An Oracle White Paper

June 2009

New Features in Oracle Forms Server 11*g*

ORACLE

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Introduction

This white paper outlines the new fatures in Oracle Forms version 11.1.1. Use it to gain an understanding of what new features there are and how to use them.

- **External Events**. This new feature is, in essence, support for the feature called Advanced Queuing (or AQ) available in all editions of the Oracle Database since 8i. AQ is a very powerful and robust asynchronous event solution. This feature brings the possibility to communicate with a Forms module from outside of Forms.

- **JavaScript**. With the help of this new feature Forms can invoke JavaScript code available in the page from which the Forms applet resides The reverse is also possible, that is you can call into Forms from JavaScript.

- **Proxy User Support**. This new feature makes it possible to use Proxy Users in Oracle Forms. Proxy Users are users with very few and limited privileges used in deployments with high security requirements and/or very many users.

- **New Enterprise Manager User interface and functionality**. Oracle Forms' support for EM has been improved with a new user interface and new features. It's now possible to correlate a specific Forms session's activities with activities seen in the database. A function that associates a Forms instance with a SSO instance has also been added.

- **Events in Pluggable Java Components**. Oracle Forms support for Pluggable Java Components (or PJC) has been augmented by adding support for dispatching events to the Forms server in PJC code.

- **Tracing improvements**. The tracing present in Oracle Forms can now log the names of called PL/SQL functions and procedures and the names, types and values of parameters used in those calls.

- **Oracle Diagnostic Logging**. Support for Oracle's standardized logging architecture is at the heart of this new feature.

- **Using the JVM Controller when integrating with Oracle Reports.** The JVM controller functionality used only for Java integration in earlier versions has now been extended to Forms' integration with Oracle Reports.

These new features will help in development, management and modernization of Oracle Forms applications and allow them to integrate with different technologies through a serviced based achitecture (SOA).

External Events

Oracle Forms now integrates the Advanced Queuing (AQ) functionality of Oracle 8i and later for outside events to trigger internal Forms code. In earlier versions of Forms, this was only possible through custom programming, usually done in Java with the help of Forms' Java Bean support. This new functionality makes it possible to call into Forms from any technology that can interface with AQ, for example Java Messaging (JMS) or BPEL.

Background

This functionality introduces a new node in the object navigator that represents an event in the database. Forms developers only need to know one thing to declare their interests in an event: the name of the queue. Armed with this knowledge, the developer can register an interest in the event that causes Forms to subscribe to it at runtime.

A new trigger called WHEN-EVENT-RAISED fires when Forms is notified about an event to which it is subscribed, and the developer can specify in that trigger, what should happen in the application when the event is raised.

Oracle Forms can also publish events to Advanced Queuing as well through the database package DBMS_AQ. It is possible to post events raised by other Forms applications, or 3rd party applications.

Client Refresh

Since Forms uses the HTTP protocol, which is exclusively a request and response protocol, situation can arise when an event has been raised by AQ; the Forms server knows about it, but it cannot execute the WHEN-EVENT-RAISED because there are no pending requests from the client.

To make it possible to execute the trigger and propagate the consequences of that trigger to the client when the client is idle, a new application property called MAX_EVENT_WAIT has been introduced. Developers can use MAX_EVENT_WAIT to specify, in milliseconds, how long the maximum delay is between when an event is raised and when its effects are propagated.

