



**IBM Rational Software Conference 2009**  
As Real as It Gets!



# IBM Rational Software Conference 2009

## **RAD Extensibility for Development based Analytics**

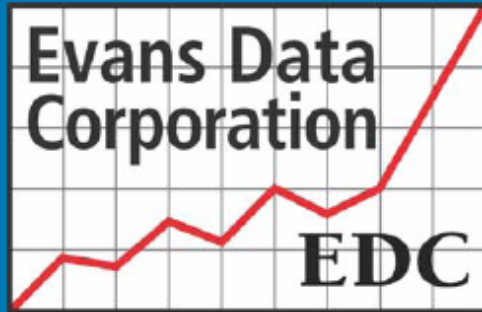
**Rajesh Kalyanaraman,**  
Staff Software Engineer, RAD  
IBM Software Labs, Bangalore

**S. Srilakshmi,**  
Architect – Technology  
Java Center of Excellence – GTO, Cognizant

**Rational** software



**Cognizant**  
Passion for building stronger businesses



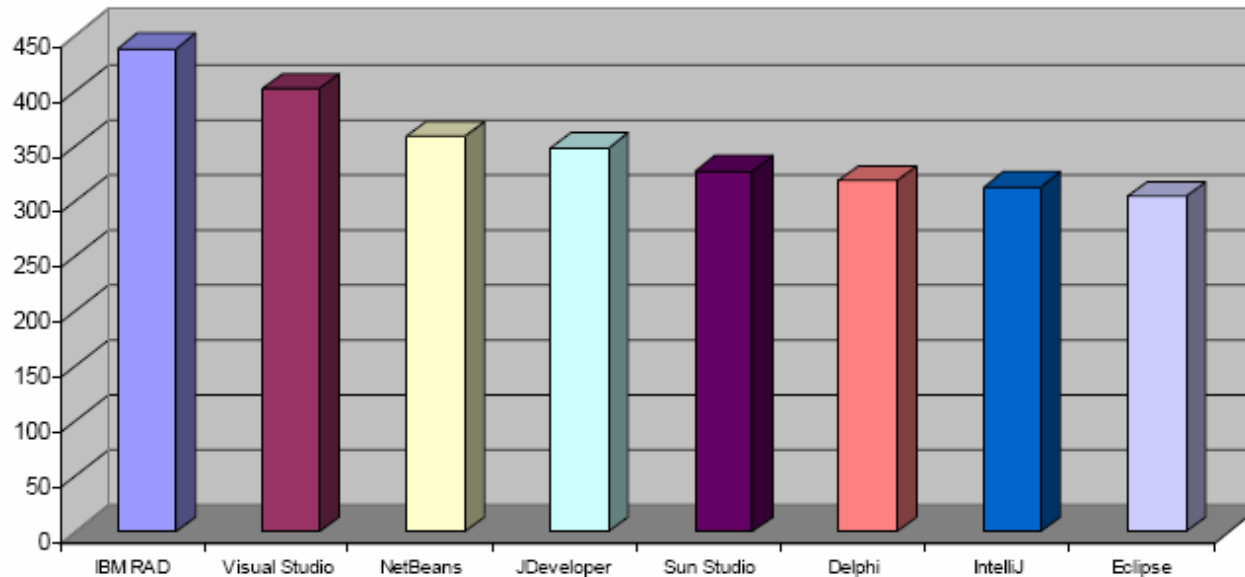
*Users' Choice:  
2009 Software Development Platforms*

A comprehensive user satisfaction survey of over 1200 software developers

June 2009

Evans Data Corp, 740 Front St, Santa Cruz, CA 95060  
www.evansdata.com (800) 831 3080

**Overall Software Development Platform Ranking**



# Agenda

- **RAD Extensibility**
  - Project Metrics API using JDT & AST
  - Custom Plug-in Development
  - Reporting Infrastructures – BIRT & Crystal Reports
  - Building Custom JSF Web Components
  - Building Visual Custom Tags
- **JACP System Overview**
- **JACP Demo**
- **Q & A**



# Project Metrics from JDT

- **Java Development Tooling**
  - JDT Core - the headless infrastructure for compiling and manipulating Java code.
  - JDT UI - the user interface extensions that provide the IDE.
  - JDT Debug - program launching and debug support specific to the Java programming language.
- **You can**
  - Programmatically manipulate Java resources, such as creating projects, generating Java source code, performing builds, or detecting problems in code.
  - Programmatically launch a Java program from the platform
  - Provide a new type of VM launcher to support a new family of Java runtimes
  - Add new functions and extensions to the Java IDE itself

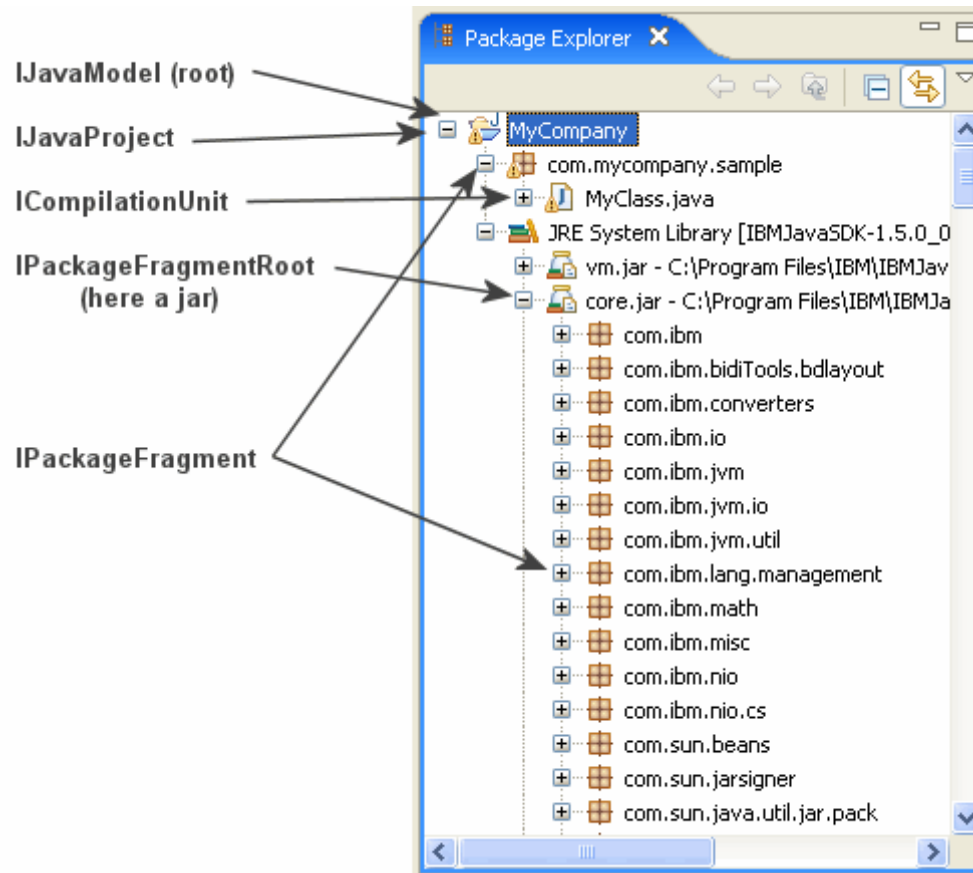


# JDT Core APIs

- [org.eclipse.jdt.core](http://org.eclipse.jdt.core) - defines the classes that describe the Java model.
- [org.eclipse.jdt.core.compiler](http://org.eclipse.jdt.core.compiler) - defines an API for the compiler infrastructure.
- [org.eclipse.jdt.core.dom](http://org.eclipse.jdt.core.dom) - supports Abstract Syntax Trees (AST) that can be used for examining the structure of a compilation unit down to the statement level.
- [org.eclipse.jdt.core.dom.rewrite](http://org.eclipse.jdt.core.dom.rewrite) - supports rewriting of Abstract Syntax Trees (AST) that can be used for manipulating the structure of a compilation unit down to the statement level.
- [org.eclipse.jdt.core.eval](http://org.eclipse.jdt.core.eval) - supports the evaluation of code snippets in a scrapbook or inside the debugger.
- [org.eclipse.jdt.core.formatter](http://org.eclipse.jdt.core.formatter) - supports the formatting of compilation units, types, statements, expressions, etc.
- [org.eclipse.jdt.core.jdom](http://org.eclipse.jdt.core.jdom) - supports a Java Document Object Model (DOM) that can be used for walking the structure of a Java compilation unit. (deprecated – use [org.eclipse.jdt.core.dom](http://org.eclipse.jdt.core.dom))
- [org.eclipse.jdt.core.search](http://org.eclipse.jdt.core.search) - supports searching the workspace's Java model for Java elements that match a particular description.
- [org.eclipse.jdt.core.util](http://org.eclipse.jdt.core.util) - provides utility classes for manipulating .class files and Java model elements.



