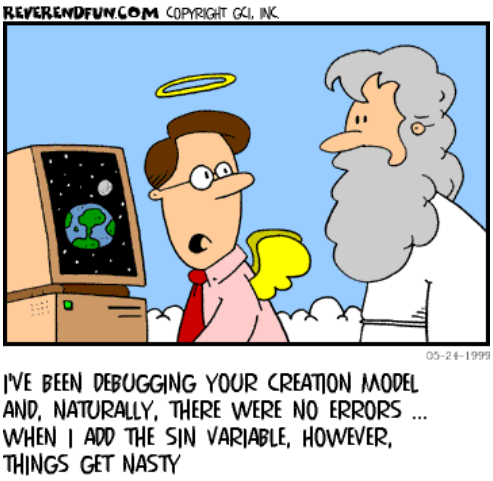
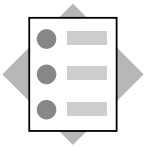
Web Dynpro Debugging

Contents:

- **Understand Debugging functionality.**
- **Activate Debugging**
- **Start a Debug Session**
- **Analyze the State of a Running Program**
- **Apply Debugging Techniques**
- **Terminate a Debug Session**



Web Dynpro Debugging: Lesson Objectives



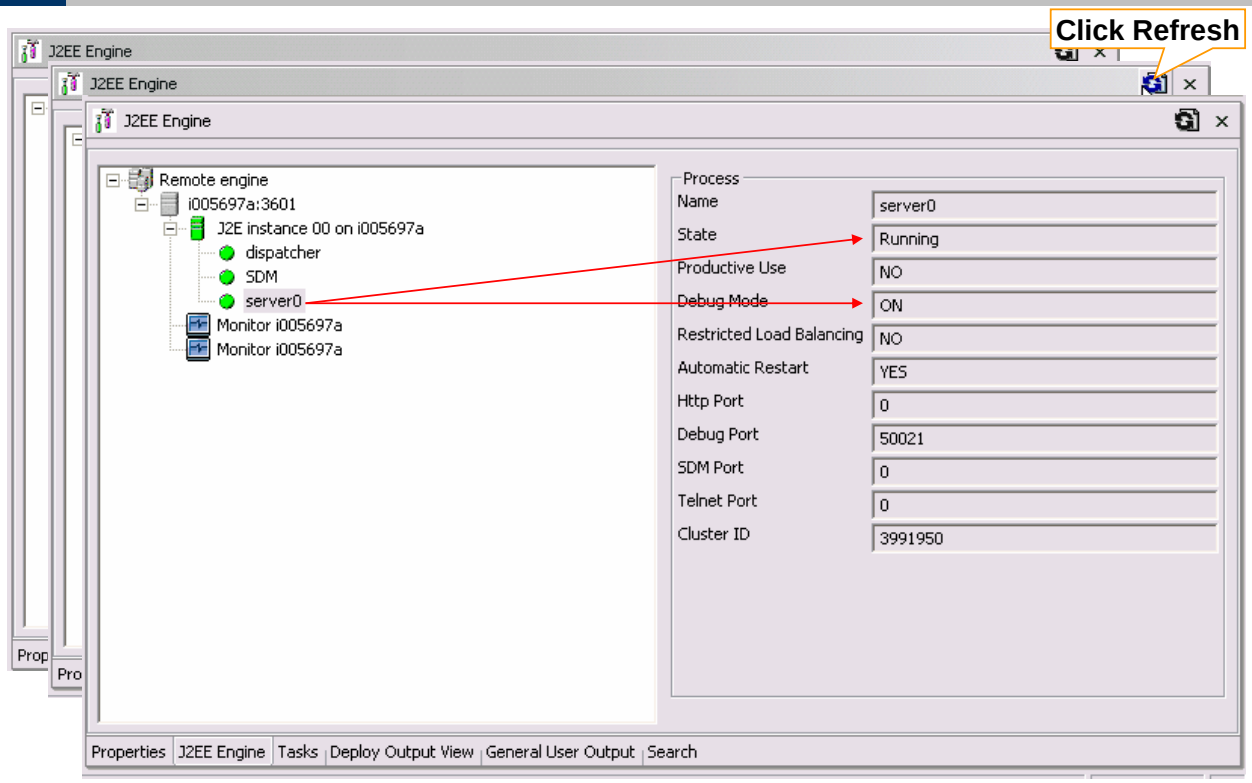
After completing this lesson, you will be able to:

- **Understand Debugging functionality.**
- **Activate Debugging**
- **Start a Debug Session**
- **Analyze the State of a Running Program**
- **Apply Debugging Techniques**
- **Terminate a Debug Session**

Web Dynpro Dynamic Programming: Business Scenario

- An important part of your programming toolkit should be mastery of debugger. It will help you save time and frustration in locating and eliminating software bugs.

Switching Server Node to Debug Mode



■ Switching Server Nodes to Debug Mode

To be able to debug within a running Web Dynpro application, you must activate debugging for the server process of the J2EE Engine. You activate this in the *J2EE Engine* view.