Example

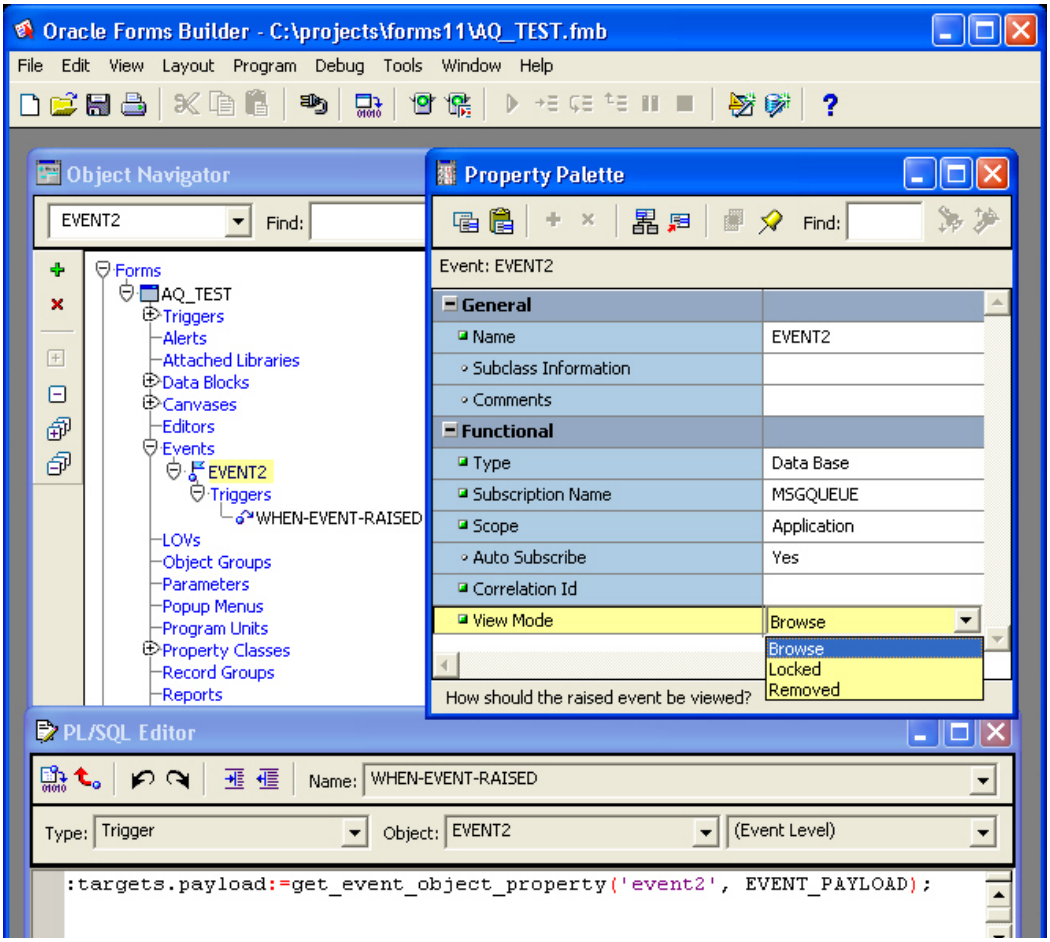

Figure 1: Use Forms Developer to declare a subscription to an event.

In this example, an event has been declared as 'EVENT2'. The queue it is subscribing to is called 'MSGQUEUE' (see the property Subscription name in the property palette). The scope in this example is set to Application so that it will be monitored even if the form where it is defined is not the current form. An alternative is to have it monitored only if this form is active.

The event is set to subscribe automatically to the queue upon start-up (see the Auto Subscribe property). When an event occurs, Forms will only browse for it and not remove it from the queue (see the View Mode property). It is also possible to remove it after reacting to it, as well as to lock it from other subscribers but leave it in the queue.

The correlation ID is not used in this example but it can be set to an arbitrary string value to specify what category of event to monitor. Each Advanced Queuing event can (but isn't required to) have a correlation ID that is set when the event is published to the queue.

When an event occurs, the called WHEN-EVENT-RAISED associated with the event fires. In this example, the field that is named 'payload' is assigned the value of the event's payload variable as found by issuing the get_event_object_property function call. The arguments to *Get_Event_Object_Property* are the name of the property and a constant designating what property to get, in this case the payload that we assign to a field called 'payload' in the block called 'targets'.

The constants are:

CONSTANT NAME	DESCRIPTION
EVENT_SUBSCRIPTION_NAME	Gets the event name as declared in the builder.
EVENT_TYPE	Gets the event type. Only 'Database' is currently supported.
EVENT_PAYLOAD	Gets the payload as supplied by the queued event.
EVENT_ENABLED	Gets the enabled status of the event.
EVENT_CORRELATION_ID	Gets the event correlation ID, if any. This is also declared in the builder.
EVENT_SCOPE	Gets the scope of the event: Application or Form
EVENT_VIEW_MODE	Gets the View mode: Browse, Removed or Locked.