# Java Model



# Java Compilation Unit

The image displays a screenshot of an IDE window showing a Java source file named `MyClass.java` and its corresponding Outline view. The code in the editor is as follows:

```
package com.mycompany.sample;

import java.util.Date;

public class MyClass {

    static int count = 0;
    private int localCount;

    MyClass() {
        localCount = 0;
    }

    public static void main(String[] args) {
        new MyClass().doSomething();
    }

    void doSomething() {
        count++;
        localCount++;
    }
}
```

The Outline view on the right shows the structure of the code:

- com.mycompany.sample
  - import declarations
  - MyClass
    - count : int
    - localCount : int
    - MyClass()
    - main(String[])
    - doSomething()

Arrows point from the Outline view to the corresponding code elements in the editor, with labels indicating the type of element:

- IPackageDeclaration** points to `package com.mycompany.sample;`
- IImportDeclaration** points to `import java.util.Date;`
- IType** points to `public class MyClass {`
- IField** points to `static int count = 0;` and `private int localCount;`
- IMethod** points to `MyClass() {`, `public static void main(String[] args) {`, and `void doSomething() {`

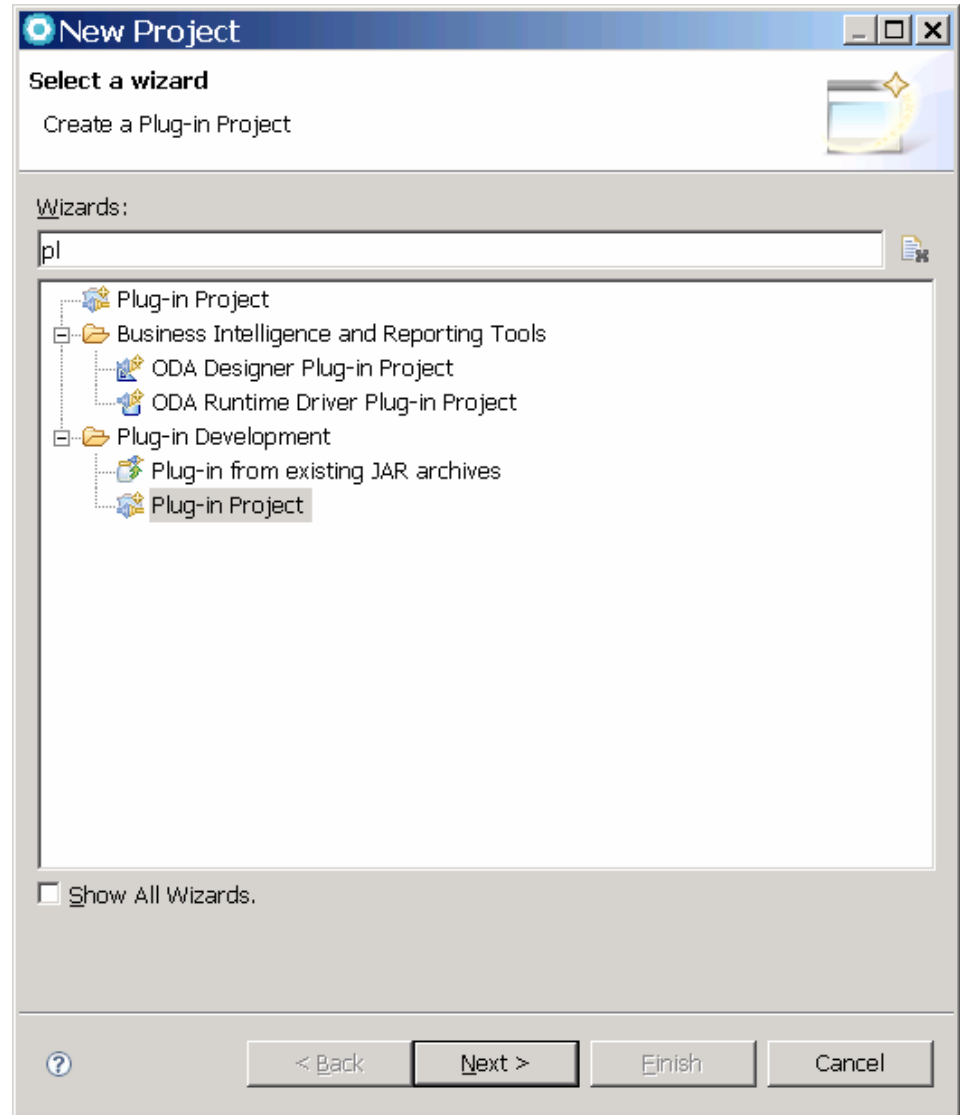
# Code modification using the DOM/AST API

- **AST (Abstract Syntax Tree)**
- **Ways to create a compilation unit**
  - [ASTParser](#)
  - [ICompilationUnit#reconcile\(...\)](#)
    - start from scratch using the factory methods on [AST](#) (Abstract Syntax Tree).
- **Creating AST from Source code – ASTParser. [createAST\(IProgressMonitor\)](#)**
  - [setSource\(char\[\]\)](#): to create the AST from source code
  - [setSource\(IClassFile\)](#): to create the AST from a classfile
  - [setSource\(ICompilationUnit\)](#): to create the AST from a compilation unit





# Plug-in Development Wizard



### New Plug-in Project

**Plug-in Project**  
Create a new plug-in project

Project name:

Use default location

Location:

**Project Settings**

Create a Java project

Source folder:

Output folder:

**Target Platform**

This plug-in is targeted to run with:

Eclipse version:

an OSGi framework:

**Working sets**

Add project to working sets

Working sets:

### New Plug-in Project

**Plug-in Content**  
Enter the data required to generate the plug-in.

**Plug-in Properties**

Plug-in ID:

Plug-in Version:

Plug-in Name:

Plug-in Provider:

Execution Environment:

**Plug-in Options**

Generate an activator, a Java class that controls the plug-in's life cycle

Activator:

This plug-in will make contributions to the UI

Enable API Analysis

**Rich Client Application**

Would you like to create a rich client application?  Yes  No



### New Plug-in Project

Templates

Select one of the available templates to generate a fully-functioning plug-in.

Create a plug-in using one of the templates

Available Templates:

- Custom plug-in wizard
- Figure definitions converter
- Hello, World
- Hello, World Command
- ODA Data Source Designer
- ODA Data Source Runtime Driver
- Plug-in with a multi-page editor
- Plug-in with a popup menu
- Plug-in with a property page
- Plug-in with a view**
- Plug-in with an editor
- Plug-in with an incremental project bui
- Plug-in with sample help content

This wizard creates standard plug-in directory structure and adds the following:

- Sample view.** This template create a workbench view. The view is contributed to the workbench by creating a category. The view can be opened by selecting **Window, Show View** and then **Other...** on the menubar. The template demonstrates implementation of pop-up menu support, local tool bar, double-click, sorting and filtering. There is also an option to add context-sensitive help the view.

**Extensions Used**

- org.eclipse.ui.views
- org.eclipse.ui.perspectiveExtensions

< Back    Next >    Finish    Cancel

### New plug-in project with a sample view

Main View Settings

Choose the way the new view will be added to the plug-in.

Java Package Name: com.ibm.example.cts2.views

View Class Name: SampleView

View Name: Sample View CTS2

View Category ID: com.ibm.example.cts2

View Category Name: Sample Category

Select the viewer type that should be hosted in the view:

Table viewer (can also be used for lists)     Tree viewer

Add the view to the java perspective

Add context help to the view

?    < Back    Next >    Finish    Cancel



com.ibm.example.cts1

- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies
- src
- icons
- META-INF
- build.properties
- contexts.xml
- plugin.xml

com.ibm.example.cts2

- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies
- src
- icons
- META-INF
- MANIFEST.MF
- build.properties
- contexts.xml
- plugin.xml

JDTProj1

NewVersion\_ProjMetricReport1

ProjBIRTReport1

ProjJavaReports

TestJavaProj1

### Overview

#### General Information

This section describes general information about this plug-in.

ID:

Version:

Name:

Provider:

Platform Filter:

Activator:

- Activate this plug-in when one of its classes is loaded
- This plug-in is a singleton

#### Execution Environments

Specify the minimum execution environments required to run this plug-in.

JavaSE-1.6	<input type="button" value="Add..."/>
	<input type="button" value="Remove"/>
	<input type="button" value="Up"/>
	<input type="button" value="Down"/>

[Configure JRE associations...](#)

[Update the classpath settings](#)

#### Plug-in Content

The content of the plug-in is made up of two sections:

- [Dependencies](#): lists all the plug-ins required on this plug-in's classpath to compile and run.
- [Runtime](#): lists the libraries that make up this plug-in's runtime.

#### Extension / Extension Point Content

This plug-in may define extensions and extension points:

- [Extensions](#): declares contributions this plug-in makes to the platform.
- [Extension Points](#): declares new function points this plug-in adds to the platform.

#### Testing

Test this plug-in by launching a separate Eclipse application:

- [Launch an Eclipse application](#)
- [Launch an Eclipse application in Debug mode](#)

#### Exporting

To package and export the plug-in:

1. Organize the plug-in using the [Organize Manifests Wizard](#)
2. Externalize the strings within the plug-in using the [Externalize Strings Wizard](#)
3. Specify what needs to be packaged in the [Deployable](#)

Overview

- Dependencies
  - org.eclipse.ui
  - org.eclipse.core.runtime
- Runtime
- Extensions
  - org.eclipse.ui.views
  - org.eclipse.ui.perspec
  - org.eclipse.help.conte
- Extension Points
- Build



# Context help with contexts.xml

Plug-in Development - com.ibm.example.cts2/contexts.xml - Rational® Application Developer™ for WebSphere® Software

File Edit Navigate Search Project Run Window Help

Package Explorer Plug-ins com.ibm.example.cts2 contexts.xml Outline

com.ibm.example.cts1

- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies
- src
- icons
- META-INF
- build.properties
- contexts.xml
- plugin.xml

com.ibm.example.cts2

- JRE System Library [JavaSE-1.6]
- Plug-in Dependencies
- src
- icons
- META-INF
- MANIFEST.MF
- build.properties
- contexts.xml
- plugin.xml

JDTPProj1

- NewVersion\_ProjMetricReport1
- ProjBIRTReport1
- ProjJavaReports
- TestJavaProj1

Context Help

Register this context help file ?

