

If you are using the in-process Reports Server, use the Oracle Enterprise Manager Application Server Control to modify the classpath:

1. In Oracle Enterprise Manager Application Server Control, display the detail page for your middle tier.
2. Under **System Components**, click **OC4J_BI_Forms**. On the **OC4J_BI_Forms** page, click **Applications**. Click the application named **Reports**.
3. On the **Applications** page, under **Properties**, click **General**.
4. Under **Library Paths**, add the following paths separately (if needed, click **Add Another Row** to add the paths):

   ```
   ORACLE_HOME/reports/jlib/plop.jar
   ```

   ```
   ORACLE_HOME/reports/jlib/SecurePdfDestination.jar
   ```

   Note that you can either add the absolute path or the relative path of the JAR. If you are adding the relative path of the JAR, make sure that the path is relative to the application deployment directory. For example,

   ```
   ../../../../jlib/SecurePdfDestination.jar
   ```

5. Click **Apply** and click **OK**.
6. On the **OC4J_BI_Forms** page click **Stop**, then **Start** to restart your application so that the new library paths take effect.

If you are using a standalone Reports Server:

1. Add the following entries to the **`REPORTS_CLASSPATH`** environment variable (on Windows the **`REPORTS_CLASSPATH`** is available in the registry, and on UNIX, the **`REPORTS_CLASSPATH`** is available in the shell script **`reports.sh`**):

   ```
   ORACLE_HOME/reports/jlib/plop.jar
   ```

   ```
   ORACLE_HOME/reports/jlib/SecurePdfDestination.jar
   ```

Registering the new destination with OracleAS Reports Services

To register the new destination with the OracleAS Reports Services , add the following entry in the Reports Server configuration file (**`ORACLE_HOME/reports/conf/`*`server_name`*`.conf`**):

```
<destination destype="SecurePDF"
class="oracle.reports.plugin.destination.securepdf.SecurePdfDestinat
ion">
</destination>
```

Restart Reports Server for the changes to take effect.

Run Reports to the SecurePDF Destination

The SecurePDF destination can be used with the command line keywords **`DESTYPE`** and **`DESNAME`** according to the following syntax:

```
...&DESTYPE=securepdf&DESFORMAT=pdf&DESNAME="<password>;<output_file
_name>;<security options separated by a whitespace each>"
```

For example:

```
...&DESTYPE=securepdf&DESFORMAT=pdf&DESNAME="plop;D:\myoutput.pdf;no
print nomodify"
```

Note: The security options have to be chosen from the following set of security settings

```
noprint nomodify nocopy noannots noforms noaccessible noassemble
nohiresprint
```

For this demonstration, run the report **`Employee.rdf`** with the following command:

```
http://host:port/reports/rwservlet?REPORT=Employee.rdf&USERID=scott/
tiger@mydb&DESTYPE=securepdf&DESFORMAT=pdf&DESNAME="plop;D:\myoutput
.pdf;noprint nocopy"
```

where:

`mydb` is the database connect string.

`noprint` means printing of the PDF document is disabled.

`nocopy` means selecting document contents and copying it to the clipboard for repurposing the contents is prohibited.

Check the Specified Security Permissions

To check whether the specified security permissions are enforced in the output PDF document, follow these steps:

1. Open the output file using Adobe Acrobat. In the password dialog box that displays, provide the password that you specified while running the report. The document will open only if the correct password is supplied.

2. Once the document in opened in Adobe Acrobat, press **Control-D**. The Document Properties dialog box displays.
3. Click the Security tab. Under Document Restrictions Summary, notice the following properties:

   ```
   Printing: Not allowed
   Content Copying and Extraction: Not Allowed
   ```

   Click **OK**.
4. To confirm that printing is not allowed, in the Adobe Acrobat menu, select **File**. Notice that the menu item **Print…** is disabled.
5. To confirm that content copying and extraction is not allowed, select **Tools** > **Basic** > **Select**. Select any text in the document and select **Edit** > **Copy**. Open a text editor, and select **Edit** > **Paste**. Notice that the selected text did not get copied.

   In Adobe Acrobat, select **File** > **Close** to close the document.
6. Further, to confirm that the document is encrypted, open the document in a text editor. Notice that the document content is not readable.

For more information on the security settings and the privileges associated with user and master passwords, refer to the PLOP Manual (available in **`<PLOP_install_folder>\doc`**).