It is also possible to set the properties of an event with set_event_object_property.

Summary

Oracle Forms can now subscribe and react to external events that are published to the database's Advanced Queuing (AQ) feature.

Since many other technologies can publish events to AQ (examples include JMS and BPEL), Forms can now interact with those technologies in an asynchronous manner. Your Oracle Forms applications can react to events outside Forms, as well as interact with other Forms applications.

Integration with JavaScript in the surrounding Webpage

This new functionality makes it possible to execute JavaScript in the page in which the Forms applet is embedded.

Oracle Forms allows you to make two new calls in the built-in package called web:

`Web.Javascript_Eval_Expr` and

`Web.Javascript_Eval_Function`

The first call, Web.Javascript_Eval_Expr, is a procedure which takes two arguments: an expression and a target, both of data type varchar2. The expression is a legal JavaScript expression that is interpreted in the web page that hosts the Forms applet. The expression can be a call to a function defined in the page or to some JavaScript that is not in the page. The expression is executed, using the native JavaScript function eval, in the context of the page or frame that is named in the target argument. If the target argument is null, it is executed in the surrounding page or frame.

The second call, Web.Javascript_Eval_Function, is a function and returns a varchar2 value. This call can be used to create a JavaScript function on-the-fly by passing in text that is legal JavaScript in the context in which the Forms applet executes.

Practical applications

In Oracle Forms 11*g*, this JavaScript functionality allows you to integrate Forms with HTML-based application technologies in the web browser. So, an existing Forms application could embed a newer Web 2.0 application within the same browser window and share data and function calls.

Summary

It's now possible, through Forms code, to interact with JavaScript code in the page in which the Forms applet is embedded. You do that with the help of two new additions to the built-in package named **`web`**.

Calling from the surrounding Webpage into Forms

You can also let JavaScript call into Oracle Forms by using JavaScript in the web page that hosts the Forms applet.

There is now functionality available on the embedded Forms object in the DOM (Document Object Model) tree. You use JavaScript to call the raiseEvent method thus:

`document.forms_applet.raiseEvent(event_name, payload);` or

```
document.forms_applet.raiseEvent(event_name);
```
or

```
document.forms_applet.raiseEvent;
```

The assumption here is that you have set the ID configuration variable to 'forms_applet'.

When the surrounding web page executes this JavaScript code, Oracle Forms fires a new type of trigger called WHEN-CUSTOM-JAVASCRIPT-EVENT. In this trigger there are only two valid system variables: :system.javascript_event_value and :system.javascript_ event_name. These variables contain the payload and event name that were passed into Forms through the raiseEvent method.

Oracle Forms can react to a call from an external JavaScript call by writing something like this in the WHEN-CUSTOM-JAVASCRIPT-EVENT trigger:

```
declare
  event_val varchar2(300):= :system.javascript_event_value;
begin
  if (:system.javascript_event_name='show') then
    handleShowEvent(event_val);
  elsif(:system.javascript_event_name='grab') then
    handleGrabEvent(event_val);
  else
    null;
  end if;
end;
```

This PL/SQL code only knows of 2 different events: **'show'** and **'grab'**. Any other event name is ignored.

Practical applications

You can synchronize an HTML based application, whether it is Java-based or otherwise, with a Forms-based application in the same hosting web page. Perhaps you can use the HTML-based application to query data and use Forms to update it if, and only if, the user has the correct access privileges.

Summary

Oracle Forms 11*g* now allows events in the same web page as your Forms application to trigger events in the Forms runtime instance. You do this through a method that has been exposed on the Forms Java applet called raiseEvent.

Making use of the Proxy User functionality

Oracle Database supports proxy user authentication, which allows a client user to connect to the database through an application server as a proxy user. The client user first authenticates with the application server, while the application server authenticates itself as the proxy user with Oracle Database. The client user's credentials are maintained all the way to the database on any proxy connection opened this way. Oracle Forms 11*g* supports this method of authenticating users.