Context Help

Edit the structure of the context help file in this section.

type filter text

- contexts
  - viewer
    - This is the context help for th
    - Context-sensitive help

Add Context

Add Topic

Add Command

Remove

Up

Down

Topic Details

Specify the display label for this topic:

Label: Context-sensitive help

Specify the location of an HTML file containing content:

Location: /PLUGINS\_ROOT/org.eclipse.platform Browse...

Definition Source

Outline

- Definition
  - contexts
    - viewer
      - This is the conte
      - Context-sensitiv



# Run Configuration for the plug-in

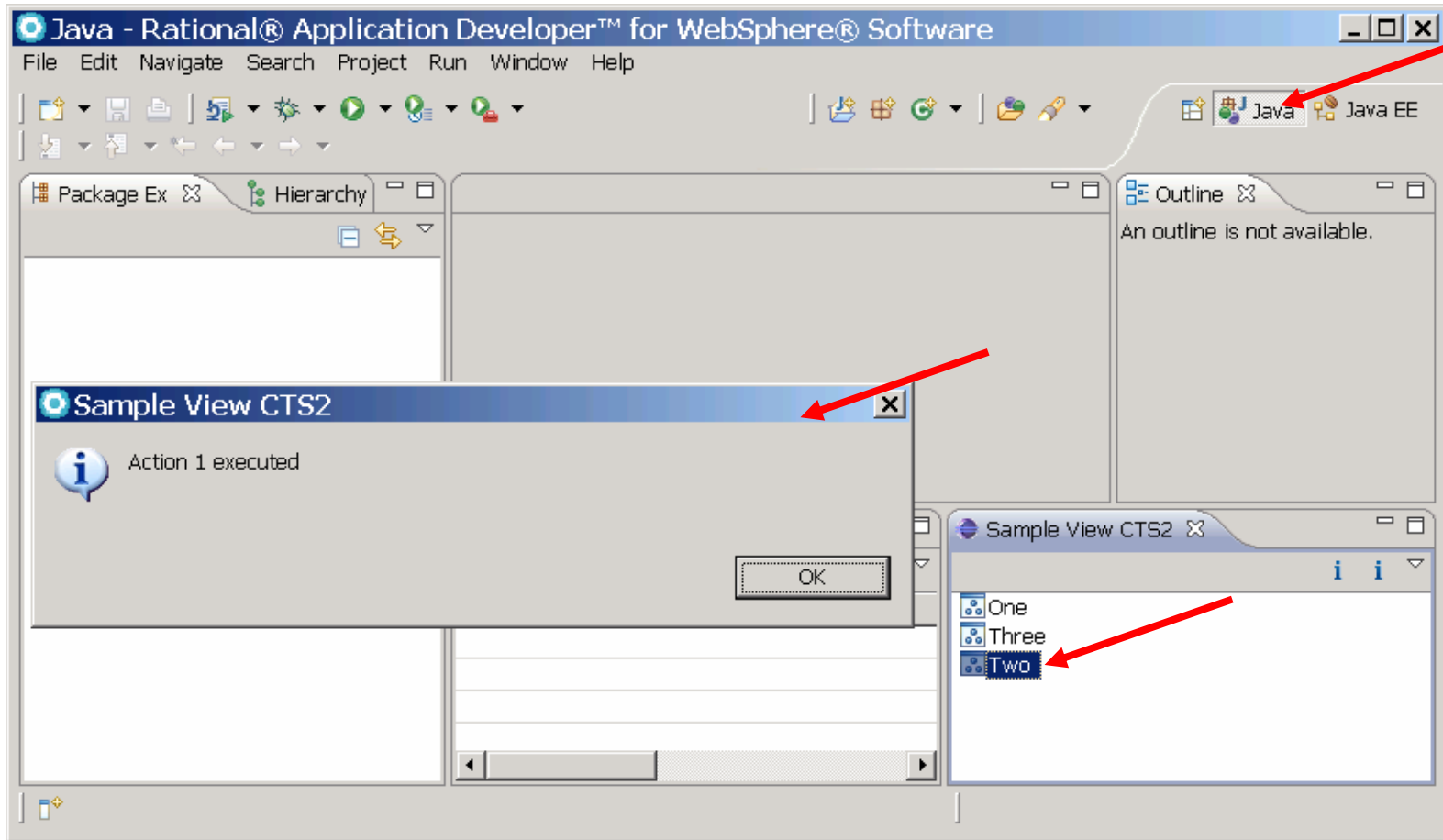
The screenshot shows the Eclipse Run Configurations dialog box for a configuration named "New\_configuration". The dialog is titled "Run Configurations" and has a subtitle "Create, manage, and run configurations". Below the subtitle is the instruction "Create a configuration to launch an Eclipse application." and a green play button icon.

The main area of the dialog is divided into several sections:

- Name:** "New\_configuration"
- Launch with:** "plug-ins selected below only" (indicated by a dropdown arrow)
- Plug-ins:** A tree view showing the selected plug-ins. The "Workspace" is checked. Under "Workspace", "com.ibm.example.cts1 (1.0.0)" and "com.ibm.example.cts2 (1.0.0)" are selected. A red arrow points to "com.ibm.example.cts2 (1.0.0)". Under "Target Platform", several plug-ins from "com.businessobjects.crystalreports.designer" are selected, including "dseintegration (1.0.5.v1246)", "core (1.0.5.v1246)", "editor (1.0.5.v1246)", "enginepreferences (1.0.5.v1246)", "preview (1.0.5.v1246)", and "enterprise.integration (1.0.5.v1246)".
- Buttons:** "Select All", "Deselect All", "Add Working Set...", "Add Required Plug-ins", "Restore Defaults", "Only show selected plug-ins" (checkbox), and "Validate Plug-ins".
- Options:** "Include optional dependencies when computing required plug-ins" (checked), "Add new workspace plug-ins to this launch configuration automatically" (checked), and "Validate plug-ins automatically prior to launching" (unchecked).
- Footer:** "Apply", "Revert", "Run", and "Close" buttons.

The left sidebar shows a list of configuration types, with "New\_configuration" selected. The filter "type filter text" is applied, and the status bar indicates "Filter matched 22 of 22 items".

# Our view in action !



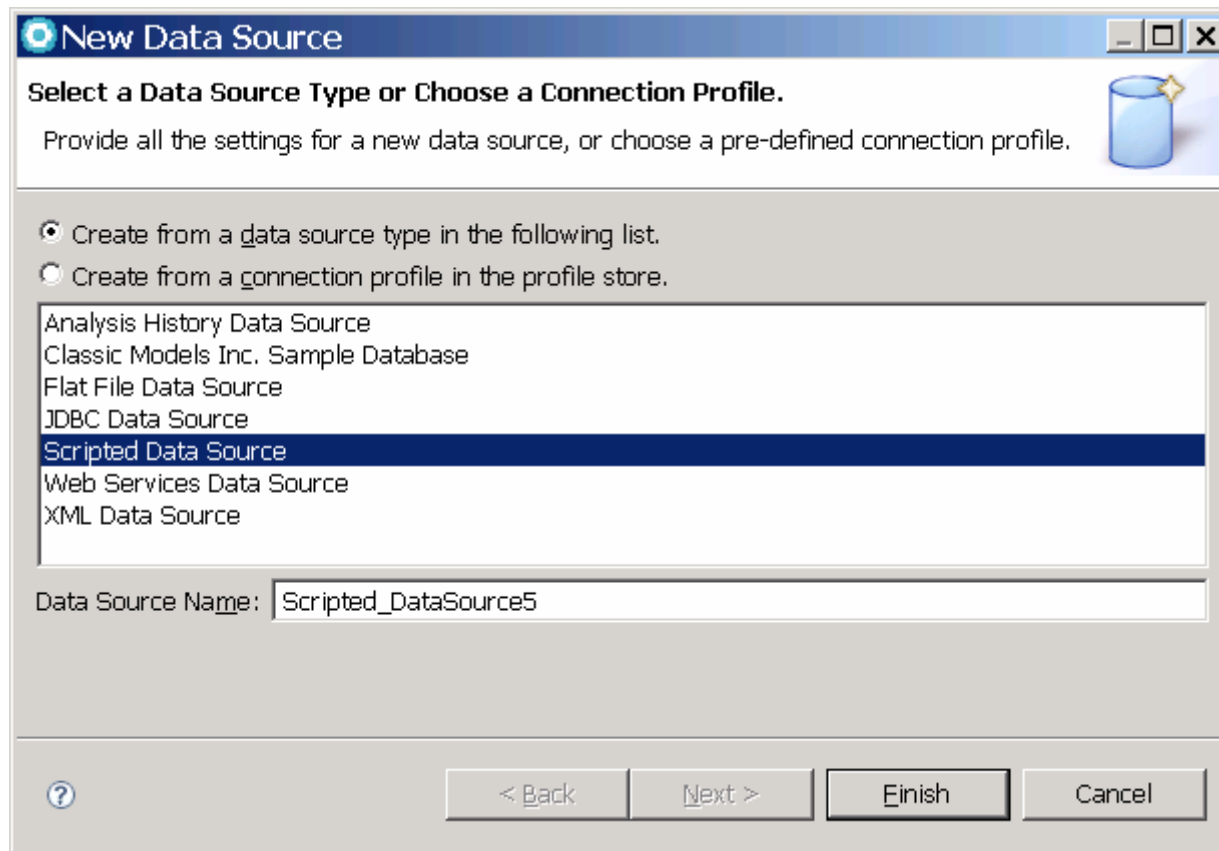
# Reporting Infrastructure

- **RAD supports 2 ways for building reports**
  - BIRT
  - Crystal Reports
- **Designing Reports with BIRT**
  - Report Layout
    - Colors, fonts and positioning
  - DataSources
    - Can be JDBC /XML/Scripted Data Source/Web Service
  - DataSets
    - Corresponds to data records used in the details added dynamically to the report
- **Caching Build Reports**
  - Either Data or the built report can be cached





# Data Source Types



# Scripting Data Set -Steps

**New Data Set**

Create a new data set.