Understand Limitations With This Sample

- If you are using the evaluation version of PLOP from PDFlib the user password should always be **`plop`**. The master password is the user password in uppercase (**`PLOP`**).
- Though this destination should work on all platforms for which PLOP is available, it has been tested only on Microsoft Windows 2000.
- Only the evaluation version of PLOP has been tested.
- The command line keyword DESNAME is used to specify the password. As a result, on rwservlet showjobs page, the password is visible under the values for the keyword DESNAME.
- The sample has been tested with PLOP version 2.1 (evaluation version). New PLOP versions in future may entail some changes, for example, new security options may be added or some of the current options may get removed. Check PLOP documentation for the latest information.

Summary

In this demonstration, we saw how to install and register a pluggable destination with Oracle Reports, using the example of SecurePDF destination that can be used to create password-protected PDF.

You can access Oracle Reports' Java API documentation and sample implementations at the Reports Software Development Kit (Reports SDK) available on the Oracle Technology Network. The SecurePDF pluggable destination can be downloaded at the Oracle Reports Plugin Exchange.

Topic 4: Adding Special Effects to HTMLCSS Reports Using CSS Pseudo-Elements and Pseudo-Classes

CSS introduces the concepts of pseudo-elements and pseudo-classes to permit formatting based on information that lies outside the HTML document tree. In this demonstration, we will use pseudo-classes and the custom style sheets feature of Oracle Reports 10*g* Release 2 (10.1.2) to define a custom look-and-feel for hyperlinks.

In Cascading Style Sheets specification 2 (CSS2), style is normally based on elements or attributes in the document tree. For example:

- Type selectors select any HTML element on a page that matches the selector, regardless of their position in the document tree. For example:

```
em {color: blue; }
```

- Class selectors select any HTML element that has a class attribute, regardless of their position in the document tree. For example:

```
.big {color: blue; }
```

- Descendant selectors select elements that are descendants of another element in the document tree. For example, the rule below will only select **`<em>`** elements that are descendants of **`<p>`** elements:

```
 p em {color: blue; }
```

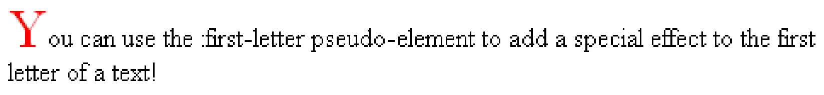
- Attribute selectors select elements based on their attributes or attribute value. For example, you may want to select any image on an HTML page that is called **`small.gif`**. This could be done with the rule below, which will only select images with the chosen name:

```
img[src="small.gif"] { border: 1px solid #000; }
```

This simple model is sufficient for many cases, but is not enough for all publishing scenarios. For example, you cannot add specific formatting to the *first line* of a paragraph, because in HTML 4.0, no element refers to it. CSS introduces the concepts of pseudo-elements and pseudo-classes to permit formatting based on information that lies outside the HTML document tree. For example:

- Pseudo-elements allow you to assign style to the first letter or first line of an element's content, as shown in Figure 15.

Figure 15: A pseudo-element allows style sheet designers to assign style to the first letter of the content


You can use the :first-letter pseudo-element to add a special effect to the first letter of a text!

- Pseudo-classes can be used to assign custom styles to hyperlinks. Pseudo-classes may be dynamic, in the sense that an element may acquire or lose a pseudo-class while a user interacts with the document. We will see an example of such a pseudo-class in this demonstration.

This topic covers the following sections:

- Prerequisites
- Demonstration
- Pseudo-Classes Used in This Demonstration
- Summary

Prerequisites

The pseudo-classes used in this example are supported by only the following browsers, so you need to have one of the following:

- Internet Explorer 4 or later
- Mozilla Firefox 1 or later
- Nestcape Navigator 8

Additionally, you should have access to the following:

- The SH (sales history) schema of Oracle Database 10*g*.
- Reports Builder 10*g* Release 2 (10.1.2) to create and edit reports. Reports Builder is part of Oracle Developer Suite.

Also see the section "Accessing the Example Reports".

Demonstration

1. Using SQL*Plus, create a table **`toc_multilevel`** with the following command:

```
    create table toc_multilevel (main_topic varchar2(100),
sub_topic varchar2(100), page number);
```

   This table will store the report topics and the corresponding page numbers, so that the report can display a table of contents with hyperlinks for easy navigation.