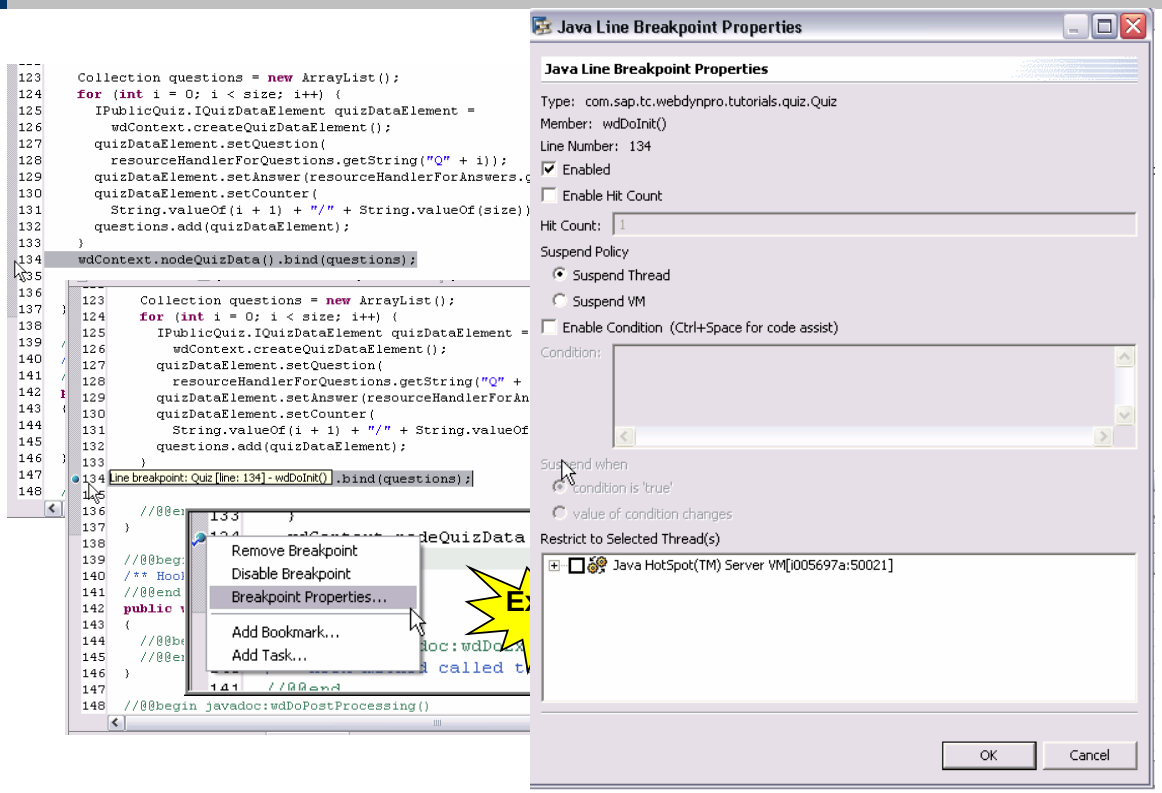
1. If necessary, open the *J2EE Engine* view. To do so, choose *Window → Show View → Other and then select J2EE → J2EE Engine*. Choose *OK* to confirm your entries. The system displays a view containing status information about the running J2EE Engine.

2. Expand the tree display fully until you can see the actual server process (for example *server0*).
3. Right-click the server node and then choose *Enable debugging of process* from the context menu.

Result:
The server process is stopped and restarted in debugging mode. Only the ON value is shown for Debug Mode. To display the current status of the server, in the view toolbar, choose *Refresh*. Wait until the server has the status *Running*.

Note: • You can only debug in non-productive server nodes (*Productive Use* has the value *NO*)

Setting a Breakpoint



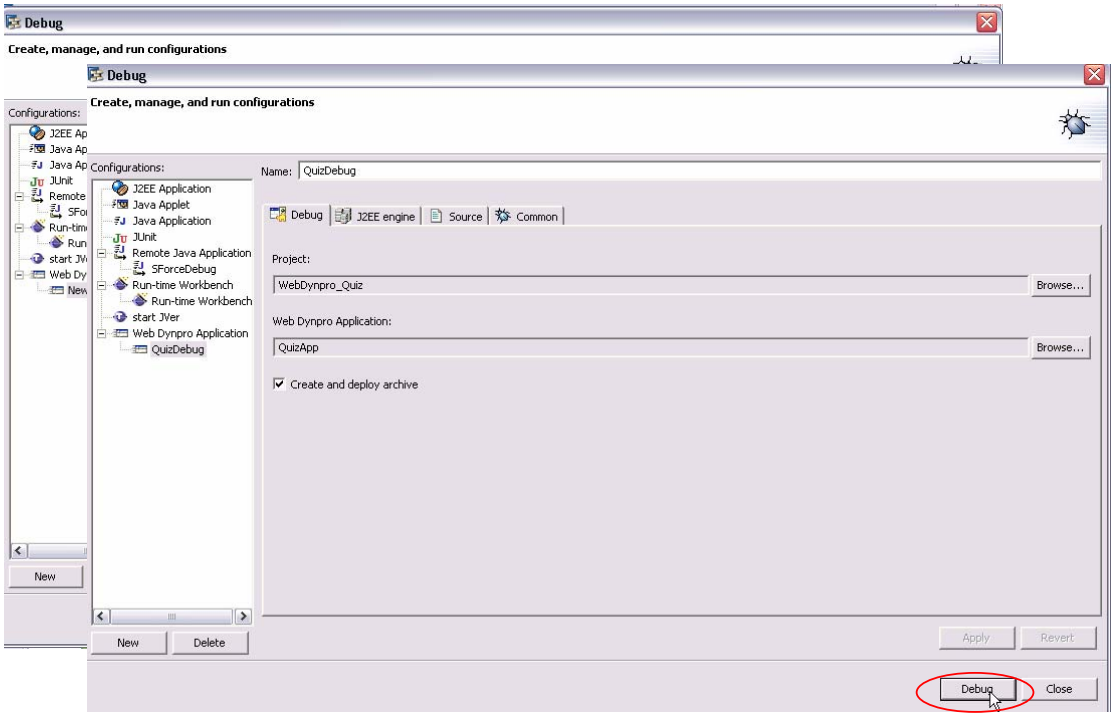
■ **Setting a Breakpoint**...

1. Open the implementation page of the Quiz Component from the QuizApp application. To do so, in the Web Dynpro Explorer, edit the Quiz Component Controller and go to the Implementation tab page.