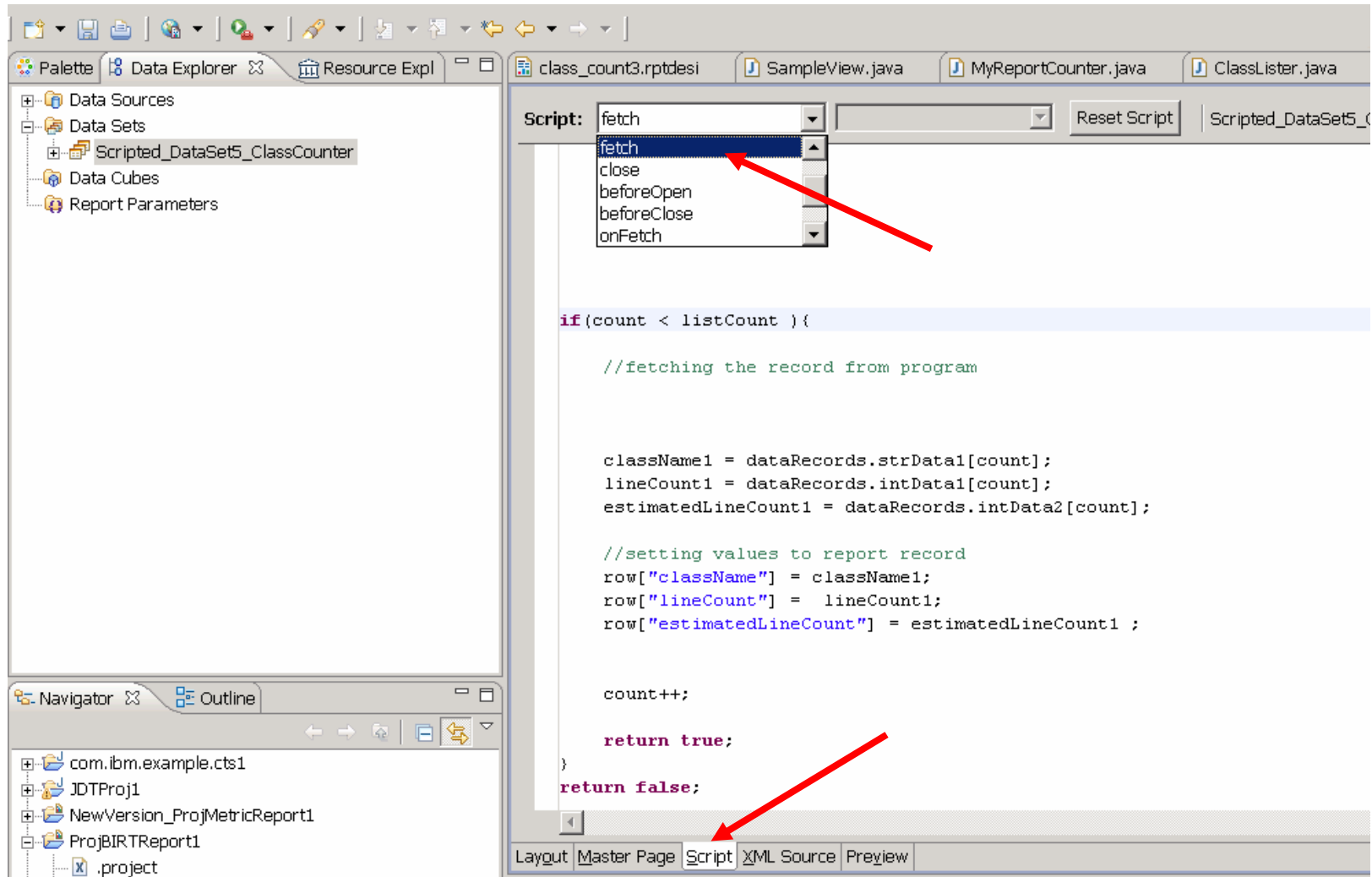
Data Set Name:

Data Set Type:

Data Source:



# Scripting Data Set -Steps



The screenshot displays the IBM Rational Software Conference 2009 interface for scripting a data set. The main window is titled "class\_count3.rptdesi" and shows a script editor with the following code:

```
Script: fetch  
fetch  
close  
beforeOpen  
beforeClose  
onFetch  
  
if(count < listCount ) {  
  
    //fetching the record from program  
  
    className1 = dataRecords.strData1[count];  
    lineCount1 = dataRecords.intData1[count];  
    estimatedLineCount1 = dataRecords.intData2[count];  
  
    //setting values to report record  
    row["className"] = className1;  
    row["lineCount"] = lineCount1;  
    row["estimatedLineCount"] = estimatedLineCount1 ;  
  
    count++;  
  
    return true;  
}  
return false;
```

The interface includes a Palette on the left with "Data Sources", "Data Sets", "Scripted\_DataSet5\_ClassCounter", "Data Cubes", and "Report Parameters". The Data Explorer shows a tree structure with "com.ibm.example.cts1", "JDTProj1", "NewVersion\_ProjMetricReport1", "ProjBIRTRReport1", and ".project". The bottom right corner shows the "Script" tab selected, with a red arrow pointing to it.

# XML Data Source – Schema & Source

**New XML Data Source Profile**

**Define the URLs to the XML file and schema information**

Define the URLs to the XML file and schema information.

Enter the URL of the XML source or browse to the file containing the data:

Enter the URL of the XML schema or browse to the file containing the schema. Leave this empty if no schema is available:

Select encoding for the XML source and schema specified above:



# XML Data Set - Computed Columns

**Edit Data Set - XMLClassCountDataSet2**  
 Computed Columns  
 Define computed columns:

Column Name	Data Type	Expression

**New Computed Column.**  
 Column Name: Difference  
 Data Type: Integer  
 Aggregation:   
 Expression:   
 Filter:   
 Expression can not be blank  
 OK Cancel

**Expression Builder**  
 Type an expression in the Expression field. Browse the lists of available objects and double-click to copy  
 1 Math.abs(row["estimated"]-row["lineCount"])

Operators: + - \* / ! = < > & | ( )  
 lineCount  
 Category: Available Data Sets Native JavaScript Functions BIRT Functions Operators  
 Sub-Category: XMLClassCountDataSet2  
 Double Click: className lineCount estimated

# Building a chart

**New Chart**

Select chart type and choose an output format.

Select Chart Type | Select Data | Format Chart

**Chart Preview**

**Bar Chart Title**

Category	Value (Series 1)
A	6
B	4
C	12
D	8
E	10

Select Chart Type

- Bar
- Line
- Area
- Pie
- Meter
- Scatter
- Stock
- Bubble
- Difference
- Gantt
- Tube
- Cone
- Pyramid

Select Subtype

Dimension: 2D | Output Format: SVG

Multiple Y Axis: None | Series Type: Bar Series

Orientation:  Flip Axis

< Back | **Next >** | Finish | Cancel | Apply





# Getting Chart Data from the Data Set

**New Chart**

Select the data to display in the chart and bind it to the series.

Select Chart Type | Select Data | Format Chart

**Chart Preview**

Bar Chart Title

Value (Y) Series: Series 1  
 Σ row["estimated"]

Optional Y Series Grouping:

Category (X) Series: row["className"]

**Select Data**

Inherit Data from Container  
 Use Data from XMLClassCountDataSet2

**Data Preview**

Use the right-click menu or drag the column into series fields.

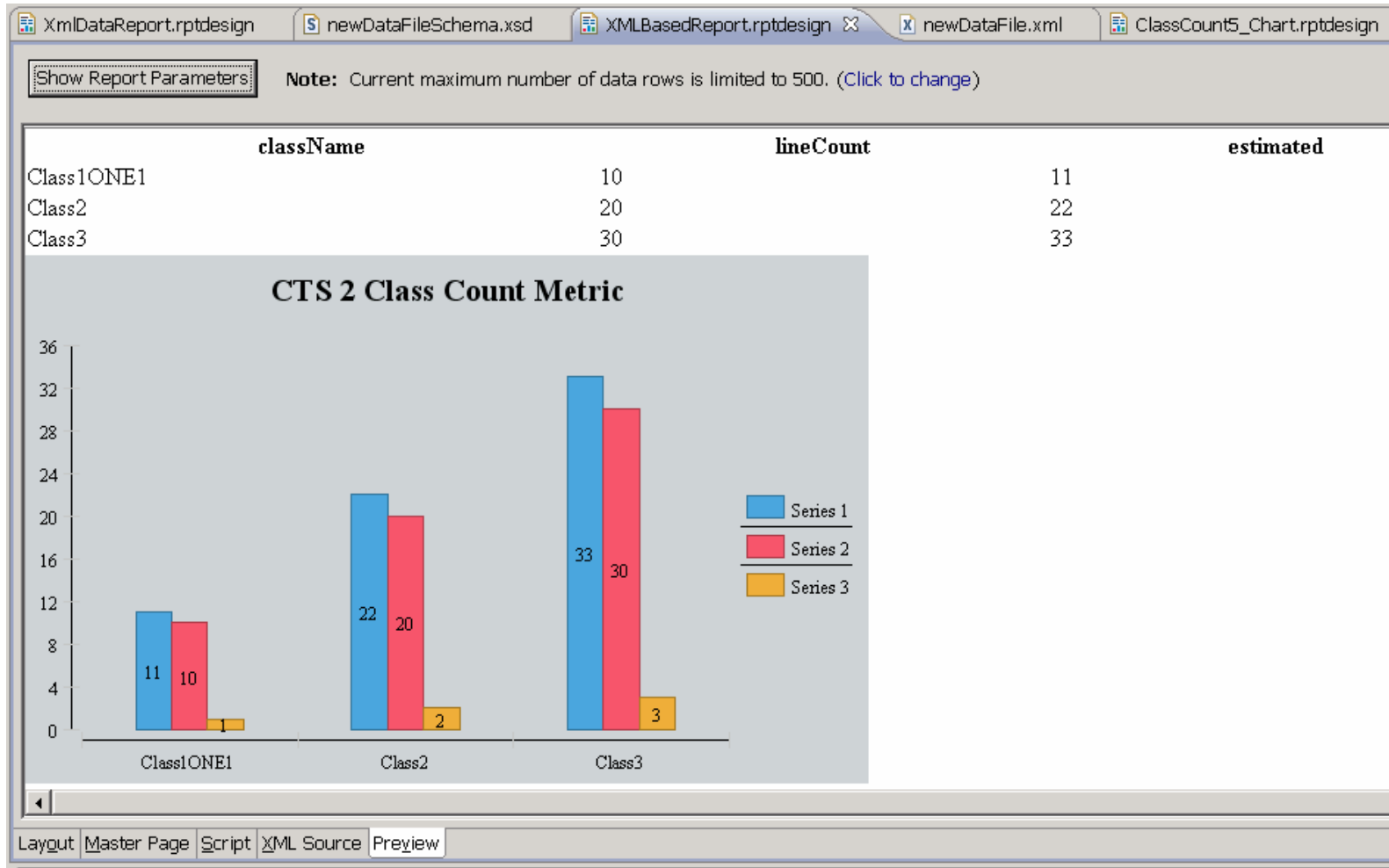
className	lineCount	estimated	Difference
Class1ONE1	10	11	1
Class2	20	22	2
Class3	30	33	3