2. Open the report **`pseudo_class.rdf`** in Reports Builder. This report is available under the folder **`OOW_FOLDER\HTMLCSS`**.
3. Select **File** > **Connect** to connect to the SH schema of your database.
4. Select **File** > **Generate to File** > **HTMLCSS**.

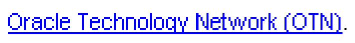
   NOTE: If you see an error about uncompiled PL/SQL program units, select **Program** > **Compile** > **All** to compile all PL/SQL program units in the report.

5. In the Save dialog box, navigate to the folder where the style sheet **`pseudo_class.css`** is located. This style sheet is available under the folder

OOW_FOLDER\HTMLCSS\result. In the Save dialog box, type the name of the output file. For example, **pseudo_class.html**.

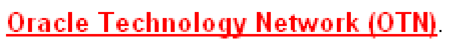
6. Click **Save**. This generates the report output.
7. Open the generated HTML file in your browser. You will see a link as shown in Figure 16:

Figure 16: Hyperlink as seen in the report output


Oracle Technology Network (OTN).

8. Hover your mouse over the hyperlink to observe the effect shown in Figure 17.

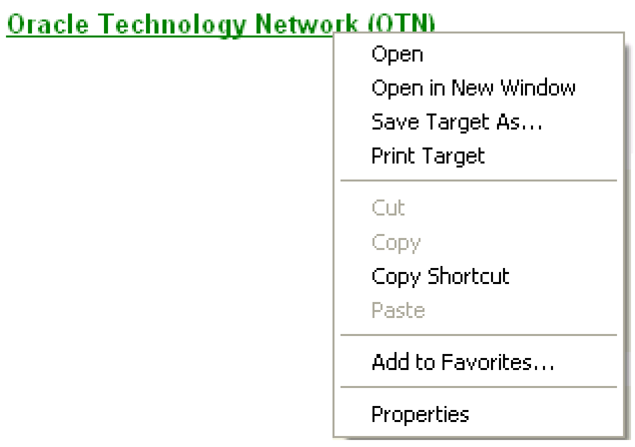
Figure 17: A hyperlink on mouse-hover


Oracle Technology Network (OTN).

Notice that the hyperlink has become bold, increased the font size, and changed its color to red.

9. Either right-click or left-click the hyperlink. Figure 18 shows how the hyperlink will appear:

Figure 18: Hyperlink in active state

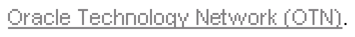

Oracle Technology Network (OTN)

Notice that the hyperlink changes the color to green as soon as you left-click or right-click it. This state is called the *active state* of the hyperlink.

10. If you right-clicked the hyperlink, click **Open**. The hyperlink will open in the same browser window as the report.
11. Click the Back button in your browser to navigate back to your report output. Notice that the hyperlink is still displayed in the active state; that is, in green color.

12. Click the Refresh button in your browser to reload the report output. You will see the hyperlink as shown in Figure 19:

Figure 19: Hyperlink for a location that has been visited


Oracle Technology Network (OTN).

Notice that the hyperlink now changes to a different color. This state is called *visited*, which means that the hyperlink points to a location that is already present in the browser history.

Pseudo-classes Used in This Demonstration

You saw in the above demonstration that the look-and-feel of hyperlinks during various stages of user interaction with the report output is different from that of the hyperlinks that you normally see on the Web. This custom look-and-feel of hyperlinks is achieved using Cascading Style Sheets (CSS) pseudo-classes, and the custom style sheets feature of Oracle Reports 10*g* Release 2 (10.1.2).

The syntax of pseudo-classes is:

```
selector:pseudo-class {property: value}
```

Open the style sheet **`pseudo_classes.css`** in any text or CSS editor. Notice the pseudo-classes, which use the above syntax. For example:

```
a:hover   { color: red; font-size: 125%; font-weight: bold }
```

The above code snippet uses the pseudo-class **`:hover`** with the selector **`a`** (the anchor tag **`<a>`** is used for creating hyperlinks in HTML). The above code defines the look-and-feel of hyperlinks when the user hovers the mouse over the hyperlink. Refer to Figure 14 to see how the hyperlink looks in the output with the above code.

Similarly, other pseudo-classes like **`:unvisited`** have been used in the CSS for defining the custom look-and-feel of hyperlinks.