2. The Editor display the source code. Navigate to wdDoInit() method. Right-click on the marker bar (along the left edge of the editor area) frame next to line of code to open the context menu and choose Add Breakpoint. You can also doubleclick in the markerbar to achieve the same results. The breakpoint lines are highlighted with a blue dot.

Extra: You can also experiment with setting conditional breakpoints by setting the breakpoint properties from the context menu

Defining a Debug Configuration and Starting the Debug Mode



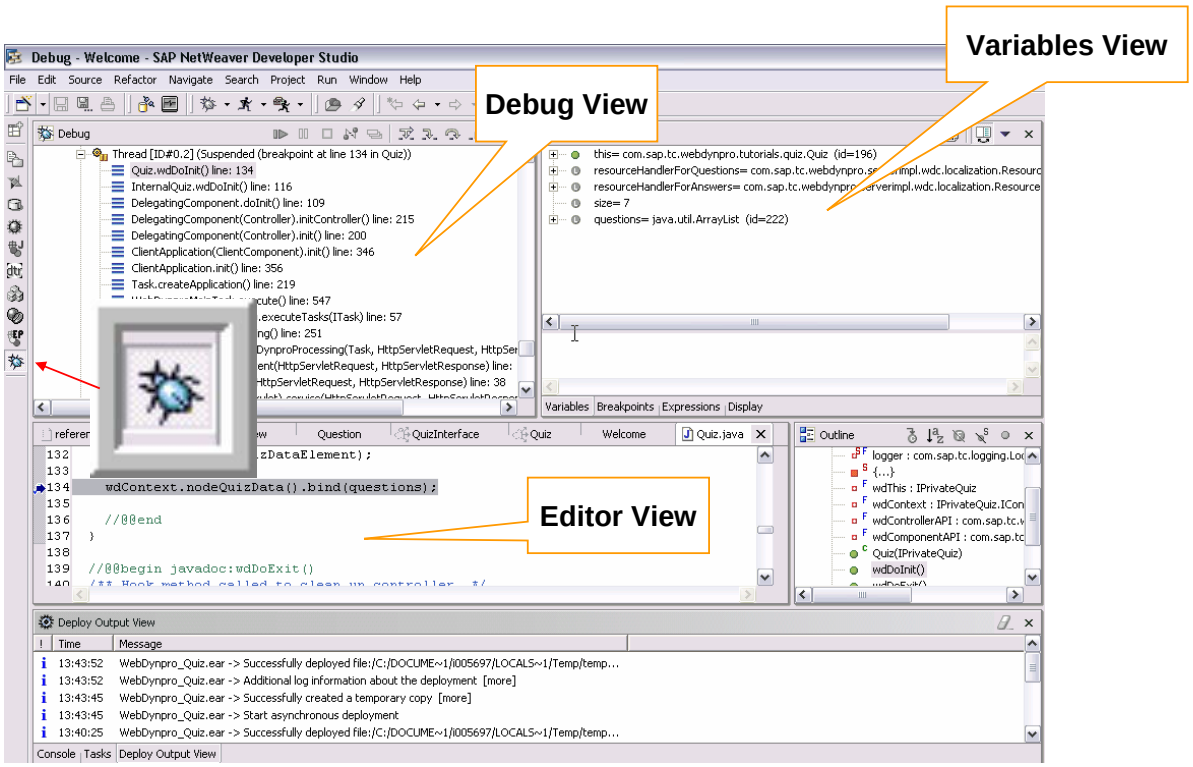
■ Defining a Debug Configuration and Starting the Debug Mode

To start the Web Dynpro application in the debugger, you require a *launch configuration*.

1. Choose *Run → Debug...* in the main menu.
2. In the list of possible configurations, select *Web Dynpro Application* and then choose *New*.
3. Under *Name*, enter QuizDebug as the name of the configuration.
4. Choose *Browse...* next to the *Project field. Next, select the WebDynpro_Quiz* project and confirm with *OK*.
5. Choose *Browse...* next to the *Web Dynpro Application field. Next, select the QuizApp* and confirm with *OK*.
6. If the *QuizApp* application to be debugged has not yet been deployed on the server, select the *Create and deploy archive* checkbox.
7. [Optional] To select the server that you want to use for the debugging procedure, choose the *J2EE engine* tab page.
8. The configuration is now complete and you can start the debugger.
9. To start the debugger, choose *Debug*.

The SAP NetWeaver Studio automatically switches to the debug perspective. The Web application is started in an external Browser. If the application appears that it can no longer be executed, you should switch back to the SAP NetWeaver Developer Studio, you will see that the application was stopped at the breakpoint and can now be analyzed.

