

Problem Determination of your WebSphere Applications with RAD

Chuck Bridgham, Senior Architect, RAD
cbridgha@us.ibm.com

IBM Software

Innovate2011

The Premier Event for Software and Systems Innovation



Software. Everywhere.

August 9-11, Bangalore | August 11, Delhi



Agenda

- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- Tuning the JVM
- Tuning WebSphere Applications
- Questions

Agenda

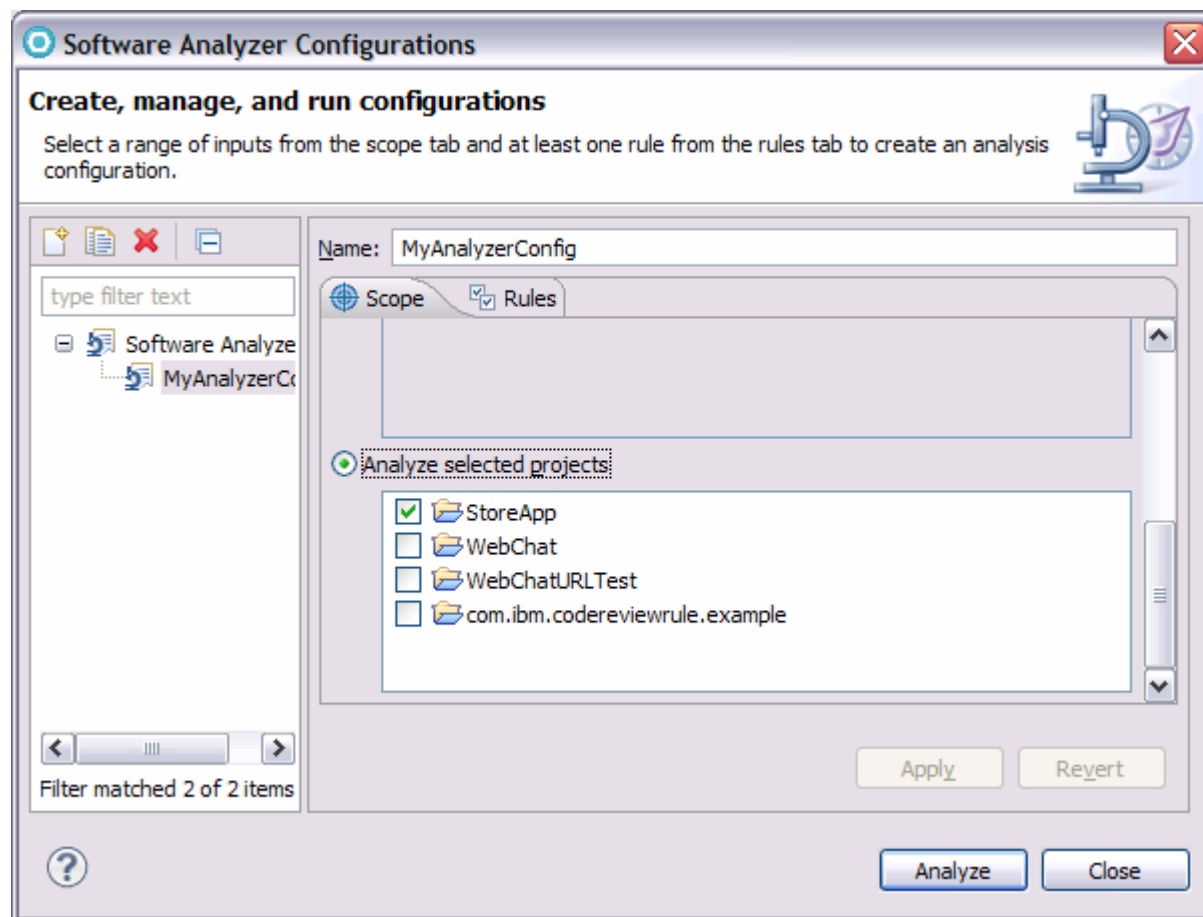
- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- Tuning the JVM
- Tuning WebSphere Applications
- Questions

Static Analysis in Rational Application Developer

- Java Code Review to find problems of various types:
 - ▶ Design Principles
 - ▶ Globalization
 - ▶ J2EE & J2SE Best Practices
 - ▶ J2EE & J2SE Security
 - ▶ Naming
 - ▶ Performance
 - ▶ Private API
- Rich integrated results view provides click to source navigation
 - ▶ Explanations, examples, and quick fixes for problems
- Allow users to create, enable and disable validation rules
- Allow users to create their own rules based on rule templates
- Produce HTML/PDF reports with violations and metrics
- Complete Java Code Review (200+ rules)

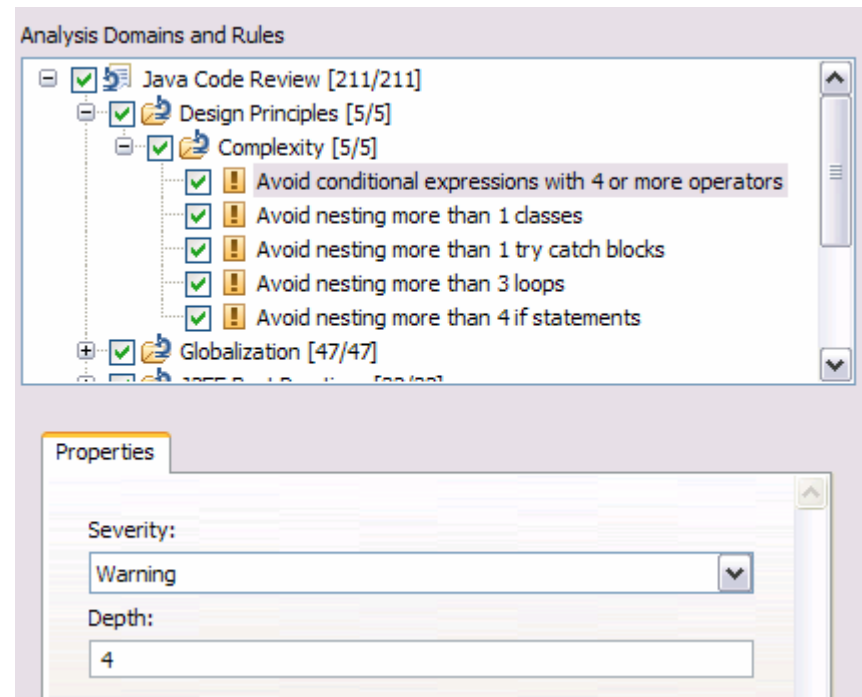
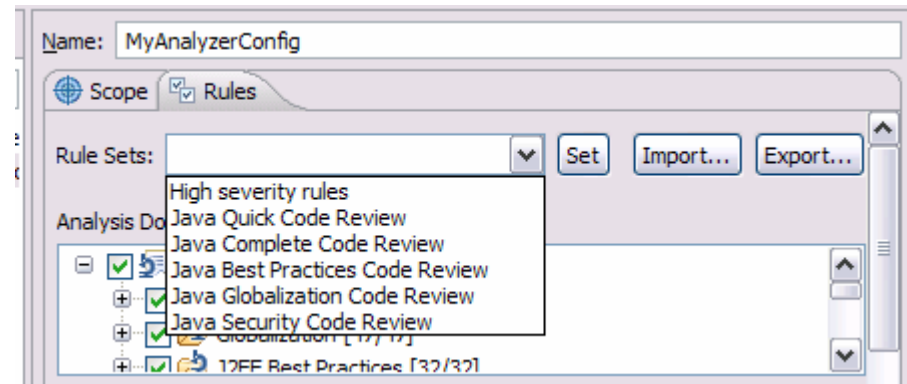
Getting Started with Static Analysis