Filters...  
Parameters...  
Data Binding...

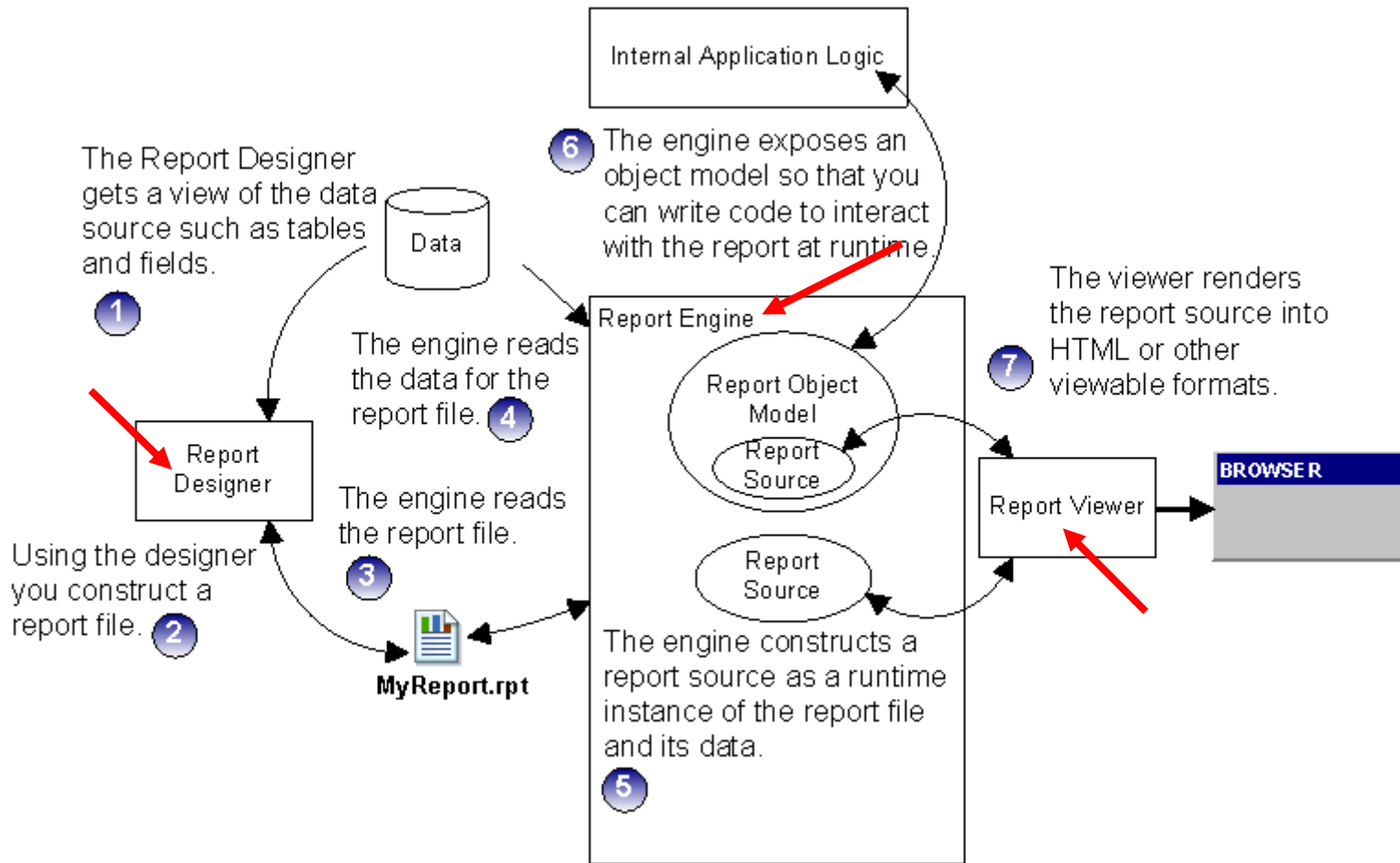
< Back | Next > | Finish | Cancel | Apply



# Preview Report



# Crystal Reports



# CR Reporting Models

- CR Embedded Reporting Model
  - uses the Java Reporting Component (JRC) and Crystal Reports Viewers Java SDK
    - to enable users to view and export reports in a web browser
    - functionality required to create and customize a report viewer object, process the report, and then render the report in DHTML.
  - The JRC (jars) keeps report processing completely internal to the Java application server to process Crystal Reports report (.rpt) files within the application itself, no external report servers
- CR Enterprise Reporting Model
  - uses the Crystal Enterprise Java SDK to leverage an external Crystal Enterprise server
  - additional functionality
    - runtime report creation
    - persisting runtime report modification back to the Crystal Reports report (.rpt) file
    - report management, security, and scheduling
    - The Crystal Enterprise server also improves scalability and increases performance to support extensive user concurrency demands.



# Developing Custom JSF Web Components

- **RAD provides for**
  - Importing Custom component Libraries
  - Building Custom JSF Component Library
  - Adding new custom JSF widgets in the library
  - Adding custom library widgets to the RAD palette
  - Sharing and using custom widgets by drag and drop from the palette

```
<h:outputText value="Name:" / >  
<h:inputText value="#{person.name}" />
```

```
<my:inputLabel value="#{person.name}" label="Name:" />
```



# New Faces Component Library wizard

**New Faces Component Library**

**Faces Component Library**  
Create a new Faces Component Library

Project name:

**Contents**

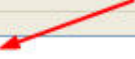
Use default

Directory:

**Target Runtime**

**Dynamic Web Module version**

**Configuration**



This configuration includes all of the functionality that you require to create a component library and custom Faces components.

**EAR Membership**

Add project to an EAR

EAR Project Name:

**New Faces Component Library**

**Component Library settings**  
Configure Component Library

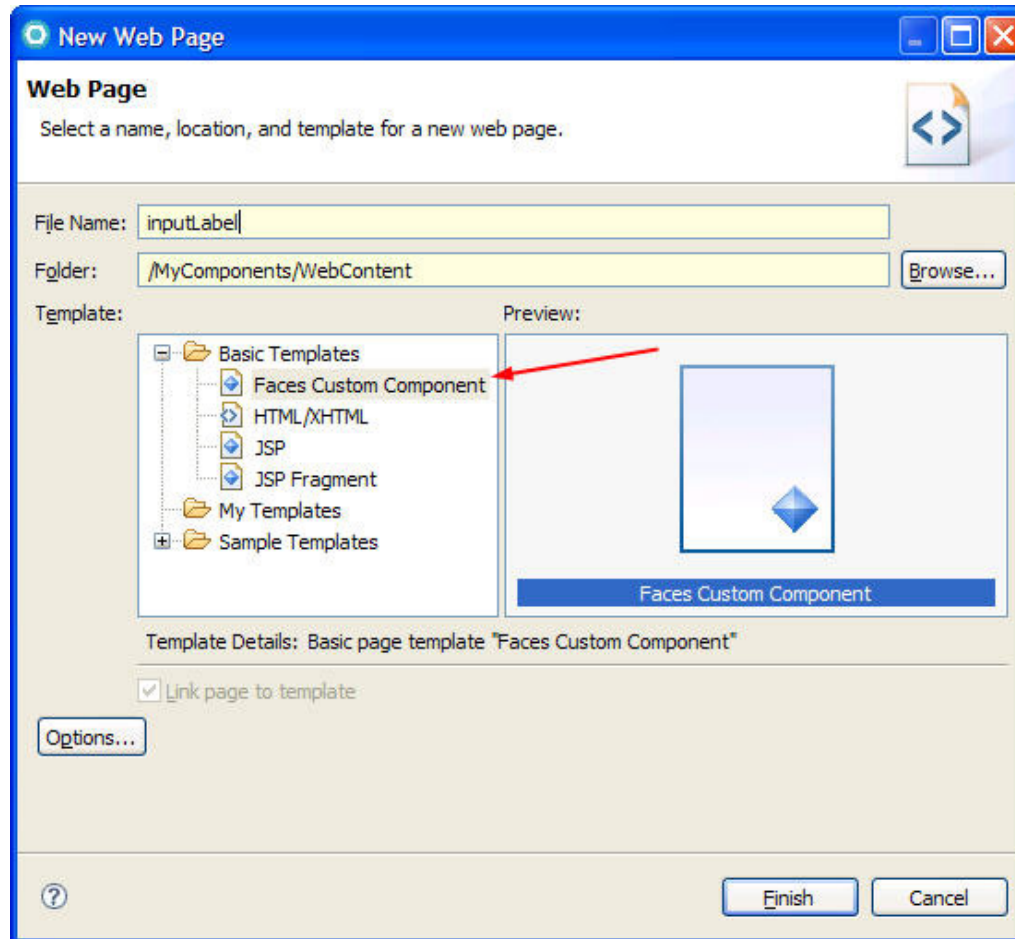
URI for generated tag library:

Prefix of generated tags:

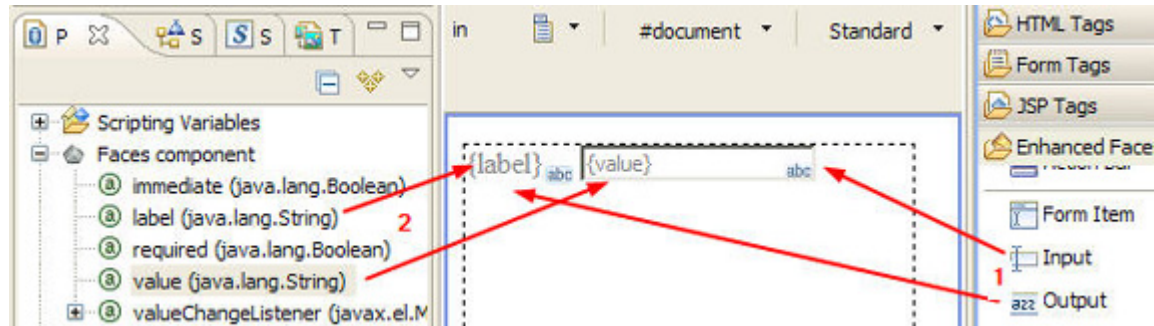
Java package prefix for generated classes:



## New custom component



## Creating component content



## Component source

```

<jsfc:component extends="javax.faces.component.UIInput">
  <h:outputText styleClass="outputText" id="text1"
    value="#{component.label}"></h:outputText>
  <h:inputText styleClass="inputText" id="text2"
    value="#{component.value}"></h:inputText>
</jsfc:component>

```



Tag name:

Description:

Component behaves as:  ▼

Basic component simply acts as a container for other JSF components inside it.

Do not overwrite Java classes when this JSP changes

Attributes:

Name	Class	Required

## <jsfc:component> Component properties

**Define component attribute** [X]

Name:

Class:  [📄]

Description:

Required attribute

[?]

Tag name:

Description:

Component behaves as:  ▼

Input component has a value attribute and supports ValueChanged events.

Do not overwrite Java classes when this JSP changes

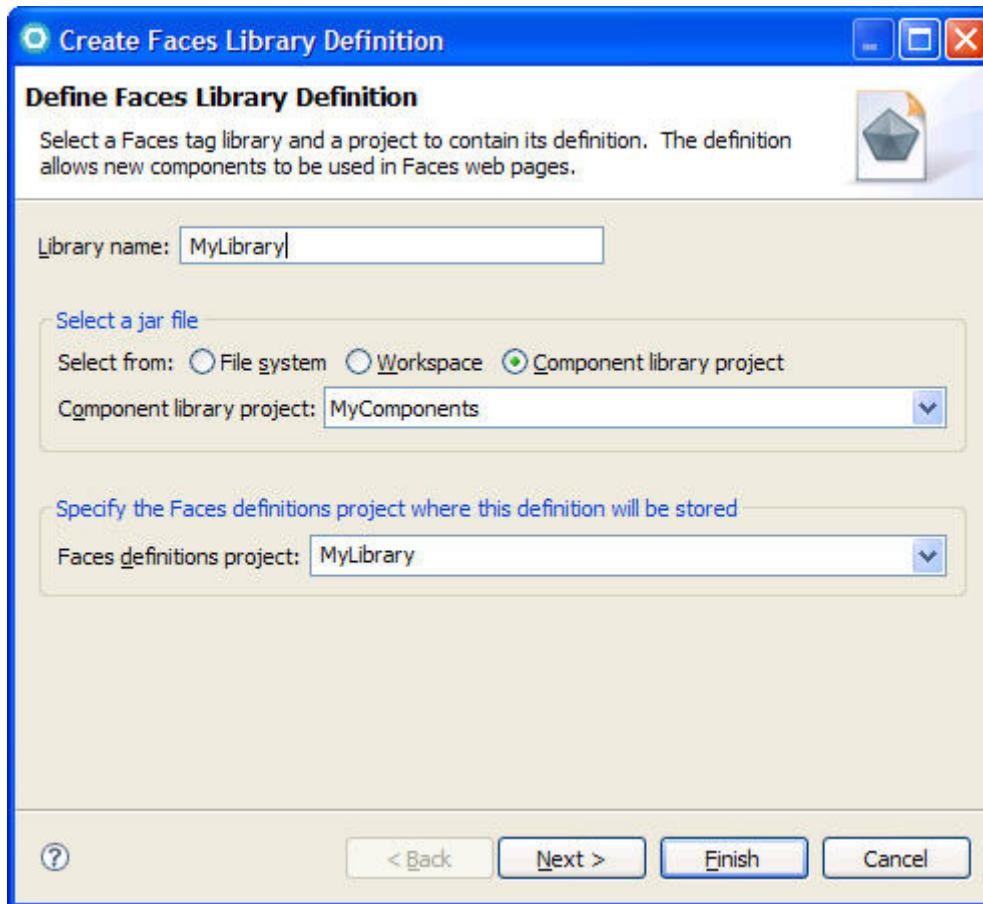
Attributes:

Name	Class	Required
value	java.lang.String	false
label	java.lang.String	false
immediate	java.lang.Boolean	false
required	java.lang.Boolean	false

## Configured component



## Library definition



**Create Faces Library Definition**

**Define Faces Library Definition**

Select a Faces tag library and a project to contain its definition. The definition allows new components to be used in Faces web pages.

Library name:

Select a jar file

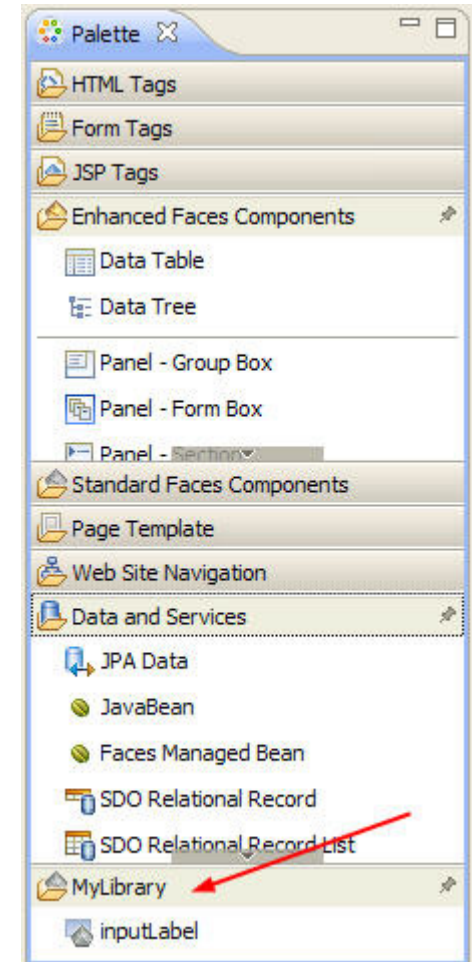
Select from:  File system  Workspace  Component library project

Component library project:

Specify the Faces definitions project where this definition will be stored

Faces definitions project:

## Added to RAD Palette



# Building Visual Custom Tags

- **Visualizing Custom Tags in the design view**
  - Building custom plug-in to visualize my tag
  - Extend CustomTagVisualizer
  - Provide the visualization information in doStart or doEnd Methods
  - Building and importing new plug-in
  - Add custom properties view for the custom tag

## Sample plugin.xml extract

```
<requires>
  <samp><import plugin="org.apache.xerces" />
  <samp><import plugin="com.ibm.etools.webedit.core" />
</requires>
<extension point="com.ibm.etools.webedit.core.visualCustomTag">
  <samp><vtaglib uri="/WEB-INF/lib/sample.jar">
    <samp><vtag name="date"
      class="com.ibm.etools.webedit.vct.sample.DateTimeTagVisualizer"
      description="Dynamically outputs current date and time"/>
    <samp></vtaglib>
  </extension>
```



# Example Custom tag visualized

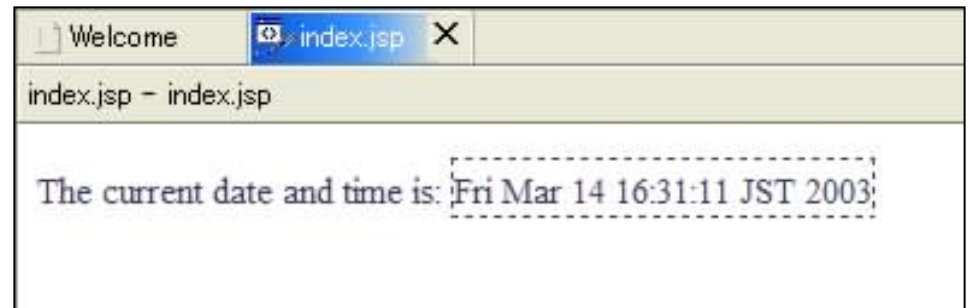
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML> <HEAD>
```