Problem

What are the problems that proxy user authentication is trying to solve? For one, many large applications, including Oracle's own E-business Suite, use a pool of connections to optimize resources. Connection pooling requires a single user. Having a single user connect optimizes resources but creates a problem with auditing; all inserts, updates and removals of records appear, from the database's perspective, to have been done by a single user. To restore auditing, the application development team must write customized auditing code in the database that requires that a user name is passed to the database from the application. This step not only takes development time but also duplicates functionality that is already implemented in the Oracle Database.

The second issue is that of security. If that single user access is ever compromised, the compromised user will have access to the entire application schema.

Solution

The solution to both of these problems is the deployment of proxy users. Although Forms does not permit connection pooling, the security and auditing advantages are still available. The single, high privilege, user in the above example, is replaced by a proxy user that has only one privilege namely "create session".

Application users are maintained in SSO/OID and connect through the proxy user, thereby maintaining the audit trail through all tiers. The application users are assigned roles that grant them insert, update and delete privileges, upon connecting. Since these roles contain no "create session" privileges, application users cannot connect to the database from outside of the application context using tools such as SQL*Plus.

Example

The application user's password will never be presented to the database, only his user name and the proxy user's user name and password. Forms will, with the help of OCI calls, issue the equivalent of:

```
SQL> connect midtier[scott]/midtierPW@databaseTnsName
```

In this example, issuing the select statement "**`select user from dual`**" in the database will return **`'scott'`**-- not **`'midtier'`**.

This example essentially tells the database to trust that the user is authenticated elsewhere and to let the user connect without a password and be granted the connect role.

Changes in Forms Built-ins

The built-in *Get_Application_Property* now takes a new parameter called **`IS_PROXY_CONNECTION`** (a Boolean). Supplied with this parameter the call returns true if the form is running in proxy user mode, false otherwise.

The built-in *Logon* is changed so as to make it possible to logon programmatically in proxy user mode.

Reports Integration when you use Proxy Users

The integration with Reports will be maintained if a proxy user is used in Forms. Reports will have to be set up to use a proxy user but that is the extent of the change needed to support proxy user with the Reports integration.

Summary

Oracle Forms 11*g* leverages the proxy user functionality in Oracle Database

To set up proxy users, you must

- Use Forms in SSO mode with users defined in OID, and a RAD section that defines what proxy user to connect as
- Have a proxy user defined in the database
- Alter your users in the database to allow connecting through the proxy user

If you use the proxy user functionality

- Automatic database auditing will work as if you used real database users.
- You will increase the security of your application by separating the accounts which creates sessions from the accounts that can modify the database. Each can, on their own, do no harm to the database without the other
- You can take advantage of new built-ins to programmatically ascertain if the form is running in proxy user mode and/or login to the database as a proxy user
- Forms integration with Reports is maintained if Reports is set up to use proxy user as well.

Forms and Grid Control in version 11*g*

Oracle Forms' support for Enterprise Manager and Grid Control has been largely rewritten for version 11*g*. What follows is a preview of what the user interface looks like and of the new functionality that has been added.

Home page

Forms' home page in Grid Control is substantially revamped. The following screenshot show the new home page (with the Forms specific menu shown):