- Launched through Software Analyzer configuration
- Can create multiple configurations
 - ▶ Scope – workspace, working set or selected projects
 - ▶ Select and configure rules – focus on a particular aspect for analysis



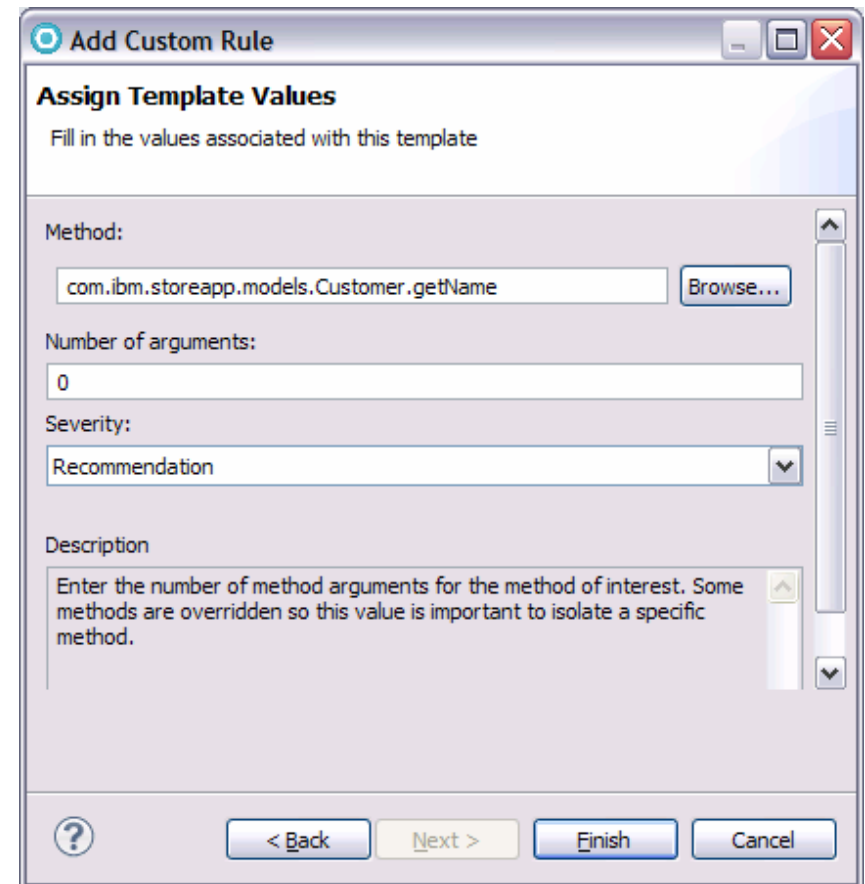
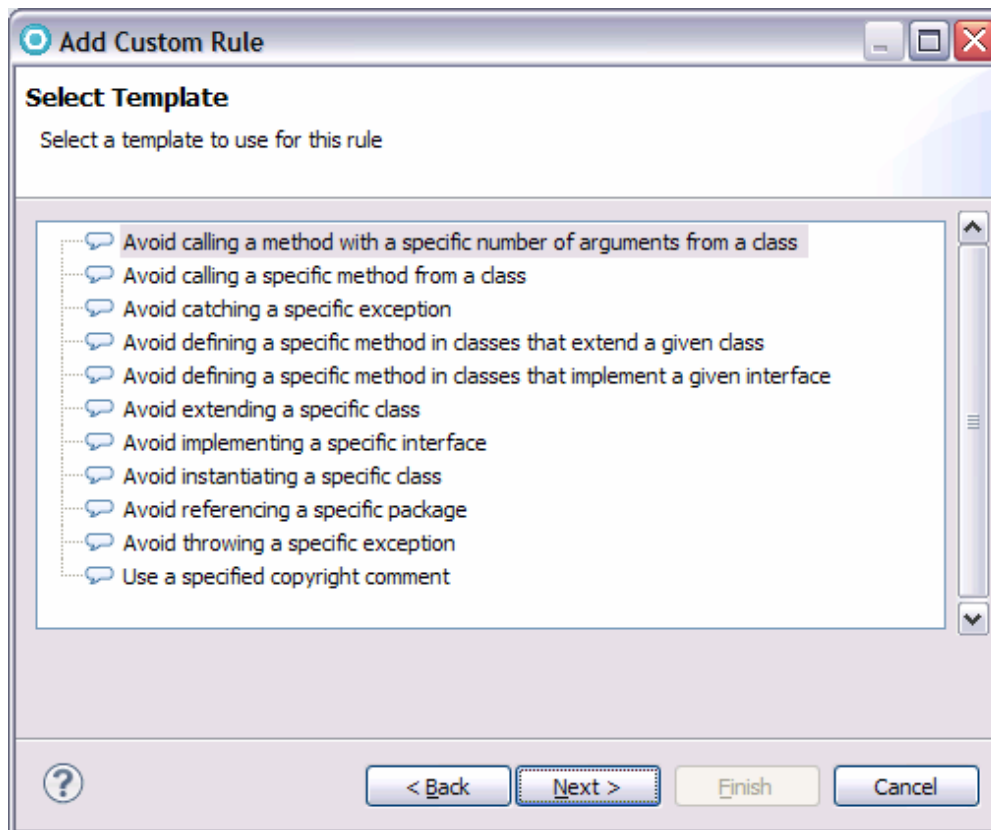
Analysis Configuration – Rules

- Rule sets
 - ▶ Grouping of selected and configured rules that are to be run against the resources in the scope of the analysis
- Custom rule sets
 - ▶ Shared through export and import
- Rule configuration
 - ▶ Select which rules should be used in the current configuration
 - ▶ Configure severity of the violations
 - ▶ Change parameters of rules