Debug Perspective

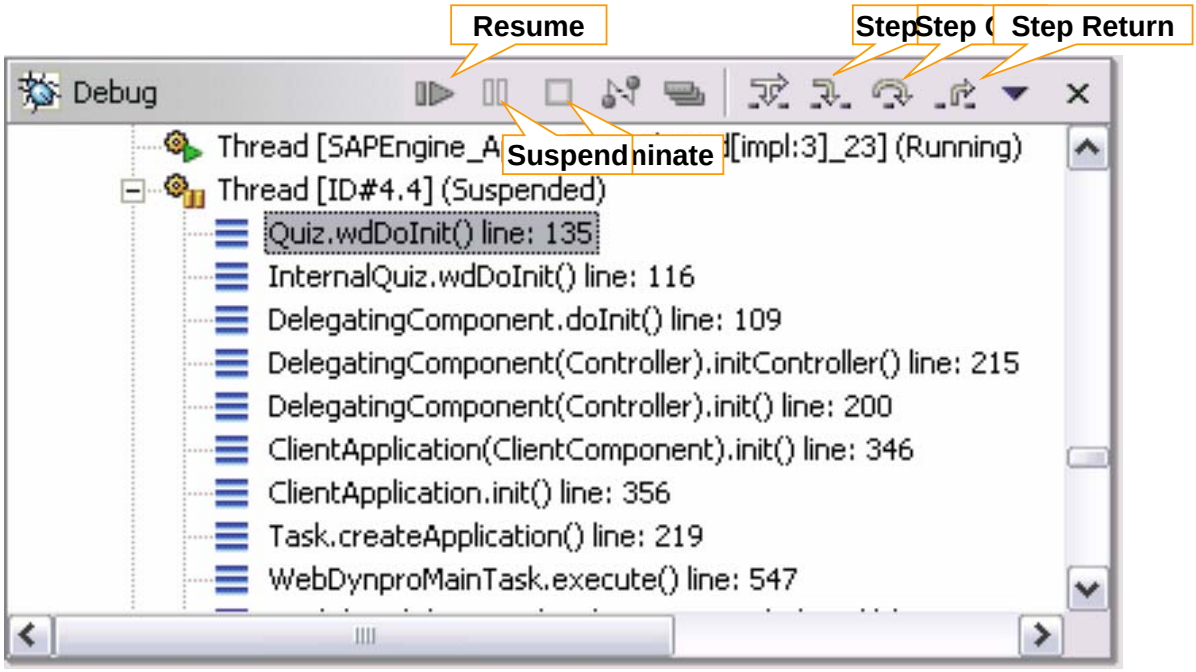


The Debugging perspective

Once you have started the debugging process, Eclipse will switch to the debugging perspective.

The **Debug view** is at the upper left of the perspective. This view allows you to manage the debugging or running of a program in the workbench. It displays the stack frame for the suspended threads for each target you are debugging. Each thread in your program appears as a node in the tree. It displays the process for each target you are running.
The panel at the upper right contains a number of tabbed views including Variables (shown), Breakpoints, Expressions, and Display. This view shows information about variables that are currently in scope (currently-selected stack frame) for your program.
The **Editor view** - The Java editor provides you with Java specific text editing support. You will see the program line the debugger is currently executing. If the execution of your program takes you to a different class, Eclipse will open up the corresponding .java file automatically.
Console view - shows the output of a process and allows you to provide keyboard input to a process, but this only works if you started the java process in eclipse. Debugging the server means that you attach to the process remotely, therefore you will need to rely on the log files to see the process output.
Tipp: If you rearrange the view you can restore the default settings by selecting Window->Reset Perspective. You can obtain more information about any of the views by clicking on the title of the view and pressing F1.

Debug View: Step Execution



- **Debug View: Step Execution**

  There are various different options for further processing on a step-by-step basis:
  F5: ***Step Into*** Jumps to the next statement. If you are at a method call and you would like to see what the method does you should use *Step Into*

  F6: ***Step Over*** The next command is executed without jumping to the current statement. Use *Step Over* if you only need to know what a method will return or how it will change your variables.

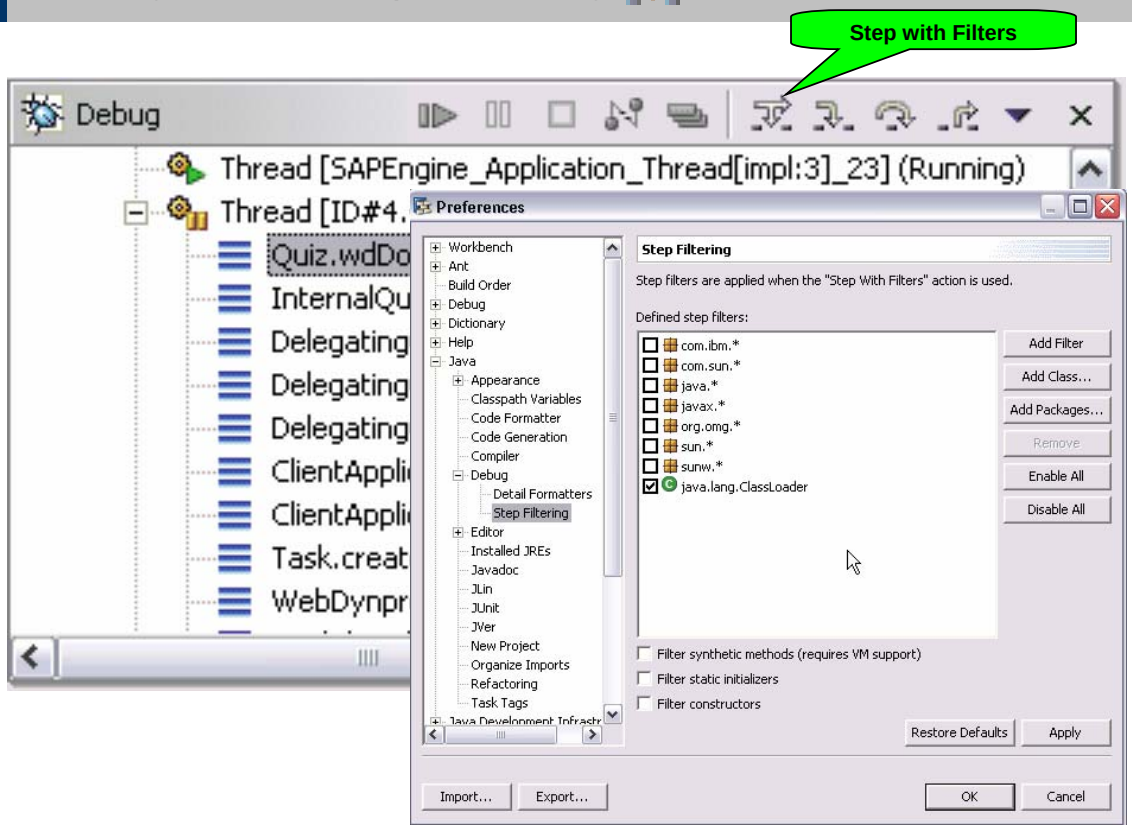
  F7: ***Step Return*** will finish the method you are currently in and return to the calling method. If you previously chose F5, you can choose F7 to cancel the debugging of the current command

  F8: ***Resume*** The application exits debug mode and continues with execution

  These buttons allow you to navigate “step thru” the program execution while allowing you to observe what values methods return and how this effects your in scope variables.