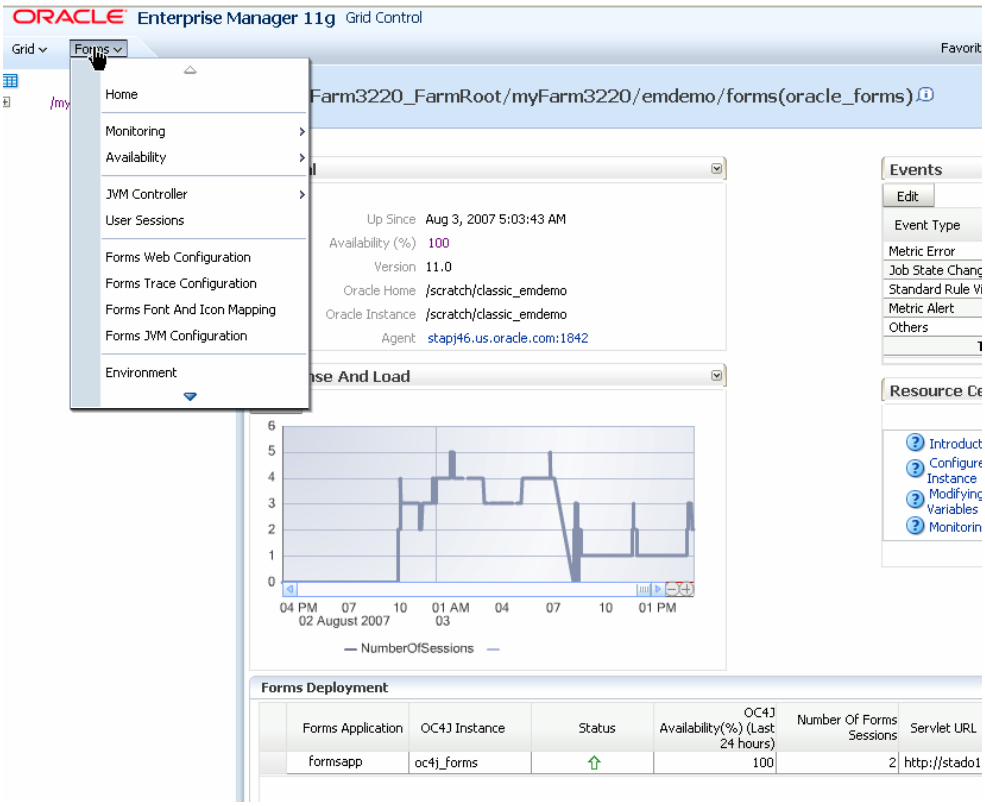

Figure 2: Forms home page with menu

To the left is the overview tree control. From here you have an overview of the entire farm that is controlled by this instance. To the right is the main Forms page with general information and the response and load graph to the left and events and the resource center to the right. Under that is the list of Forms instances in this node. Here you can monitor the overall status and what configuration and environment files are in effect.

The menu shown in this screenshot replaces the horizontal tabs in earlier versions of Enterprise Manager/Grid Control

User Sessions

The user sessions page provides all the functionality from the last version in a fresh new way:

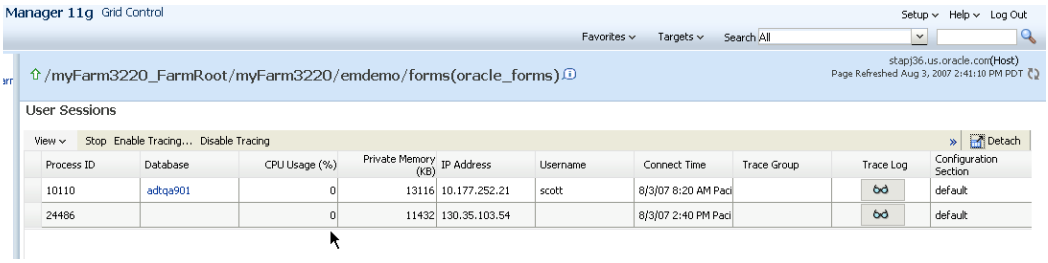

Figure 3: User sessions page

In this page you can ascertain the process ID, CPU and memory usage for each process, IP address and username, and what configuration section is used for each session. It's also possible to turn on and off tracing from here. Stopping a session, for example if it has a runaway CPU usage pattern, is also done from here.

In the column called database is another new feature. It's possible to drill down from each user session into the database session used by that Forms session. Clicking on the link in that column brings you to a page that shows information about the host the database session runs on, the database's name, the name of the form that initiated this session and other relevant information. The database sessions used by the Forms session is listed next, together with database username, session ID and memory usage. The SQL statement that was last issued by the form is shown in the SQL Statement section along with the execution plan.

Configuration page

Now let's focus on the configuration page.