Adding Custom Rules

- Static analysis component comes with several templates that can be used to define custom rules through the preferences



Advanced Rule Creation

- Why?
 - ▶ Sometimes the prepackaged or template based rules aren't specific enough
 - ▶ Domain specific constraints or company wide governance enforcement rules

- How?
 - ▶ Software Analyzer component is extensible
 - ▶ Can create new categories/subcategories for your new rules
 - ▶ API framework makes it easier to implement
 - Rules, quickfixes, and even custom reports

- What?
 - ▶ Need to have some background in
 - Eclipse
 - Java
 - Eclipse JDT for Java rules

Example – Rule to check for unused imports

- New category to hold our rule
- New rule definition as part of the new category
- Extend the abstract class `AbstractCodeReviewRule`
 - ▶ Implement `analyze()`
- Optionally implement an associated quickfix by extending `JavaCodeReviewQuickFix`
 - ▶ Implement `fixCodeReviewResult()`

Demo

Example – Rule implementation

```
public class AvoidUnusedImports extends AbstractCodeReviewRule {

    static private final SearchEngine engine = new SearchEngine();

    @Override
    public void analyze(AnalysisHistory history, CodeReviewResource resource) {

        try {
            // Get list of all imports
            List<ASTNode> imports = resource.getTypeNodeList(resource.getResourceCompU

            // Search for which types are used in the class
            UsageSearchRequestor requestor = new UsageSearchRequestor();
            engine.searchDeclarationsOfReferencedTypes(resource.getICompilationUnit(),
            Set<String> usedTypes = requestor.getUsedTypes();

            // compare list of imports with the types used in class
            for (ASTNode node : imports) {
                ImportDeclaration importDecl = (ImportDeclaration)node;
                String name = importDecl.getName().getFullyQualifiedName();
                if (!usedTypes.contains(name)) {
                    resource.generateResultsForASTNode(this, history.getHistoryId(), node);
                }
            }
        } catch (JavaModelException e) {
            Log.severe(e.getLocalizedMessage());
        }
    }
}
```

Example – QuickFix implementation

```
public class UnusedImportsQuickFix extends JavaCodeReviewQuickFix {  
  
    @Override  
    public TextEdit fixCodeReviewResult(ASTNode theNode, IDocument docToChange) {  
        AST ast = theNode.getAST();  
        ASTRewrite rewriter = ASTRewrite.create( ast );  
        rewriter.remove(theNode, null);  
        TextEdit edits = new MultiTextEdit();  
        edits.addChild(rewriter.rewriteAST(docToChange, null));  
        return edits;  
    }  
}
```

Analyzing Results

The screenshot shows an IDE window titled 'Store.java' with the following code:

```
private static void testCustomers() {  
    // creating 10 customers  
    int numCustomers = 10;  
  
    Store store = new Store(1);  
  
    // creating customers  
    for (int i = 0; i < numCustomers ; i++) {  
        store.addCustomer(new Customer("Customer " + i ));  
    }  
  
    // fetching customers
```

Below the code editor is a 'Java Code Review' panel. It displays a tree view of analysis results:

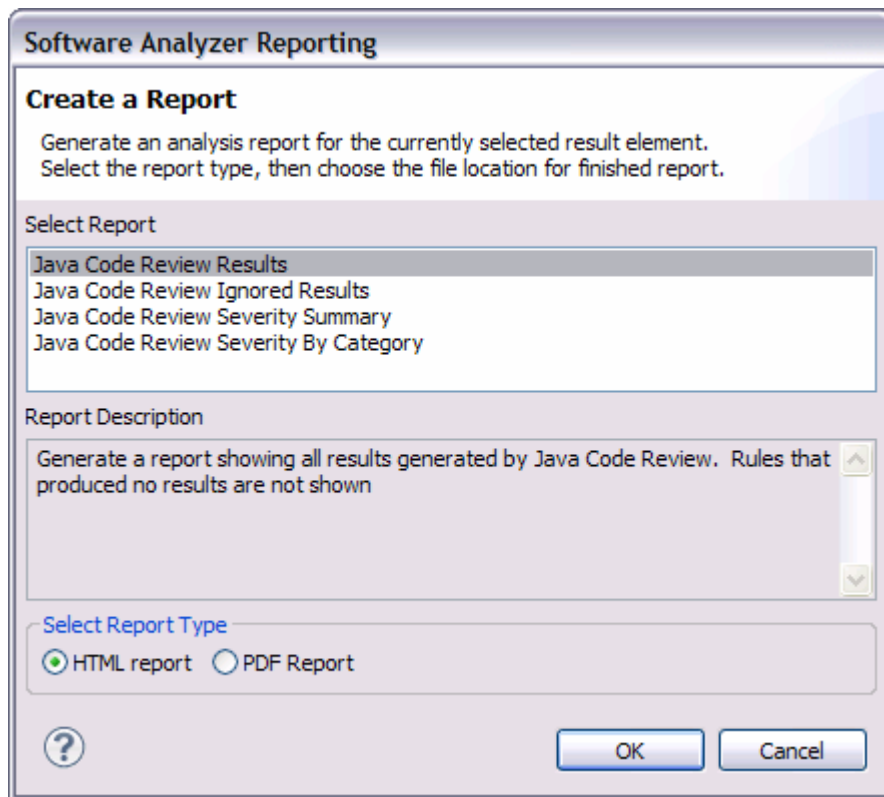
- Globalization [7 results in 81ms]
- J2SE Best Practices [21 results in 183ms]
- Performance [16 results in 54ms]
- Memory [6 results in 12ms]
 - Avoid java.lang.String + operator in loops [6 results in 0ms]
 - Store.java:45 Avoid java.lang.String + operator in loops
 - Store.java:57 Avoid java.lang.String + operator in loops
 - TestCustomer.java:19 Avoid java.lang.String + operator in loops

Three callout boxes provide instructions:

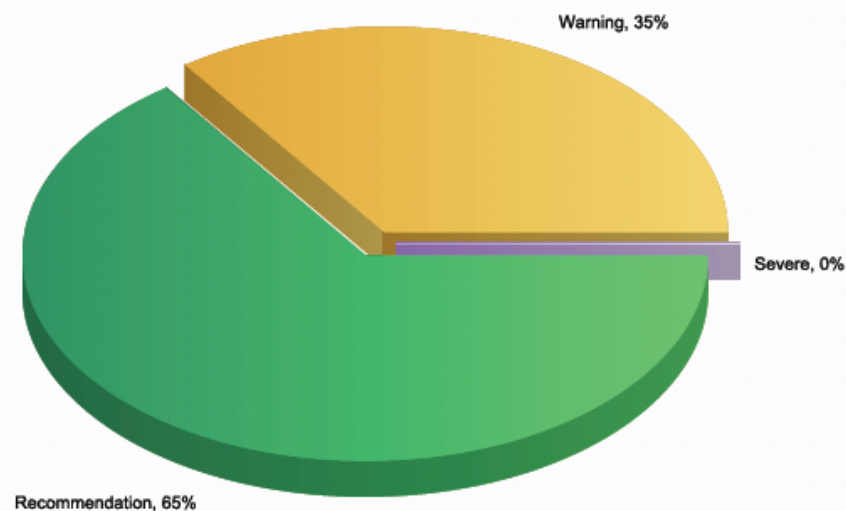
- Export results in XML for other reporting tools
- Generate HTML/PDF reports.
- Double click and view the violation in the source view.