  Suspend - will pause execution and allow you to view state of variables
  Terminate - will terminate execution of the program

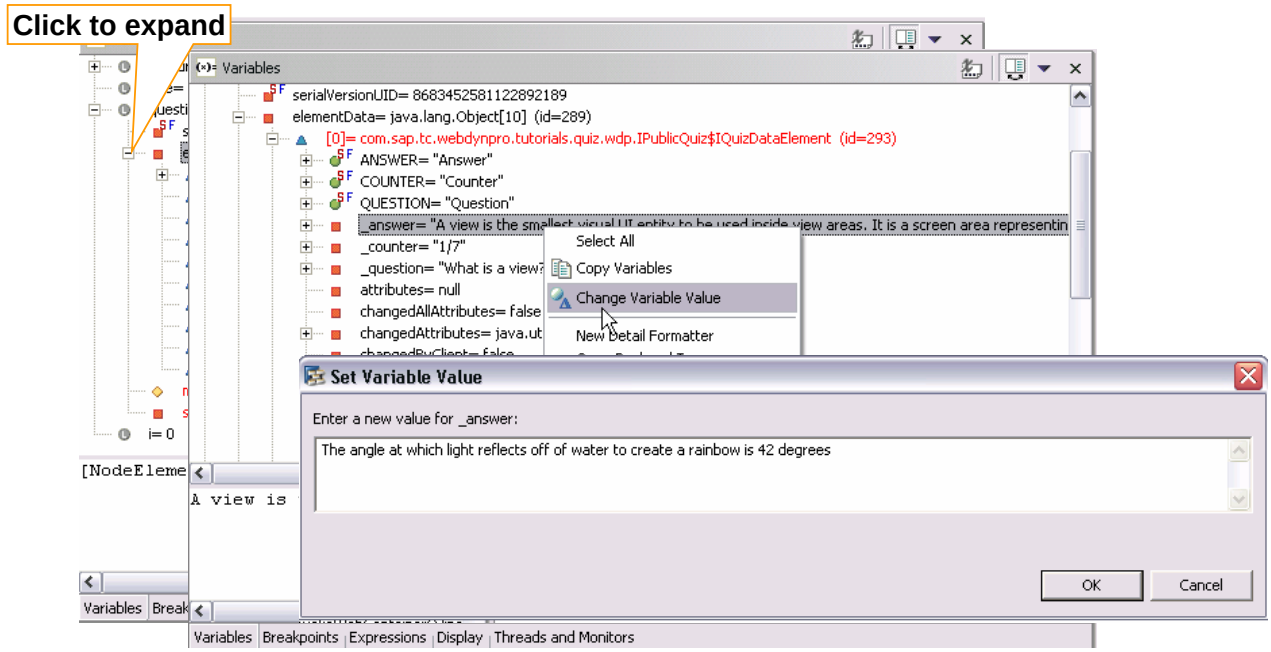
Debug View: Step Filtering



Debug View: Step Filtering

The Step With Filters button works like Step Into, but you can specify which methods Step Filter will execute and return from immediately by selecting (Window > Preferences > Java > Debug > Step Filtering). and defining *step filters* by checking the packages and classes listed there. This way you can step only into methods in your own classes and not into the standard Java packages or third-party packages.

Variables view



Variables View

Inspecting values - When stack frame is selected, you can see the visible variables in that stack frame in the Variables view. The Variables view shows the value of primitive types. Complex variables can be examined by expanding them to show their members.

Additional Functionality
Add/Remove watchpoints
Change variable value
Selected variable(s) can be inspected