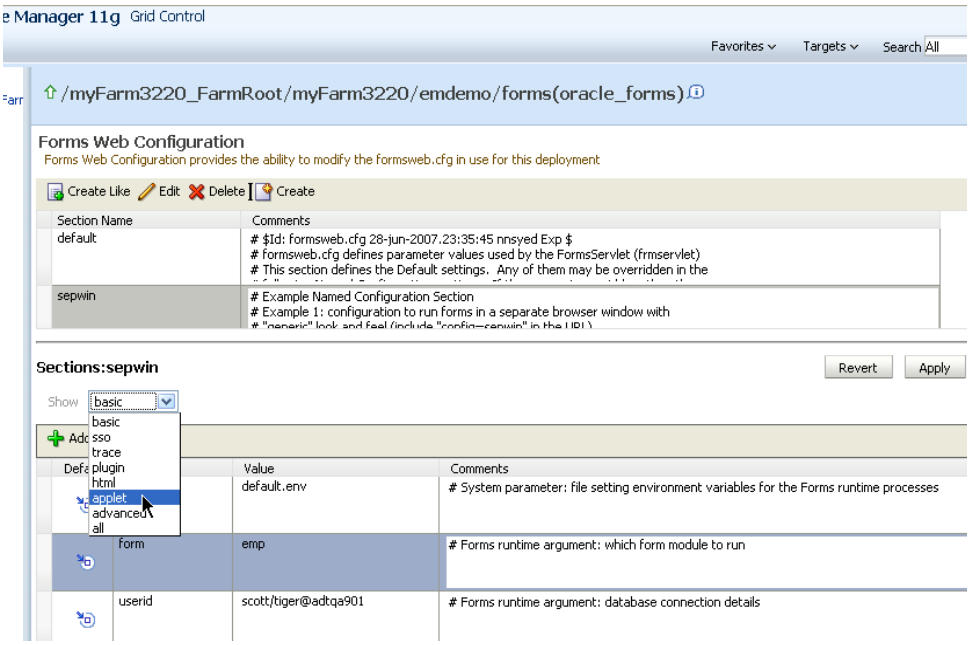

Figure 4: Configuration page

This page contains two areas. The top area shows the different configuration sections that are defined in the formsweb.cfg file. The bottom area contains the variable defined in the section selected in the top area. In this screenshot the "sepwin" section is selected at the top.

The lower area shows off a new feature. It contains variables not just from the selected section but variables that are inherited from the default section. Notice how the variable shown has an icon with a blue incoming arrow. That means that the variable is inherited from the default section. In 10.1.2 it was often unclear what variables were in effect for a certain section because you didn't see the inherited variables in each section.

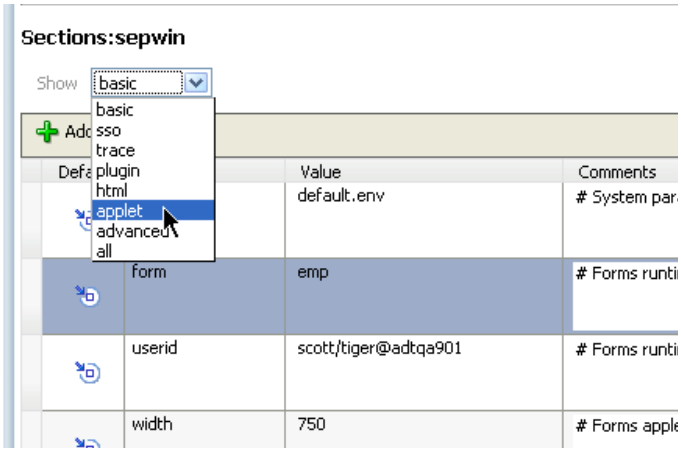

Figure 5: Variable groups

The drop down menu in the upper left corner of the lower area makes it possible to show only certain predefined groups of variables or all variables.

To focus exclusively on the variables that are unique to the selected section you check the button labeled "Hide inherited".