Software Analyzer Reports

- Create HTML/PDF reports of different metrics
 - ▶ Results, Ignored Results, Severity Summary, Severity by Category

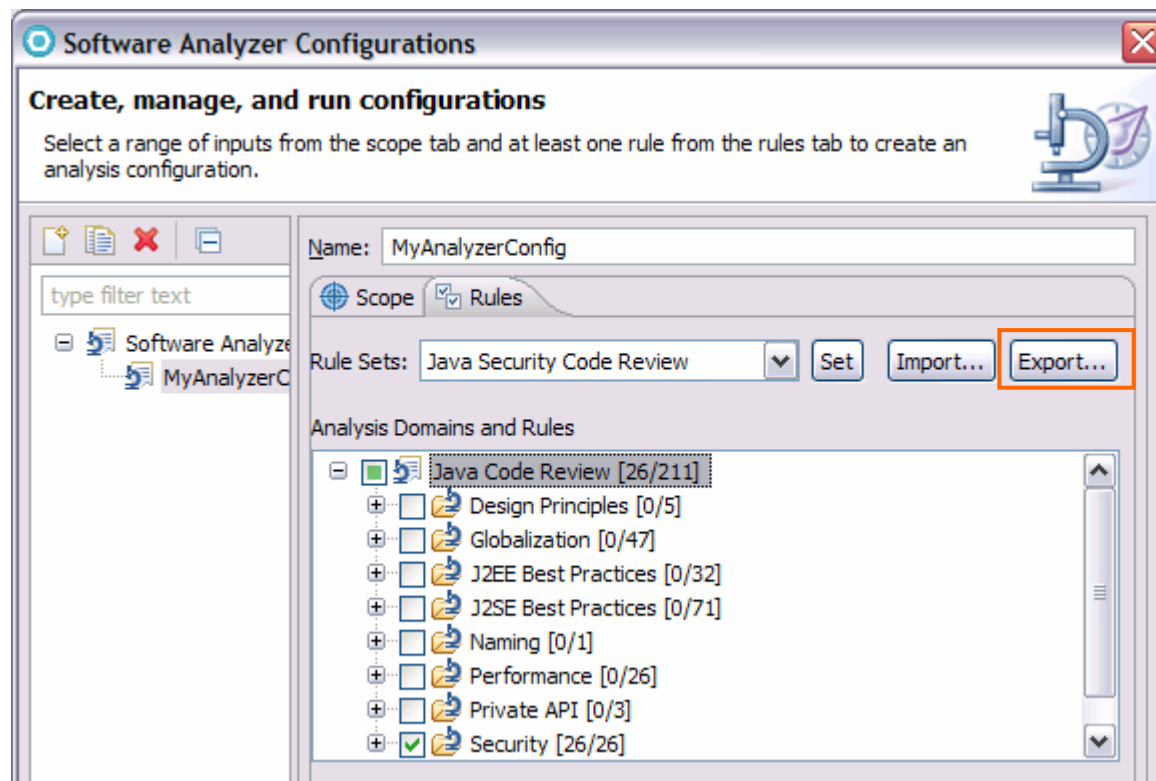


Java Code Review Severity Summary



Integrating Static Analysis Code Reviews – Builds – Step 1

Export your rule set (*.dat) from the Analyzer Configuration



Integrating Static Analysis Code Reviews – Builds – Step 2

Add a step to your build process to invoke the analysis.

```
set
  WORKSPACE="C:\PROGRA~1\IBM\TeamConcertBuild\buildsystem\buildengine\
  eclipse\buildDir"
set ECLIPSE="C:\PROGRA~1\IBM\SDP_1"
%ECLIPSE%\eclipse.exe
  -application com.ibm.xtools.analysis.commandline.AnalyzeApplication
  -rulefile "d:/rules.dat"
  -data %workspace%
  -directory %workspace%
  -exportdirectory %WORKSPACE%\RADcodeReviewXML
```

Best Practices

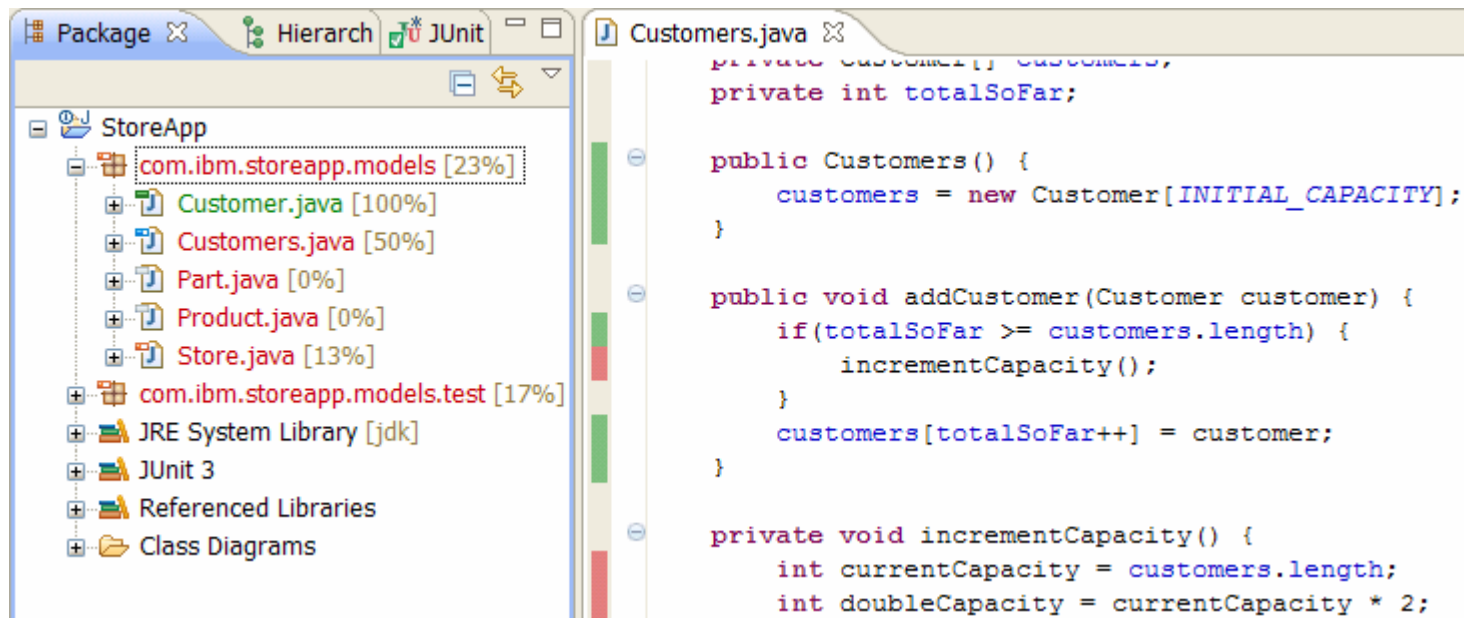
- Run Analysis early and often
 - ▶ As part of developer best practice to run periodically
 - ▶ Cost to fix defects rises the further they get into the application's lifecycle
 - ▶ Logical separation of your application in terms of groups for analysis
 - ▶ Java code for business logic
 - ▶ Java code for connection services
 - ▶ Java code for presentation
- Smaller groups of analysis to avoid overwhelming number of results
 - ▶ Especially when first beginning to use the code review capability
 - ▶ Can increase the scope of analysis as code quality improves
- Create multiple rule sets to apply to different group types
 - ▶ Globalization rules apply more for java code involved in presentation
 - ▶ Different requirements for code complexity enforcement for business logic vs presentation tier