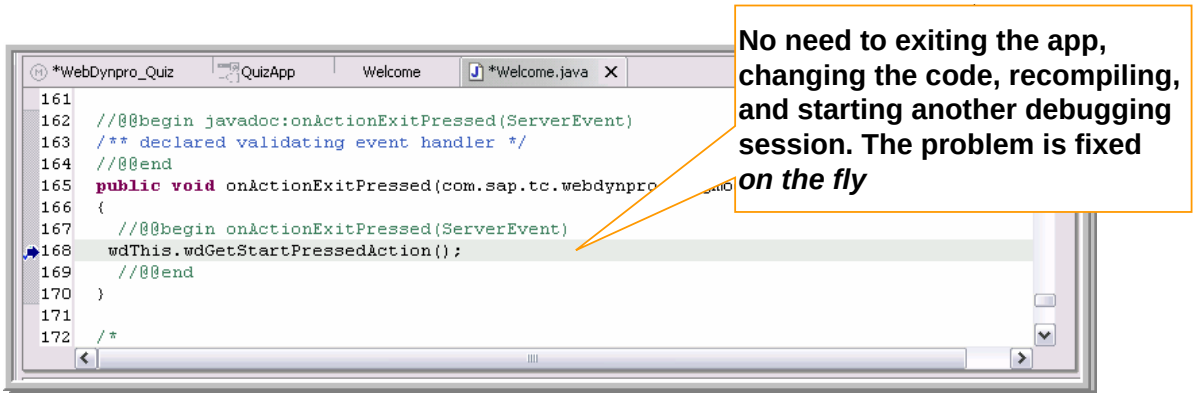
Hot Swap

Allows you to change code “on the fly” during your debug session

Steps

1) Change the code in the editor while your debugging

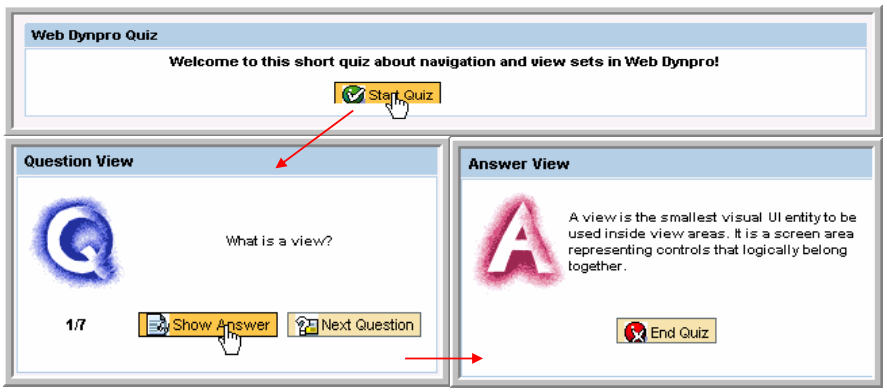
2) Save the file, then press the Resume button to get the Web Dynpro Application running again with your changes



■ **Hot Swap**

A new feature of JVM 1.4 (Java Virtual Machine) and Eclipse version of NetWeaver Developer Studio is Hotswap Bug Fixing. JVM 1.4 is compatible with the Java Platform Debugger Architecture (JPDA). The JPDA allows you to replace modified code in a running application. To use this debugging function, simply change the code in the editor, save and resume debugging. This is a much easier approach than exiting the app, changing the source code, recompiling, and then starting another debugging session. With HotSwap, You can fix the problem *on the fly* while the debugger is still running.

Overview of Web Dynpro QuizApp



Application: QuizApp

Component: Quiz

Views: **Welcome**

Question

QuestionMark

Answer

Resource Files

```
Q0 = What is a view?
Q1 = What is a view set?
Q2 = What is a view area?
Q3 = What is a window?
Q4 = What is a view composition?
Q5 = How do you embed views into other views?
Q6 = What is a component interface view?
SIZE = 7
```

```
##################################################################
##
##  Answer resources for Web Dynpro Tutorial WebDynpro_Quiz
##
##################################################################
A0 = A view is the smallest visual UI entity to be used inside view areas. \
     It is a screen area representing controls that logically belong \
     together.
```


Overview of Web Dynpro QuizApp

QuizApp is the sample application we will use to show how to view and change the component context using the debugger. It will also be used for the hands-on exercise. By seeing the application, it should make the next slides clearer.

This example application can be downloaded from SDN - Creating an Extended Web Dynpro Application (2)

In this quiz application you will familiarize yourself with some main concepts of Web Dynpro-based application development. The example covers the following aspects of the Web Dynpro programming model:

Creating a Web Dynpro application with multiple views and a more complex navigation structure

Working with actions and event handlers

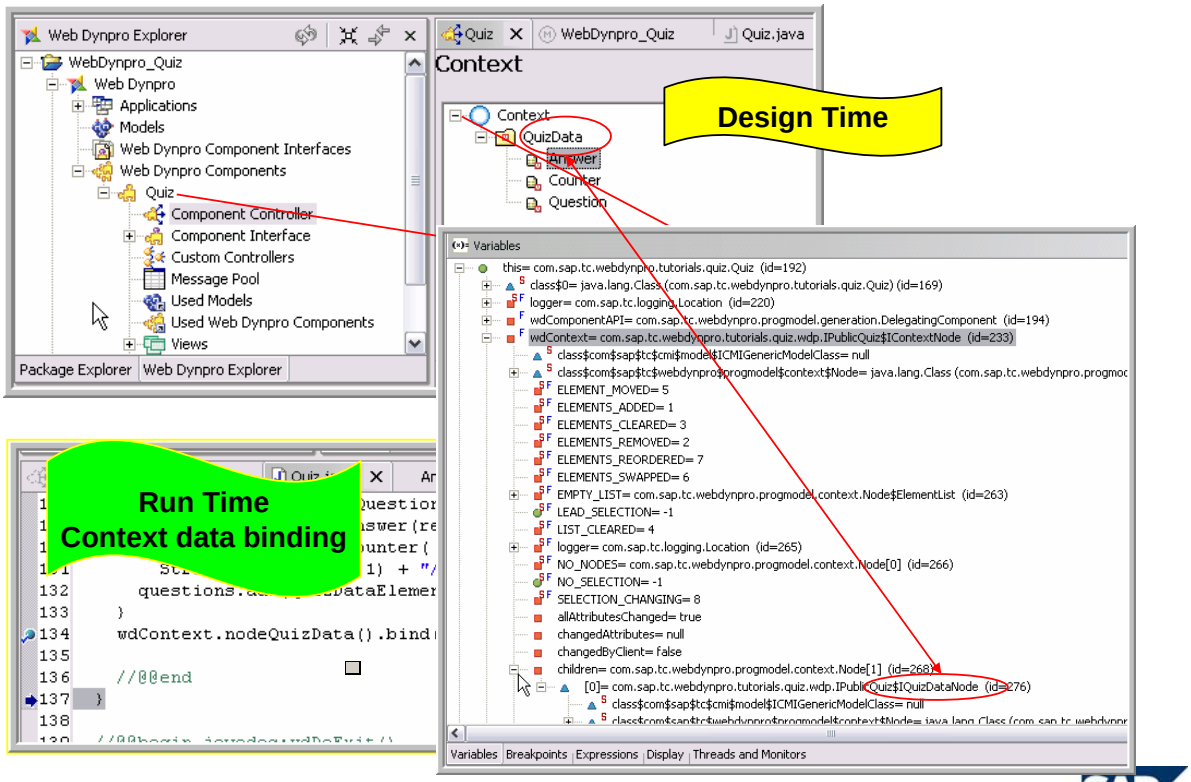
Reading contents from the property resource bundles and storing them in a Web Dynpro context