Open the report **`pseudo-classes.rdf`** in Reports Builder. Open the report Property Inspector. Notice the following property:

```
Property name:  Style Sheets
Property value: pseudo_class.css
```

This applies the CSS **`pseudo_class.css`** to the report output using the custom style sheets feature of Oracle Reports.

Summary

Pseudo-classes and pseudo-elements can be used to apply formatting based on information that lies outside the document tree; for example, the various stages of user interaction with the HTML document. In this example, we used pseudo-classes and the custom style sheets feature of Oracle Reports 10*g* Release 2 (10.1.2) to define a custom look-and-feel for hyperlinks.

You can get more information about pseudo-classes and pseudo-elements here:
`http://www.w3.org/TR/REC-CSS2/selector.html`

Topic 5: Handling Error Conditions in JavaServer Page (JSP) Reports

Using JavaServer Pages (JSP) reports, you can make use of the control that the Java 2, Enterprise Edition (J2EE) platform offers over how to format error information when a Web component throws an unexpected exception. In this demonstration, we will show how to use JSP error pages to catch, present, and report exceptions gracefully.

Few things make an application look less polished and professional than a server's default exception page. Even the most well-designed page may show the stack trace and exception name, which makes your application look broken. Your application may indeed have a bug, but that's no reason for it to look bad, as well.

Using JavaServer Page (JSP) reports, you can make use of the control that the Java 2, Enterprise Edition (J2EE) platform offers over how to format error information when a Web component throws an unexpected exception. This demonstration shows how to use JSP error pages to catch, present, and report exceptions gracefully, instead of presenting the user with a technical, and possibly confusing, exception.

This topic covers the following sections:

- Prerequisites
- Demonstration
- Summary

Prerequisites

In this example we will create a J2EE application containing an Oracle Reports JSP. To create the JSP application, you will need JDeveloper 10*g* Release 3 (10.1.3), which can be downloaded from the Oracle Technology Network (OTN).

To deploy and run the J2EE application, you will need one of the following:

- Oracle Application Server 10*g* Release 2 (10.1.2) Enterprise Edition
- Oracle Application Server Forms and Reports Services 10*g* Release 2 (10.1.2)
- Oracle Business Intelligence 10*g* Release 2 (10.1.2)

You will also need Reports Builder 10*g* Release 2 (10.1.2) to create reports. Reports Builder is part of Oracle Developer Suite.

Also see the section “Accessing the Example Reports”.

Demonstration

This section covers the following sub-sections:

- Make a JSP Report Part of an Existing J2EE Application
- Configure the JSP Report and the JDeveloper Project

- Deploy the J2EE Application on Oracle Application Server
- Run the Application
- Redirect Users to a Meaningful Error Page
- Redeploy the J2EE Application on Oracle Application Server
- Run the Application to See the Meaningful Error Page
- Redirect Users to a Meaningful Error Page on the Basis of HTTP Status Code
- Redeploy the J2EE Application on Oracle Application Server
- Run the Application to See the Meaningful Error Page

Make a JSP Report Part of an Existing J2EE Application

1. Double-click the following executable to start JDeveloper:

   `JDEV_HOME`**`\jdeveloper.exe`**
   where ***`JDEV_HOME`*** is the installation directory of JDeveloper.

2. Select **File** > **Open**. In the Open dialog box, select **`OOW_FOLDER\JSP\source\JSPDemo.jws`** and click **Open**.

3. You should now see an application named **`JSPDemo`** in the JDeveloper Application Navigator.

4. Expand this application, and you will see a Web project named **`ErrorPages`**. Notice the following files and folders under this project:

Table 1: Description of folders and files in the JDeveloper project

Folder / File Name	Contents
`ErrorPages\Web Content`	All JSPs and other sub-folders.
`ErrorPages\Web Content\WEB-INF\web.xml`	Web deployment descriptor.
`ErrorPages\Web Content\WEB-INF\lib`	the JAR files referred in the JSP code (for example, **`reports_tld.jar).`**
`ErrorPages\Web Content\css`	Cascading Style Sheet (CSS) files.
`ErrorPages\Web Content\images`	Images.
`ErrorPages\Resouces\ErrorPages.deploy`	Deployment Profile for this project. A deployment profile names the source files, deployment descriptors, and other auxiliary files that are needed to deploy a J2EE application.