Agenda

- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- Tuning the JVM
- Tuning WebSphere Applications
- Questions

Code Coverage

- Provides detailed information on what code paths have been encountered during program execution
- Powerful tool to help determine xUnit test coverage, potential dead code
- Command line and Ant capability for build integration
 - ▶ JUnit, code coverage data collection and html report generation



Extend team collaboration to IBM middleware developers - deliver better tested software early in the cycle

- Share code coverage information from automated testcase execution
- Improve test coverage and quality based on code coverage results

Code Coverage Report (Apr 26, 2010 11:44:54 AM)

Code Coverage Report

Code Coverage Summary

Code coverage report for 'StoreApp', generated Apr 26, 2010 11:44:54 AM

Component	Coverage	Covered Lines	Total Lines
[StoreApp] com.ibm.storeapp.models.test	16%	11	69
[StoreApp] com.ibm.storeapp.models	24%	21	89
Store.java	13%	7	54
Product.java	0%	0	6
Part.java	0%	0	8
Customers.java	53%	8	15
Customers	53%	8	15
incrementCapacity(): void	0%	0	6
getCustomers(): Customer[]	100%	1	1
addCustomers(): void	100%	3	3
addCustomer(Customer): void	80%	4	5
	100%	6	6

```

public void addCustomer(Customer customer) {
    if(totalSoFar >= customers.length) {
        incrementCapacity();
    }
    customers[totalSoFar++] = customer;
    for (int i = 0; i < 5000; i++) {
        // ...
    }
}
    
```