Dividing data between view controllers and component controllers using context mapping

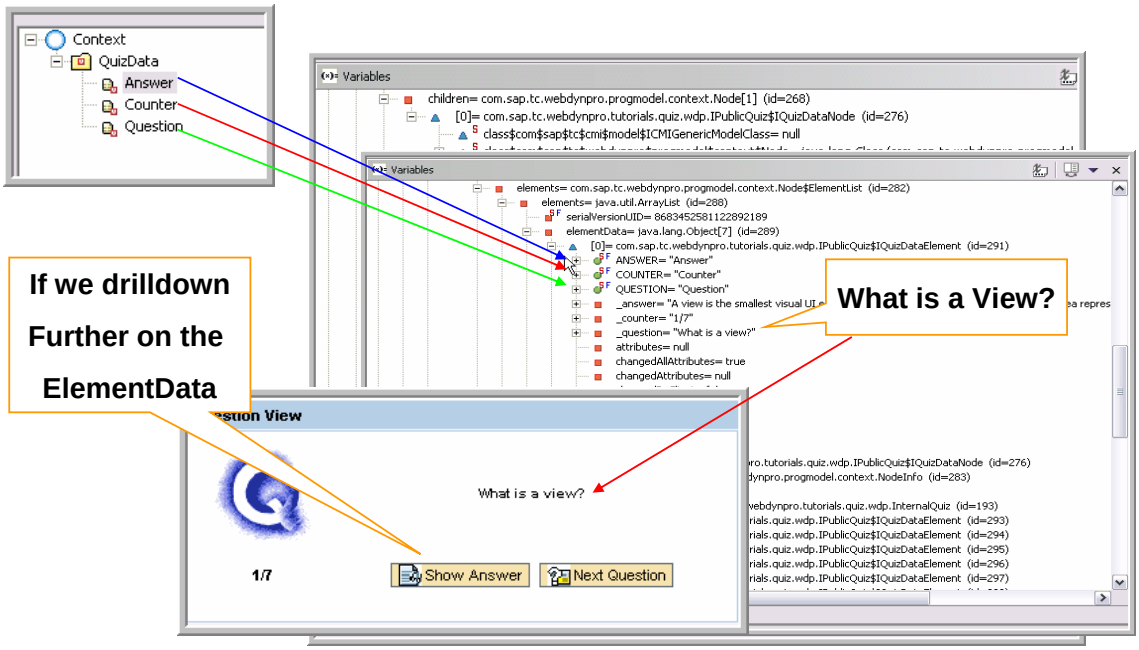
Manipulating UI element attributes at runtime using data binding

Leaving a Web Dynpro Application using an exit plug

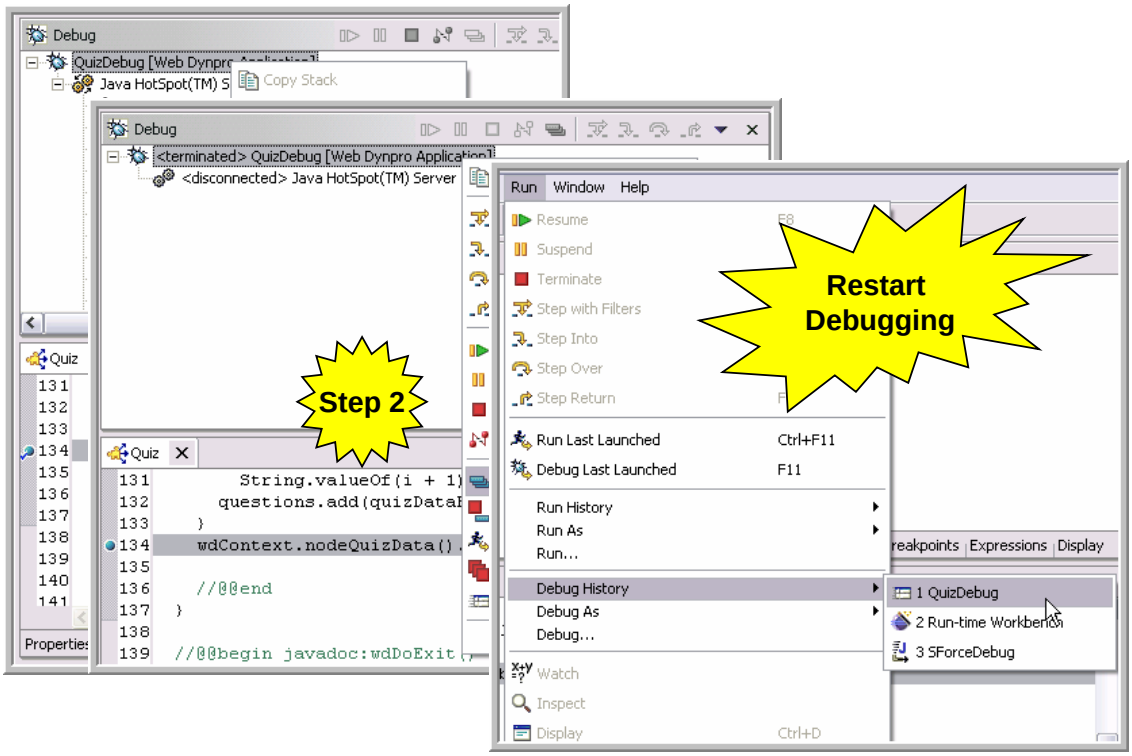
Web Dynpro Component Context



Web Dynpro Component Context



Terminating a Debug Session



Terminating Debugging

If you want to exit debugging, you must terminate the threads in the SAP NetWeaver Developer Studio.