5. We will now create a simple JSP report and make it part of the JDeveloper project. To do this, start Reports Builder.
6. Connect to the HR schema of the database.
7. Create a simple JSP-based Web report showing employees data based on the HR schema. Use the Beige predefined template to create the report.
8. Save this report as `employees.jsp` in the following folder:

   ```
   OOW_FOLDER\JSP\source\ErrorPagesDemo\public_html
   ```

9. Switch to JDeveloper. Click the **Refresh** ( ) icon in the Application Navigator. You should now see `employees.jsp` inside the `Web Content` folder.

Configure the JSP Report and the JDeveloper Project

1. Double-click `employees.jsp` to view its contents in the Editor pane.
2. Click the **Source** tab at the bottom of the Editor pane. This shows you the JSP code.
3. Locate the following line of code:

   ```
   <%@ page language="java" import="java.io.*"
   errorPage="/rwerror.jsp" session="false" %>
   ```

   Change the code to the following:

   ```
   <%@ page language="java" import="java.io.*" session="false" %>
   ```

   The `errorPage` attribute redirects the user to the specified page regardless of the type of error encountered. We have removed this attribute because we will use the deployment descriptor to redirect the user to relevant help pages based on the type of error encountered.

4. To be able to compile the Reports JSPs correctly, we will need to add some JAR files in the project classpath. Right-click the project **ErrorPages**, and select **Project properties**.

5. In the Project Properties dialog box, select **Libraries** in the tree at the left.

6. In the Libraries panel that displays on the right, click **Add Jar / Directory**.

7. In the file chooser dialog box that displays, select the following file:

   ```
   ORACLE_HOME\reports\jlib\rwrun.jar
   ```

   where `ORACLE_HOME` is the location where Oracle Developer Suite 10*g* Release 2 (10.1.2) is installed.

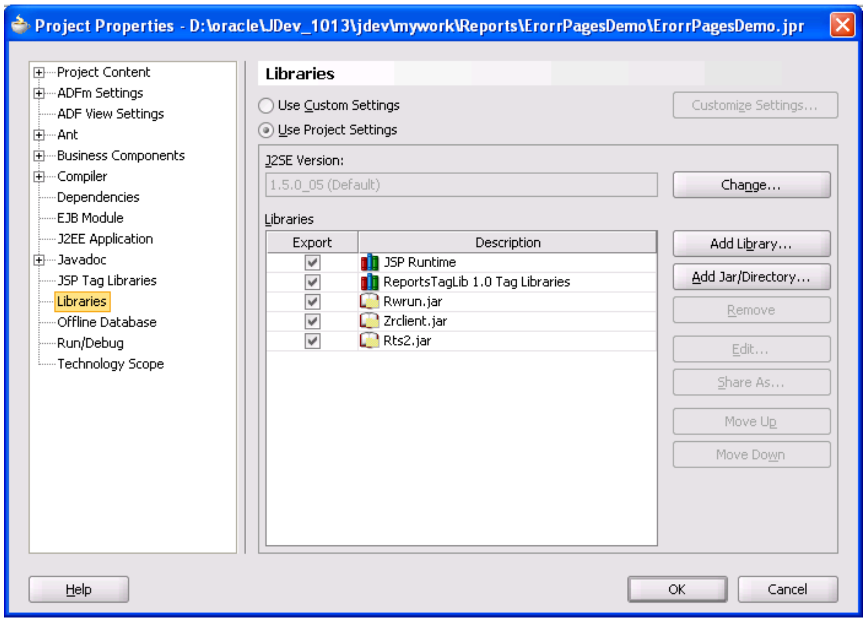
8. Repeat steps 6 and 7 to add the following JAR files to Project Libraries:

   ```
   ORACLE_HOME\jlib\zrclient.jar
   ORACLE_HOME\jlib\rts2.jar
   ```

   where ***`ORACLE_HOME`*** is the location where Oracle Developer Suite 10*g* Release 2 (10.1.2) is installed.

9. Now your Project Properties dialog box should look as shown in Figure 20:

Figure 20: Project Properties dialog box after adding the required JAR files



Project Properties - D:\oracle\JDev_1013\jdev\mywork\Reports\ErorrPagesDemo\ErorrPagesDemo.jpr

Libraries

( ) Use Custom Settings
(●) Use Project Settings

J2SE Version: 1.5.0_05 (Default)

Libraries

Export	Description
✓	JSP Runtime
✓	ReportsTagLib 1.0 Tag Libraries
✓	Rwrun.jar
✓	Zrclient.jar
✓	Rts2.jar

10. Click **OK** to close the Project Properties dialog box.

11. Right-click the **ErrorPages** project, and click **Make** to compile the project. Confirm the following message in the Messages pane at the bottom:

```
[…] Successful compilation: 0 errors, 0 warnings.
```

12. Now we will configure the deployment profile, which will enable us to create the WAR (Web Archive) file.

    Double click `ErrorPages.deploy`.