Lines 18 to 20: Covered

Rational Application Developer

Rational Team Concert Developer

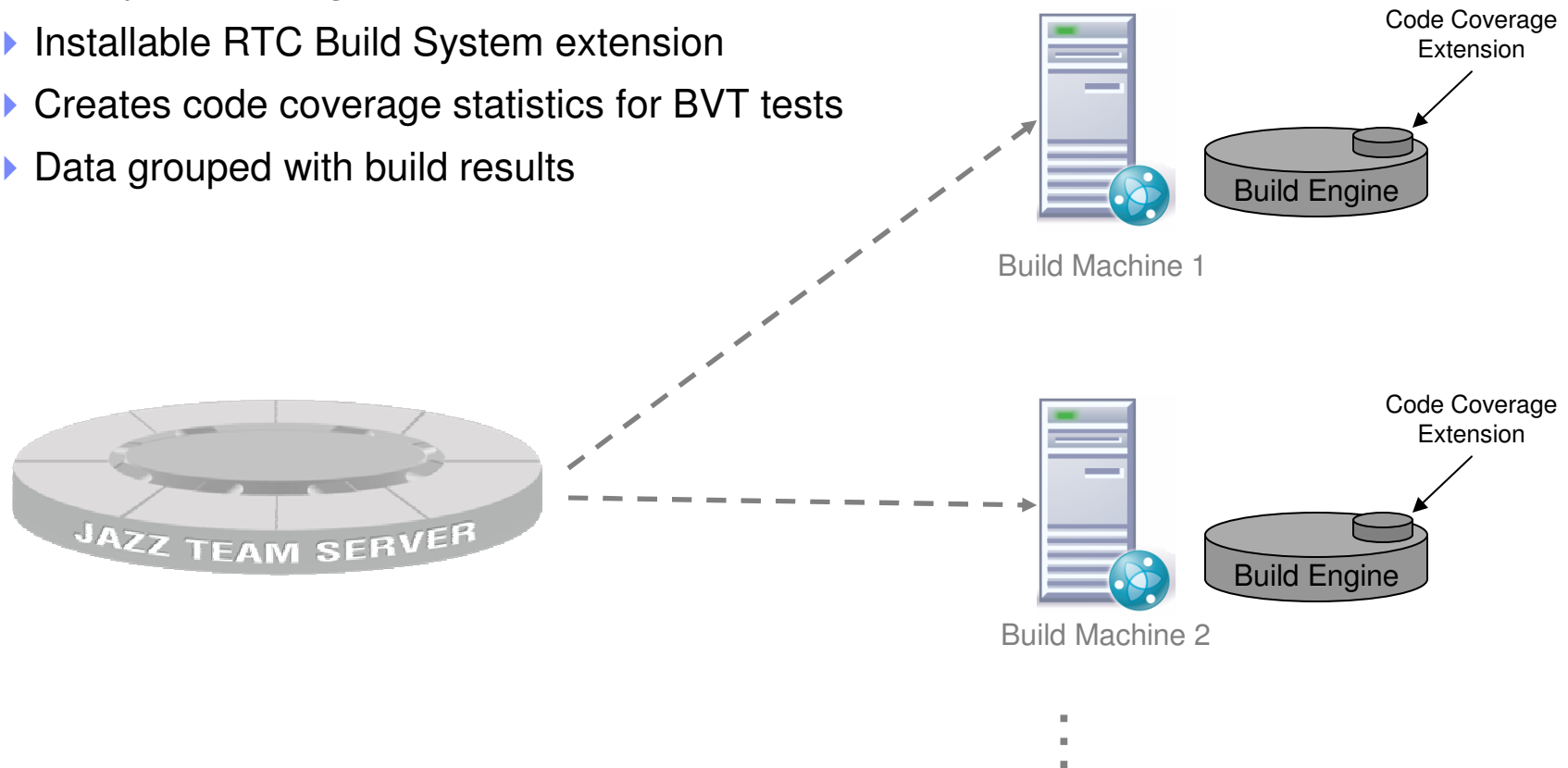
Rational Application Developer

Rational Team Concert Developer



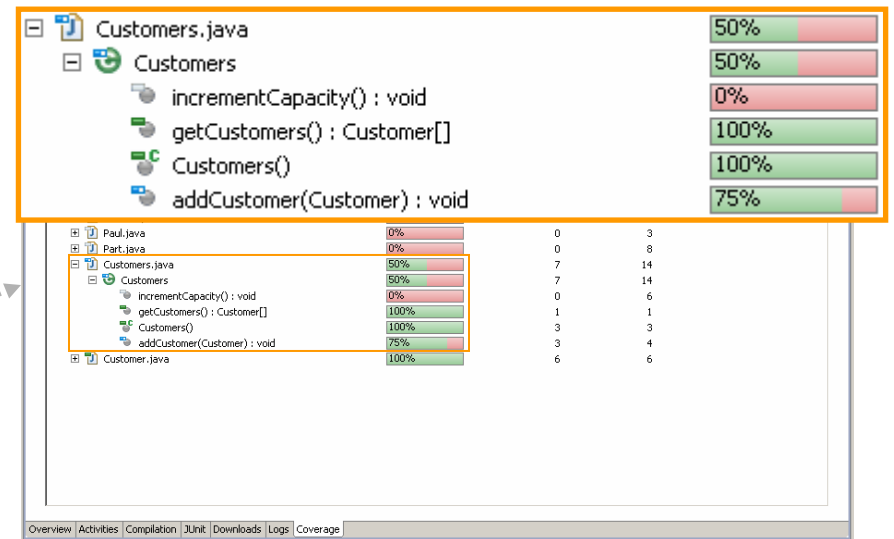
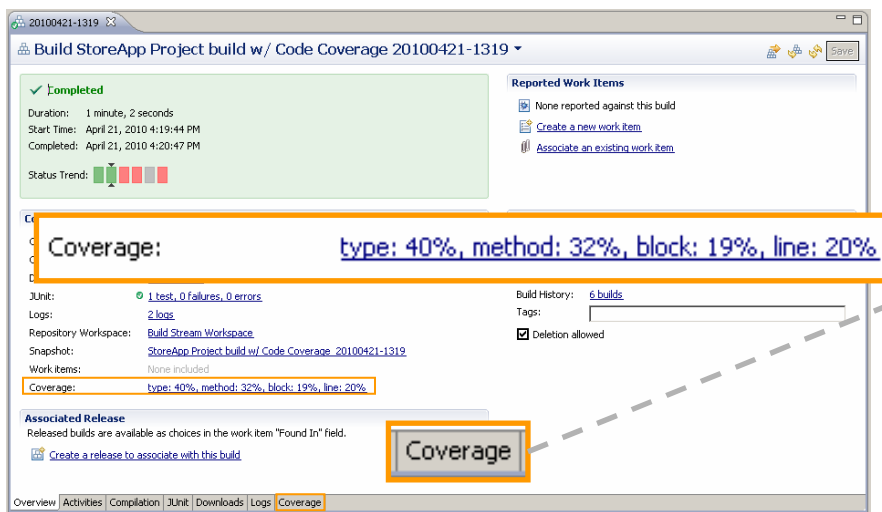
Code Coverage Integration with RTC

- Build system integration
 - ▶ Installable RTC Build System extension
 - ▶ Creates code coverage statistics for BVT tests
 - ▶ Data grouped with build results



Code Coverage Integration with RTC

- Client side
 - ▶ RTC Build details viewer extension – RAD and Web browser
 - Show summary of code coverage statistics
 - Additional Code Coverage tab to show detailed coverage statistics report
 - ▶ Integrated work item search and creation
 - ▶ Import to local workspace for rich viewing in navigator and source views



Code Coverage Comparison

Demo

Code Coverage for WebSphere Applications

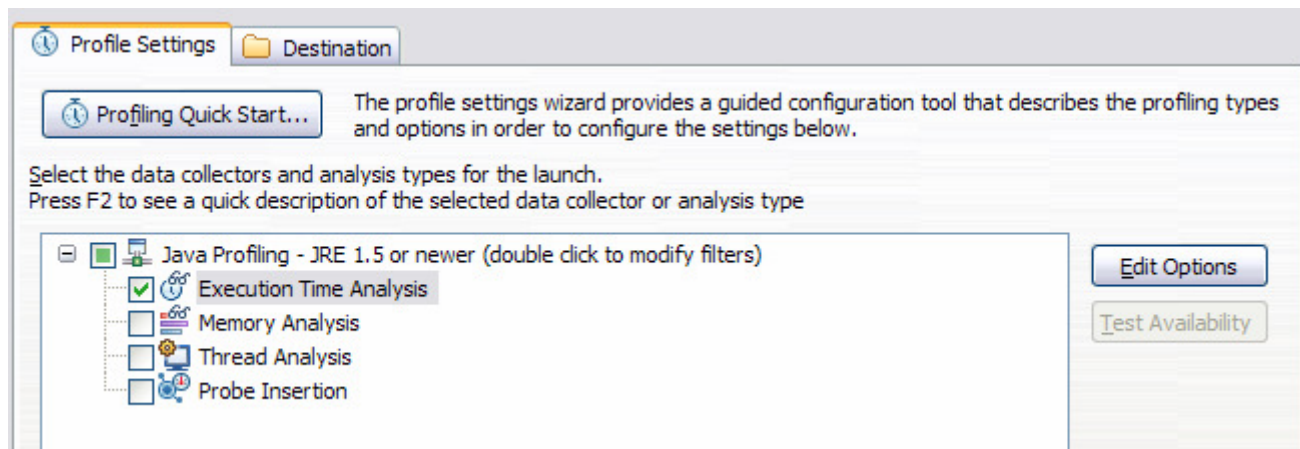
- Dynamically run code coverage on your WebSphere server
 - ▶ Only collects statistics for your code
 - ▶ Easier to configure a server to collect statistics.
 - ▶ Determines if any projects with code coverage enabled have been deployed to server.
 - If so, server configured to collect coverage statistics
- Interactive code coverage collection
 - ▶ Get updates to coverage statistics as you interact with the Web application
 - ▶ Configurable refresh interval
 - Dynamic adjustment heuristic
 - Fixed

Agenda

- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- Tuning the JVM
- Tuning WebSphere Applications
- Questions

Profiling with Rational Application Developer

- Instrumentation based tools for profiling Java applications
 - ▶ Execution Time Analysis
 - ▶ Memory Analysis
 - ▶ Thread Analysis
 - ▶ User written probes





Profiling Quick Start Settings Wizard


- Guided, scenario based approach to resolving performance issues
- Helps configure profiling options to get started faster

 My application is slow

There's a performance bottleneck that is causing transaction throughput to be limited, or is hurting application response time.