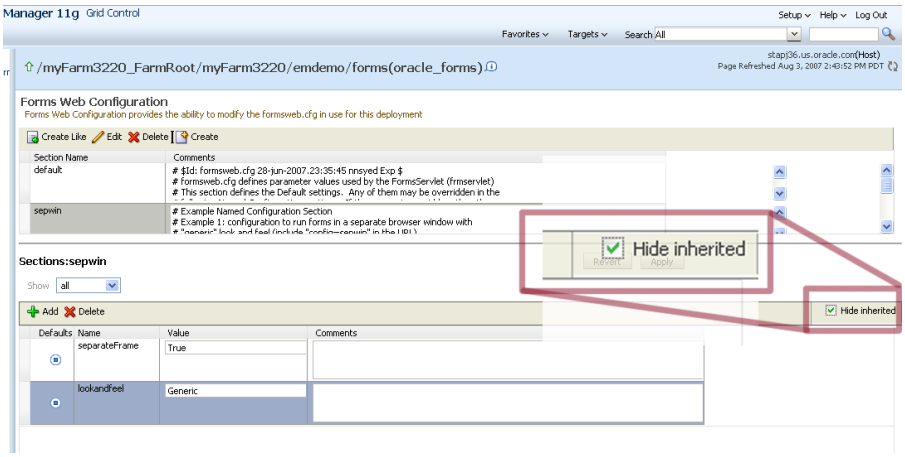

Figure 6: Hide inherited variables

Environment

The Environment section makes it possible to change, add to and remove the environment variables governing the instance.

Associating an Single Sign-on (SSO) instance

An administrator can associate a Forms instance with an OID instance using Grid Control. It's also possible to de-associate the Forms instance from within Grid Control. This makes what was previously a somewhat complex task a simple thing to accomplish.

Monitoring

The monitoring section is next:

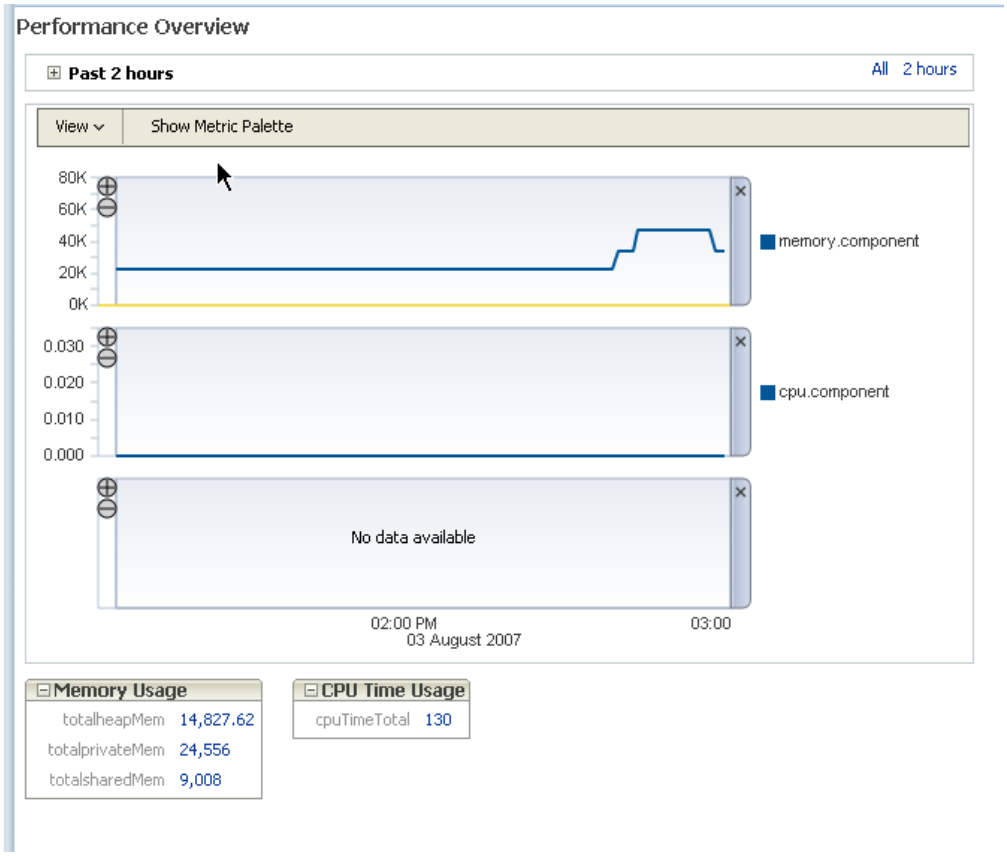

Figure 7: Monitoring

The default view in this section is shown here. Memory and CPU over time are the two graphs shown in this screenshot even though the labels say different. The metric palette will make it possible to configure the page. You can choose among the available metrics and add and remove them from this page.

Summary

Oracle Forms' support for remote monitoring and administration in Grid Control has received a significant face lift making user interaction a lot more intuitive. It also has a few enhancements to the functionality available in earlier versions.