```
<%@ taglib uri="/WEB-INF/lib/sample.jar" prefix="vct" %>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<TITLE>index.jsp</TITLE> </HEAD>
<BODY>
```

**The current date and time is: <vct:date/>**

```
</BODY>
</HTML>
```



```
import com.ibm.etools.webedit.vct.*;
```

```
public class DateTimeTagVisualizer extends CustomTagVisualizer {
    public VisualizerReturnCode doEnd(Context context) {
        Date now = new Date();
        context.putVisual(now.toString());
        return VisualizerReturnCode.OK;
    }
}
```



# Resources

- **RAD**

- <http://publib.boulder.ibm.com/infocenter/radhelp/v7r5/index.jsp>

- **JDT**

- [http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.jdt.doc.isv/guide/jdt\\_int\\_model.htm](http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.jdt.doc.isv/guide/jdt_int_model.htm)
- <http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/org.eclipse.jdt.doc.isv/reference/api/org/eclipse/jdt/core/dom/AST.html>
- [http://www.jdg2e.com/ch27\\_jdt/doc/index.html](http://www.jdg2e.com/ch27_jdt/doc/index.html)
- [http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.jdt.doc.isv/guide/jdt\\_int\\_model.htm](http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.jdt.doc.isv/guide/jdt_int_model.htm)

- **BIRT**

- <http://www.eclipse.org/birt/phenix/>
- [http://wiki.eclipse.org/Integration\\_Examples\\_%28BIRT%29](http://wiki.eclipse.org/Integration_Examples_%28BIRT%29)
- <http://www.vogella.de/articles/EclipseBIRT/article.html>
- <http://download.eclipse.org/birt/downloads/examples/scripting/scripteddatasource/scripteddatasource.html>
- <https://www6.software.ibm.com/developerworks/education/dw-r-umlbirtreport/index.html> (UML Model reports in RSA)

- **Crystal Reports**

- <http://publib.boulder.ibm.com/infocenter/radhelp/v6r0m1/index.jsp?topic=/com.businessobjects.integration.eclipse.doc.devtools/developer/ArchitectureOverview2.html>

- **Plug-in development**

- <http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/firstplugin.htm>

- **Web Tools Customization**

- [http://www.ibm.com/developerworks/websphere/library/techarticles/0304\\_hosokawa/hosokawa.html](http://www.ibm.com/developerworks/websphere/library/techarticles/0304_hosokawa/hosokawa.html)
- [http://www.ibm.com/developerworks/rational/library/09/0106\\_kats/](http://www.ibm.com/developerworks/rational/library/09/0106_kats/)



# A Case Study @ CTS

- **Extracting Quality metrics from Source code**
  - Using available CTS project metric tools
- **Packaging the custom plug-ins and Integrating to existing QA systems**
  - Modes of running the customized plug-ins
- **Usage tracking**
- **Productivity tracking**



# Extending RAD @ CTS...



# GTO structure



**Cognizant**

**Global Technology Office**

Passion for making a difference



Innovative and differentiating solutions deploying emerging technologies

Solutions focused towards reuse, user productivity and software management

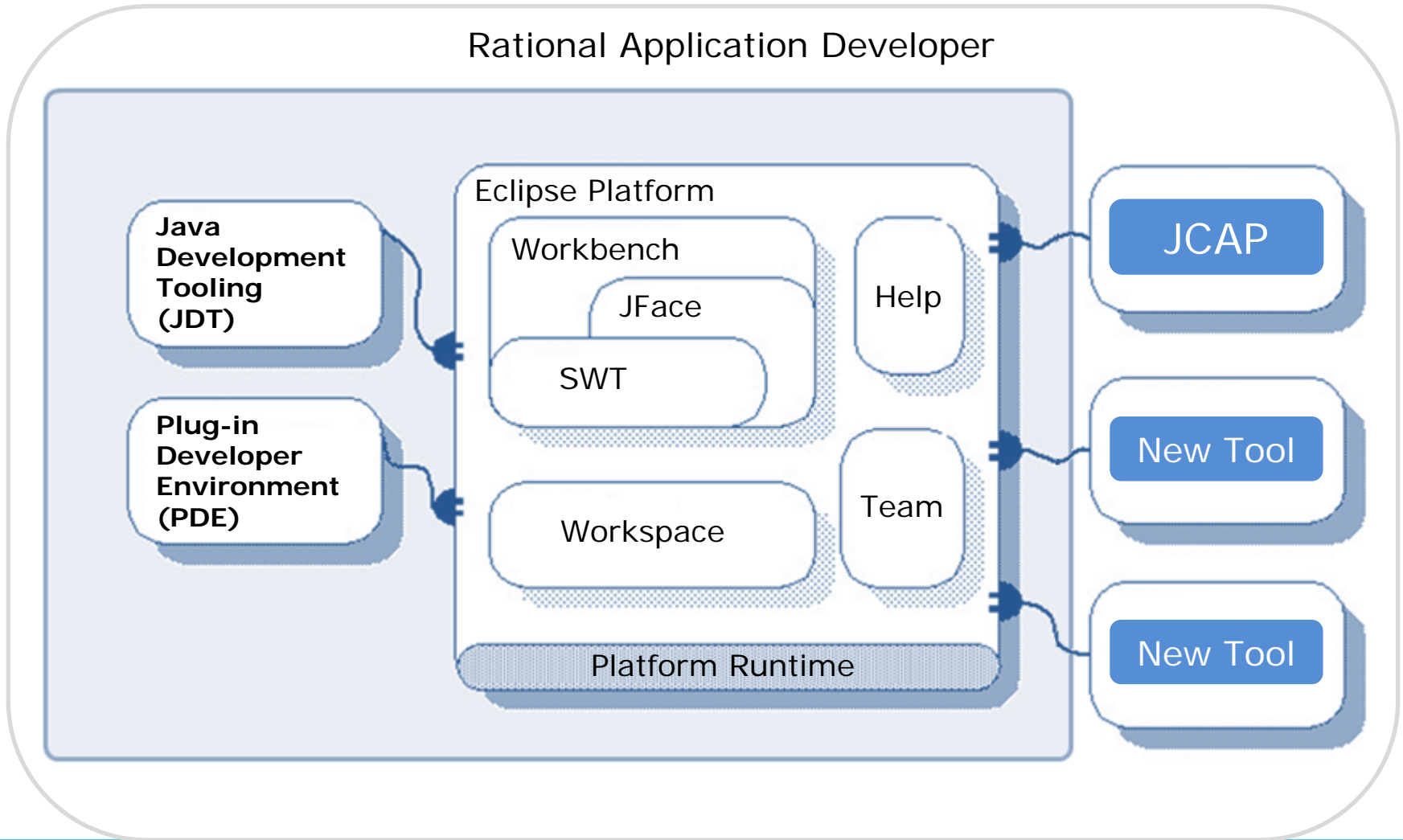
Practice & methodology with focus on building agile, scalable and high performance enterprise architecture

In-depth & end to end coverage of the core technologies with focus on the development, productivity & quality





# RAD Extension – JCAP Plug-in



# JCAP – Java Code Assessment Platform



## Objective

- To build extensions to Rational Products to:
  - » Collect
  - » Analyze
  - » Integrate Code quality Metrics with Cognizant Delivery Platform

## Project Quality Management - Need of the Hour

- Project Code Quality Metrics at development milestones, On demand
  - » Overall Project health to be known at build time (entire project code base)
  - » Also Get to know the application health from time to time On demand