Proceed as follows:...

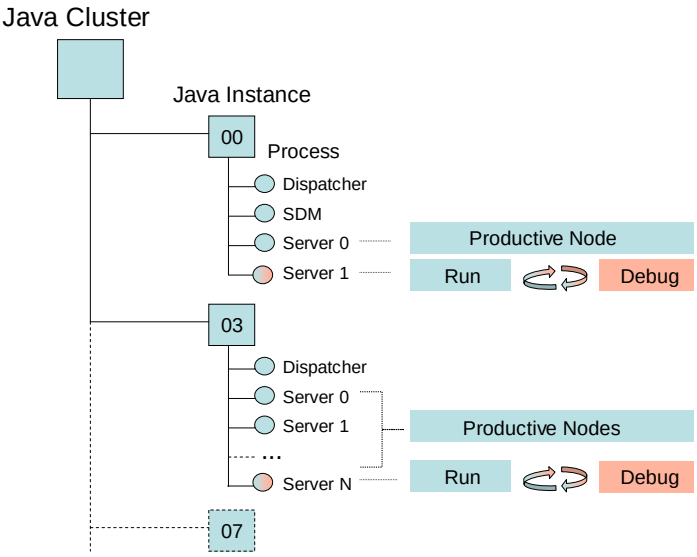
1. In the *Debug View*, call the context menu for the top node *(QuizDebug[Web Dynpro Application]).* Choose *Terminate.*

2. Call the context menu again and then choose *Remove All Terminated Launches*. The Remove All Terminated Launches tool button clears the Debug view of threads that have been Terminated.

3. You can restart debugging of the *QuizApp* application by choosing Run ->*Debug History* and select *QuizDebug*.

Debugging Scenarios

- **Standalone**
- **LAN Scenario**
  - Single Server
  - Cluster
- **WAN Scenario**



■ **Debugging Scenarios**

(Standalone) developer workplace installation -The Web AS is installed on the same local developer PC as the Developer Studio. By default, the debugging mode is deactivated for this single server configuration. However, the (only) server node (server0) can be switched from run mode to debugging mode. In this way, the connection to the server is completely reserved for the debugging session. The server is then stopped.

LAN scenario - Web AS is installed at any location in the LAN. The simple developer workplace installation can also be in effect in the LAN scenario. However, the Developer Studio and the server (including the database) are installed on different hosts. In general, a LAN will feature a cluster configuration. It defines a group consisting of several instance nodes. In such a cluster configuration, one or more server nodes can be reserved for debugging. When debugging is activated the message server deactivates the relevant server node for the cluster communication. As a result, incoming requests are no longer passed on to the reserved node. A debugging session can be executed from within the Developer Studio via the debug port without affecting the other server nodes in the cluster.

WAN scenario - The Web AS runs on the customer side. Debugging takes place through firewalls using SAP Router technology.

Debugging ABAP Code from within Web Dynpro

Prerequisites

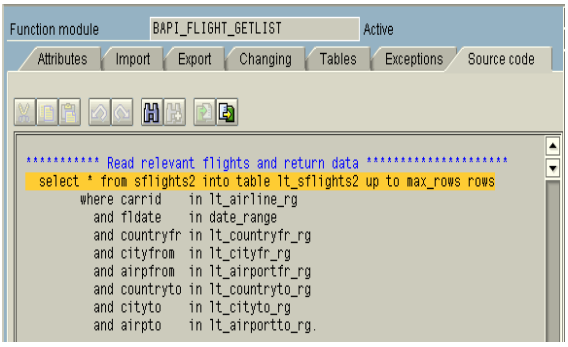
Adaptive RFC

SAP GUI

Debugging Authorizations

Set the ABAP breakpoint in

Dedicated ABAP application server (no load balancing)



Steps

1. Open SAP GUI
2. Activate external debugging in ABAP
3. Set HTTP (external) breakpoint in the ABAP
4. Start J2EE engine – deploy your app
5. Start your test application on the J2EE

Results

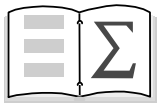
The ABAP Debugger will stop at the ABAP breakpoint

■ **Debugging ABAP Code from within Web Dynpro**

You can debug ABAP code from Web Dynpro application that use Adaptive RFC Prerequisites:
SAP GUI is installed on your local machine.
ABAP system is connected to via a dedicated server (no load balancing). You then only have to activate the relevant setting for external debugging in the ABAP Workbench.
Set the breakpoint in the ABAP code before starting the Web Dynpro.

Since debugging in the ABAP system via load balancing is not supported, you must reconfigure your JCO destinations so that the calls take place on a dedicated ABAP application server. For Web Dynpro, this is done in the Web Dynpro Content Administrator. In the destination maintenance, specify *Single Server Connection* as the *Destination Type*.
Step-by-Step Procedure
Use the SAP GUI to log on to the system or server to which the RFC call is to take place.
You should log on with the same user with which the call will take place.
Activate external debugging in the ABAP Workbench.
Set an HTTP (external) breakpoint in the ABAP code.
Start the J2EE Engine and – if you have not already done so – deploy your application. Start your test application on the J2EE Engine. Results:
The J2EE Engine establishes a connection to the SAP GUI and starts the ABAP Debugger in a new GUI session. The Debugger is stopped at the line in the ABAP code at which the breakpoint is set.
Hint: A detailed document - Remote Debugging in the Developer Studio is available in SDN.

Web Dynpro Debugging Lesson Summary



You should now be able to:

- **Understand Debugging functionality**
- **Activate Debugging**
- **Start a Debug Session**
- **Analyze the State of a Running Program**
- **Apply Debugging Techniques**
- **Terminate a Debug Session**

■ **Web Dynpro Debugging Unit Summary**

No bugs have been harmed during the making of this presentation.