 My application takes up too much memory

There's a memory leak causing out of memory errors or inflating application size. The amount of memory used in application sessions is too high causing the number of concurrent sessions to be limited.

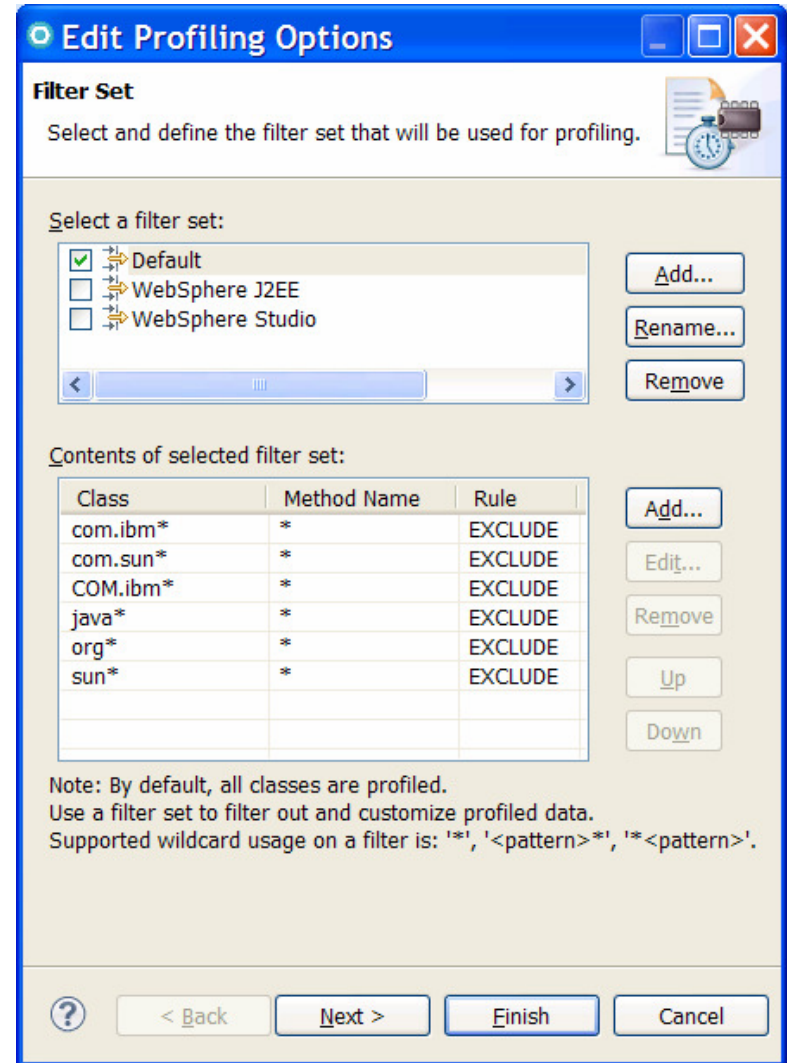
 My application experiences long pauses independent of CPU activity

There are long pauses or hangs in the application that do not seem to be related to processor usage, but rather to thread contention, such as blocked threads or race conditions.

Profiling Options

- Data collection can be expensive
 - ▶ Application under test slows down
 - ▶ Large amount of data for both human and machine

- Filter Sets
 - ▶ Collect data of interest
 - ▶ Focus on your code
 - ▶ Can be exported for sharing



Profiling Metrics

- Determine application behavior
 - ▶ Execution time, number of calls, call tree

Thread name	<Percent Per Thread	Cumulative...	Min Time	Avg Ti...	Max Time	Calls
main[9696]	100.00%	2.381987				
↳ TestSuite(java.lang.Class)	26.88%	0.640376	0.640376	0.640376	0.640376	1
↳ addTestMethod(java.lang.reflect.Met	25.93%	0.617642	0.000046	0.051470	0.616993	12
↳ createTest(java.lang.Class, java.	25.89%	0.616682	0.616682	0.616682	0.616682	1
↳ -clinit-()	9.25%	0.220281	0.220281	0.220281	0.220281	1
↳ ctaticInitializer/java lang Stri	0.77%	0.018303	0.018303	0.018303	0.018303	1

- ▶ Memory allocation

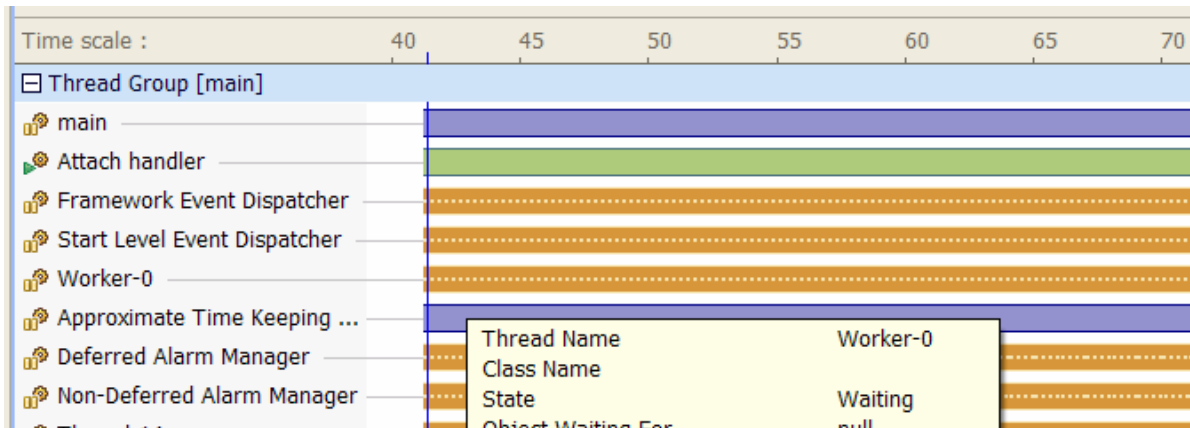
Class Name	Package	Live Inst...	Active Size...	Total Instances	<Total Siz...	Avg. Age
byte[]	(default package)	230	171808	1457	963528	0.08
char[]	(default package)	91	68544	301	114848	0.04
int[]	(default package)	394	21112	731	50664	0.48
long[]	(default package)	3	2232	3	2232	0.67
short[]	(default package)	2	1568	2	1568	0.5
boolean[]	(default package)	9	1456	12	1552	0.75
int[][]	(default package)	0	0	3	120	0