- Individual developer's work quality - required for mentoring new recruits/trainees
- Monitor in a continuous basis – decreasing trend or increasing trend



## JCAP – Features



- **Java Code Assessment Platform** (henceforth called as **JCAP**) – implemented as a RAD Extension RAD extensions provided as eclipse plug-ins; extensions to the Menu, Toolbar, Project Explorer and View
- **Data Acquisition**
  - » Source Analysis & Metrics Capture
  - » Violations against Coding standard rules
  - » Size metrics [Lines of Code and Documentation Lines of Code]
  - » Cyclomatic Complexity
  - » Code duplication
  - » Class coupling
- **Data Integration and Reporting**
  - » IDE level dashboards
  - » Web dashboards integrated with the Organization Governance dashboards



# JCAP Plug-in

The screenshot shows the JCAP Configuration dialog box with the following components and their associated component names:

Component Name	Component Name
StatAnalysisEngine.java	Enter component name
ObservationComparator.java	Enter component name
STANPlugin.java	Enter component name
STANVisitorRunner.java	Enter component name
STANVisitor.java	Enter component name
STANAbstractVisitor.java	STAN
MarkerInfo.java	
STANProcessor.java	
actions	ACTION
STANRemoveMarkersAction.java	
RunAction.java	
views	VIEW
RemoveObservationAction.java	
SeverityFilterAction.java	
ProjectSelectionAction.java	
PartListener.java	
ObservationSorter.java	

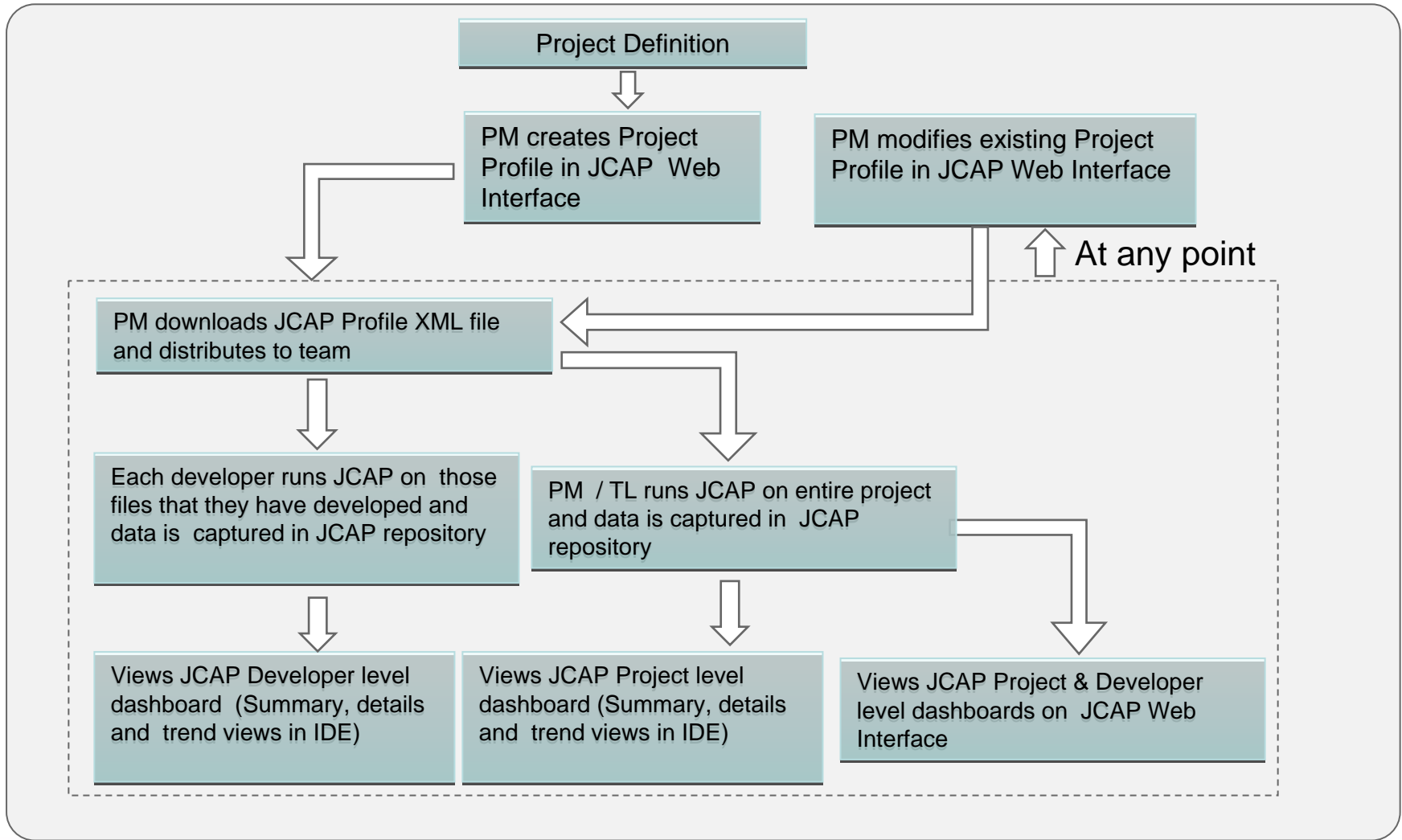
Annotations on the left side of the dialog:

- I JavaProject** points to the JCAPDemoProject folder.
- I PackageFragment** points to the com folder.
- I CompilationUnit** points to the StatAnalysisEngine.java file.

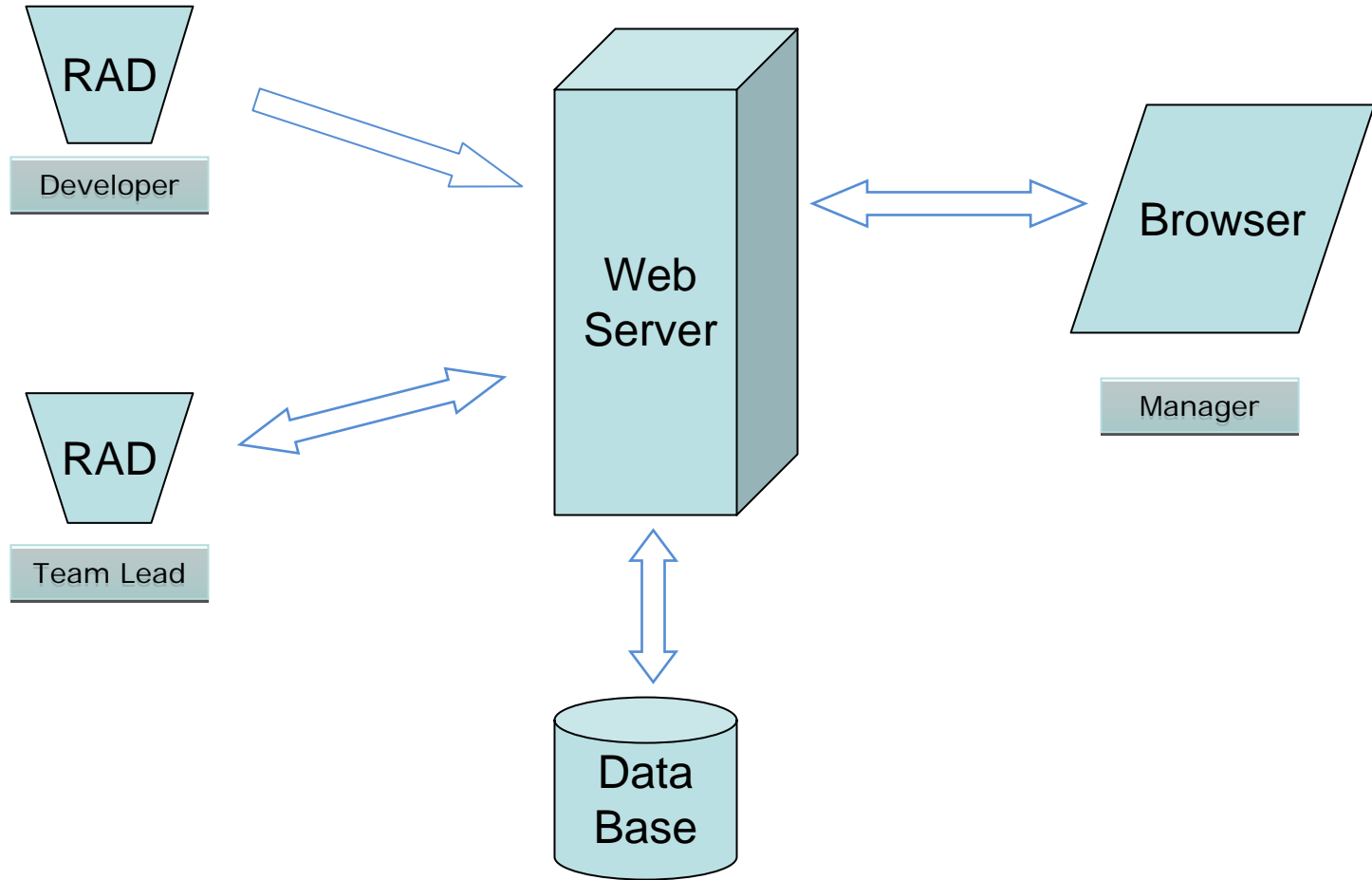
At the bottom of the dialog, there is a checkbox for "Capture for Trend Data", a "Run Level" section with radio buttons for "Project" (selected) and "Developer", and an "Analyze" button.



# JCAP – Functional view



# JCAP – Architectural View



# Data Reporting - Project Quality Dashboard

Code Assessment Platform - Microsoft Internet Explorer provided by Cognizant Technology Solutions

Address: http://localhost:8040/CAPWeb1.3/jsp/doLogin.action

## Java Code Assessment Platform

Profile Management | Code Quality

Project ID - 22211 Dashboard Data Collection - Wed, 5 Aug 2009

Select Project: 22211 | Select Developer: Project

### Scoring Metrics as on 2009-08-05

Metric Name	Value(per 100 loc)
Inline-doc	99.05
CyclomaticComplexity	11.06
Code Duplication	5.15

### PMD Summary as on

# File	LoC	Lines/File	Violation
58	3935	67.84	77

[Detail](#) | [Trend](#)

### CPD Summary as on

# File	LoC	Lines/File	Duplication
35	2111	60.31	21

[Detail](#)

### Scoring Metric Trend

**Metric Trend**

Date	Inline-doc	CyclomaticComplexity	Code Duplication
30/Jul	60	30	15
31/Jul	65	28	15
01/Aug	70	25	15
02/Aug	85	25	15
03/Aug	90	25	15
04/Aug	95	20	10
05/Aug	99.05	11.06	5.15

### Componentwise Trend

**PMD Summary Trend**

Date	Client	DataObj	Test	Total	Total LoC
30/Jul	100	100	100	300	3935
31/Jul	100	100	100	300	3935
01/Aug	100	100	100	300	3935
02/Aug	100	100	100	300	3935
03/Aug	100	100	100	300	3935
04/Aug	100	100	100	300	3935
05/Aug	77	11.06	5.15	93.21	3935

### Code Duplication (Copy-Paste) Count

**CPD Summary Trend**

Date	Total LoC	Duplication count
30/Jul	2111	21
31/Jul	2111	21
01/Aug	2111	21
02/Aug	2111	21
03/Aug	2111	21
04/Aug	2111	21
05/Aug	2111	21

### Developer Violation Trend

**Developer wise total PMD Violation count**

Date	178538	123776
30/Jul	55	30
31/Jul	55	30
01/Aug	40	25
02/Aug	30	20
03/Aug	35	25
04/Aug	15	15
05/Aug	12	10

Class : C1 - Highly Critical



# DEMO





# Questions





Thank You

© Copyright IBM Corporation 2009. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