13. In the WAR Deployment Profile Properties dialog box that displays, note the location of the WAR file. By default, the location is:

```
OOW_FOLDER\JSP\source\ErrorPagesDemo\deploy\ErrorPages.war
```

    We will need this location later when we deploy the WAR file on Oracle Application Server.

14. Under Web Application's Context Root click **Specify J2EE Web Context Root**, and type:

```
ErrorPages
```

15. Click **OK** to close the WAR Deployment Profile Properties dialog box.

16. Now we are ready to create the WAR (Web Archive) file. To do this, right-click **`ErrorPages.deploy`** and select **Deploy to WAR file**.

17. Look for the following message in the Deployment Log at the bottom:

```
----  Deployment finished.  ----    …
```

    The above message means that the WAR file has been created. This WAR file contains our J2EE application with the Reports JSP and other auxiliary files.

Deploy the J2EE Application on Oracle Application Server

1. Now we are ready to deploy the WAR file on the Oracle Application Server. To do this, make sure that the Oracle Application Server instance is running. Open the Oracle Enterprise Manager 10*g* Application Server Control by typing its URL in the browser. For example:

    `http://`*`host`*`:18100`

2. Log in to the Application Server Control by specifying your credentials.

3. Navigate to the following page:

    OC4J_BI_Forms > **Applications**

4. Under Deployed Applications, click **Deploy WAR file**.

5. On the Deploy Web Application page, enter the following values:

    Web Application: *Complete path to the WAR file created earlier*

    Application Name: **`JSP Demo`**

    Map to URL: **`/ErrorPages`**

6. Click **Deploy**.

7. After the deployment is finished, you will see a success message. Click **OK**.

8. On the detail page that displays, you should now see your application (**`JSP Demo`**) listed under Deployed Applications.

9. Click your application name (for example, **`JSP Demo`**).

10. On the Application page, under Properties, click **General**.

11. Under Library Paths, check if **`rwrun.jar`** is included. If not, click **Add Another Row**, then add the missing path(s):

    On Windows:

    ***`ORACLE_HOME`*`\reports\jlib\rwrun.jar`**

On Unix:

`ORACLE_HOME/reports/jlib/rwrun.jar`

where ***ORACLE_HOME*** is the location where Oracle Application Server is deployed.

12. Click **Apply**, then click **OK**.

Run the Application

1. Now we are ready to run our application. Open a new browser window, and type the following URL:

   `http://host:port/ErrorPages/employees.jsp?userid=hr/hr@mydb`

   where ***mydb*** is your database connect string.

   You will see the JSP report output in the browser.

2. Now we will change the URL intentionally to generate an exception. Type the same URL as in the previous step, but deliberately specify a non-existent database connect string.

3. You will see the following error and underlying exception on your browser.

```
500 Internal Server Errorjavax.servlet.jsp.JspException: rwlib-
1: REP-0501: Unable to connect to the specified database.
at oracle.reports.jsp.ReportTag.doStartTag(ReportTag.java:464)
…
…
```

   Notice that by looking at the above exception in the browser, a non-technical user may not be able to guess the steps needed to resolve the problem. Additionally, the exception stack trace leaves a bad impression about your application in the mind of the user.

Redirect Users to a Meaningful Error Page

Our objective in this demonstration is to show the users a page that provides useful tips on how the problem can be resolved, and does not look completely different from the rest of our application. To do this, follow these steps:

1. Switch to JDeveloper, and double click **`Web Content\WEB-INF\web.xml`** to see the contents of this file in the Editor pane.

2. Locate the following code at the end of the file:

```
      …
    </mime-mapping>
  </web-app>
```

3. Change the code to the following:

```
      …
    </mime-mapping>
      <error-page>
        <exception-type>
            javax.servlet.jsp.JspException
        </exception-type>
        <location>myExceptionPage.jsp</location>
      </error-page>
    </web-app>
```

The above code specifies that whenever the Application Server encounters the exception `myExceptionPage.jsp`, the user should be directed to the page `myExceptionPage.jsp`.