Enhanced Java Support

Oracle Forms provides Java classes that define the appearance and behavior of standard user interface components such as buttons, text areas, radio groups, list items, and so on. A Forms pluggable Java component (PJC) can be thought of as an extension of the default Forms client component. When you create a PJC, you write your own Java code to extend the functionality of any of the provided default classes.

Dispatching Events from Forms Developer

In earlier releases of Oracle Forms, Forms user interface components implemented a version of the IView interface that did not have any special method to add or remove CustomListener from the pluggable Java component or the view. In Oracle Forms 11*g*, you can add or remove CustomListener in the IView interface thus making it possible to raise events on the Forms Server. Your Java code that runs on the client can effectively interact with your PL/SQL code on the server.

Enhancements to Forms Trace

The tracing functionality in Forms is a comprehensive and performant feature with very many discrete instrumentation points. With this functionality it is possible to find out what the application is doing at a very granular level. In versions prior to 11*g* it is however not possible to have the names of called program units (procedures or functions in the form or in the database) or the names, types and values of parameters used when calling program units show up in the trace files. You could see that a program unit had been called but not find any information about it.

New instrumentation points

In version 11*g* of Oracle Forms Services new instrumentation points have been added for the Forms trace. The following table lists the new points and a description of each point.

NEW TRACE POINTS

EVENT NUMBER	DESCRIPTION
65	Client-side Program Unit start/end
66	Trigger start/end

96	Built-in start/end
100	Database PL/QL Started
194	Built-in Arguments
196	Program Unit start/end

Support for Oracle Diagnostic Logging

Version 11*g* of the Oracle Application Server introduces a common logging function called Oracle Diagnostic logging (or ODL). This new functionality seeks to harmonize the different logging standards that have evolved among Oracle's products. While these standards have served each product well, the many differing standards are making managing several of Oracle products a much more challenging task than is strictly necessary. As an administrator you needed to learn and use several different methods, log file locations and formats and had to develop methods of rotating log files and to restrict their sizes.

Benefits of Oracle Diagnostic Logging

Oracle Diagnostic Logging is based on the standard Java logging framework in J2SE (java.util.logging). That brings, besides the obvious benefits of being a Java standard, automated size restriction handling and log file rotation. Thus it is possible to set a maximum size of a log file and have the framework enforce that limit automatically. It is also possible to specify that a log file that has reached its maximum size be swapped out for a new one under defined circumstances, making it possible to back up log files with ease. Additionally the log files can be managed in Enterprise Manager, centralizing the log file handling

Message correlation

The Diagnostic Logging is managed from Enterprise Manager and because Database logging can be managed from there as well, it is possible to correlate events seen in Forms Diagnostic log files with related events in the database, thus facilitating application specific problem discovery and resolution.

Performance metrics

The log messages can be specified to include performance related information such as stop and start times, wait and block timings, duration of single request/response cycles and network transport timings and volume.

Integration with Reports

In 10*g*, the Forms Runtime Process created a separate JVM before calling Oracle Reports and Reports used this JVM to execute the report. The JVM was part of the Forms Runtime Process. In 10*g*, the JVM pooling feature was used only by the Java Importer. In version 11*g* of Oracle Forms Services, Forms can use the shared JVM controller for Oracle Reports requests as well, substantially reducing memory consumption.

When using JVM pooling, a process called JVM controller is available which hosts the JVM. Several Forms Runtime Processes can share a single JVM.

Seting up JVM pooling for Reports Integration

There is no need to set up JVM pooling specially for use with Oracle Reports. If JVM pooling is already set up for Java integration it will automatically be set up for integration with Oracle Reports.

To set up JVM pooling, see the chapter called "Configuring and Managing Java Virtual Machines" in the Oracle Fusion Middleware Forms Services Deployment Guide.

Conclusion

With the new features in Oracle Forms developers are further aided to be able to evolve and modernize their Forms applications through technolgies such as Advanced Queueing, JavaScript, Java and Enterprise Manager/Grid Control.

New Features in Oracle Forms 11*g*
June 2009
Author: Jan Carlin
Contributing Authors:

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

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

0109