4. Select **File** > **Save**.

5. Right-click `Resources\ErrorPages.deploy`, and select **Deploy to WAR file**.

Redeploy the J2EE Application on Oracle Application Server

1. Switch to the Application Server Control. Navigate to **OC4J_BI_Forms** > **Applications**.

2. Select the application **JSP Demo**, and click **Undeploy**. On the confirmation page, click **Yes**.

3. Follow the steps under the section **Deploy the J2EE application on Oracle Application Server** above to redeploy the WAR.

Run the Application to See the Meaningful Error Page

1. Open a new browser window, and type the following URL:

   `http://host:port/ErrorPages/employees.jsp?userid=hr/hr@mydb`

   where ***mydb*** is your database connect string.

2. You will see the JSP report output in the browser.

3. Now we will change the URL to intentionally generate an exception. Type the same URL as in the earlier step, but deliberately specify a non-existent database connect string. This time you should get redirected to the page `myExceptionPage.jsp`. Notice that this page gives you some basic information about the error, and provides the steps you can take to resolve the issue.

   This shows you how you can catch an exception at the Application Server, and redirect the user to a meaningful error page.

You can redirect users to meaningful error pages not only on the basis of exceptions, but also on the basis of HTTP status codes; for example, the commonly encountered **404 Not Found** status code. We will see an example of this capability in the next section.

Redirect Users to a Meaningful Error Page on the Basis of HTTP Status Code

Hypertext Transfer Protocol (HTTP) 1.1 defines certain status codes that the Web Servers can return to the browser depending on certain conditions. In this section we will catch one of the common status codes; that is, `404 Not Found` status code, and show a meaningful error page instead. Follow these steps:

1. Switch to JDeveloper, and double-click `Web Content\WEB-INF\web.xml` to see the contents of this file in the Editor pane.

2. Locate the following code at the end of the file:

```
    …
  </error-page>
</web-app>
```

3. Change the code to the following:

```
  …
  </error-page>
  <error-page>
    <error-code>404</error-code>
    <location>fileNotFound.jsp</location>
  </error-page>
</web-app>
```

   The above code specifies that whenever the Web Server encounters the status code 404, the user should be directed to the page `fileNotFound.jsp`.

4. Select **File** > **Save**.

5. Right-click `Resources\ErrorPages.deploy`, and select **Deploy to WAR file**.

Redeploy the J2EE Application on Oracle Application Server

1. Switch to the Application Server Control. Navigate to **OC4J_BI_Forms** > **Applications**.

2. Select the application **ErrorPages**, and click **Undeploy**. On the confirmation page, click **Yes**.

3. Follow the steps under the section **Deploy the J2EE application on Oracle Application Server** above to redeploy the WAR.

Run the Application to See the Meaningful Error Page

1. Open a new browser window, and type the following URL:

   http://*host*:*port*/ErrorPages/employees.jsp?userid=hr/hr@*mydb*

   where ***mydb*** is your database connect string.

   You will see the JSP report output in the browser.

2. Now we will change the URL intentionally to generate an exception. Type the same URL as in the previous step, but deliberately specify a non-existent JSP name. For example:

   http://*host*:*port*/ErrorPages/employees1.jsp?userid=hr/hr@*mydb*

3. This time you should get redirected to the page **fileNotFound.jsp**. Notice that this page gives you some basic information about the error, and provides you the steps you can take to resolve the issue.

   This shows you how you can catch an HTTP status code, and redirect the user to a meaningful page.

Summary

In this demonstration you learned how to accomplish the following:

- Catch an exception at the Application Server, and redirect the user to a meaningful error page.
- Catch an HTTP status code, and redirect the user to a meaningful error page.

The J2EE platform provides several ways of handling error conditions, of which we have explored two in this demonstration. Refer to J2EE documentation and tutorials on **http://java.sun.com** to explore the other ways. For example:

http://java.sun.com/developer/EJTechTips/2003/tt0114.html

Summary

In this paper, we have seen some tips and tricks that Oracle Reports developers can use to create polished and user-friendly reports. For more information, refer to the other examples, white papers, demonstrations, and how-to documents available on the Oracle Technology Network (OTN):
`http://www.oracle.com/technology/products/reports/index.html`

Oracle Reports: Tips and Techniques
October 2006
Author: Navneet Singh
Contributing Authors: Philipp Weckerle, Ingrid Snedecor

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com