Thread metrics

- Thread Statistics

Thread Name	Class Name	State	<Running Time	Waiting Time	Blocked Time	Block Count	Deadlocked Time	Deadlock Count
P=243359:O=0:WST		Running	00:44:036					
P=243359:O=0:WST		Running	00:44:036					
ORB.thread.pool : 0		Waiting	00:00:075	00:43:961				
ORB.thread.pool : 1		Waiting	00:00:073	00:43:962				
Deferred Alarm Mana		Waiting	00:00:007	00:44:029				
Deferrable Alarm : 0		Waiting	00:00:005	00:44:030				
Non-Deferred Alarm I		Waiting	00:00:003	00:44:033				
Deferrable Alarm : 1		Waiting	00:00:003	00:44:032				
server.startup : 0		Stopped	00:00:002	00:13:033				

- Threads Visualizer – view timeline of threads with state



Custom profiling

- Custom Profiling
 - ▶ Implement your own Java probes to collect runtime information from your application

- Java code fragments
 - ▶ Class variables
 - ▶ Runtime fields
 - thisObject
 - className
 - ...
 - ▶ Target filters

File Name:

Source Folder:

XML Encoding:

Add content to the probe file:

Method Probe
 Callsite Probe
 No Content (Blank Probe File)

This type of probe is inserted anywhere within the body of a method. For method probes, the class or jar files containing the target methods are instrumented by the byte-code instrumentation (BCI) engine.

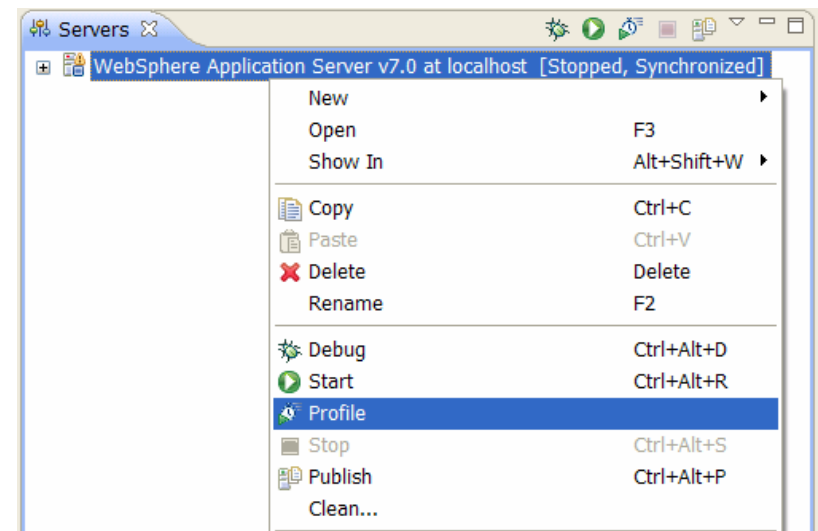
Fragment types:

catch
 entry
 executableUnit
 exit
 staticInitializer

entry fragments execute upon method entry. entry fragments will not execute for methods that were inserted into the class by Probekit.

Application Server Profiling

- Tight profiling integration with WebSphere Application Server
 - ▶ Easily start the server in profile mode with the Servers view
 - ▶ Local and remote servers
- Automatic server configuration
 - ▶ Local and remote servers
 - ▶ No environment variables!
- Agent for remote application servers
 - ▶ IBM Rational Agent Controller available in 7 platforms, 31-bit (zOS), 32/64 bit



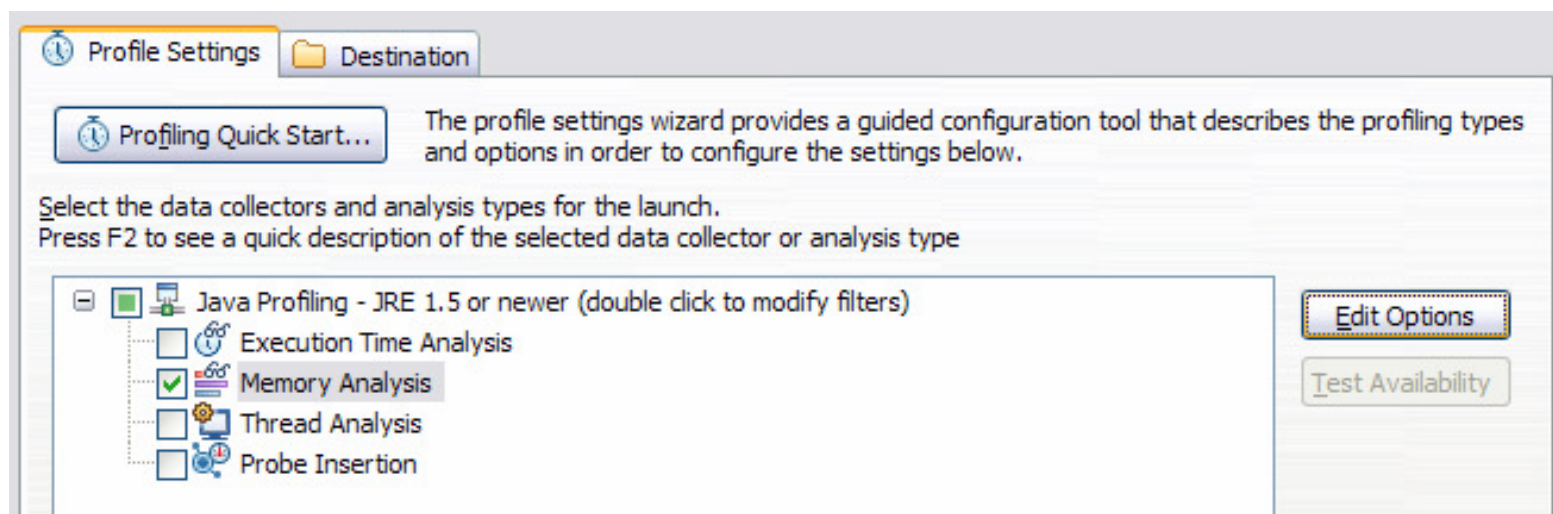
Memory Analysis – Instance data collection

- Instance data collection
 - ▶ Inspect the composition of Objects
 - Live Object values
 - Variable names
 - Instance size.

- Data import/export
 - ▶ Export to an XML file
 - ▶ Import back into workspace for rich viewing

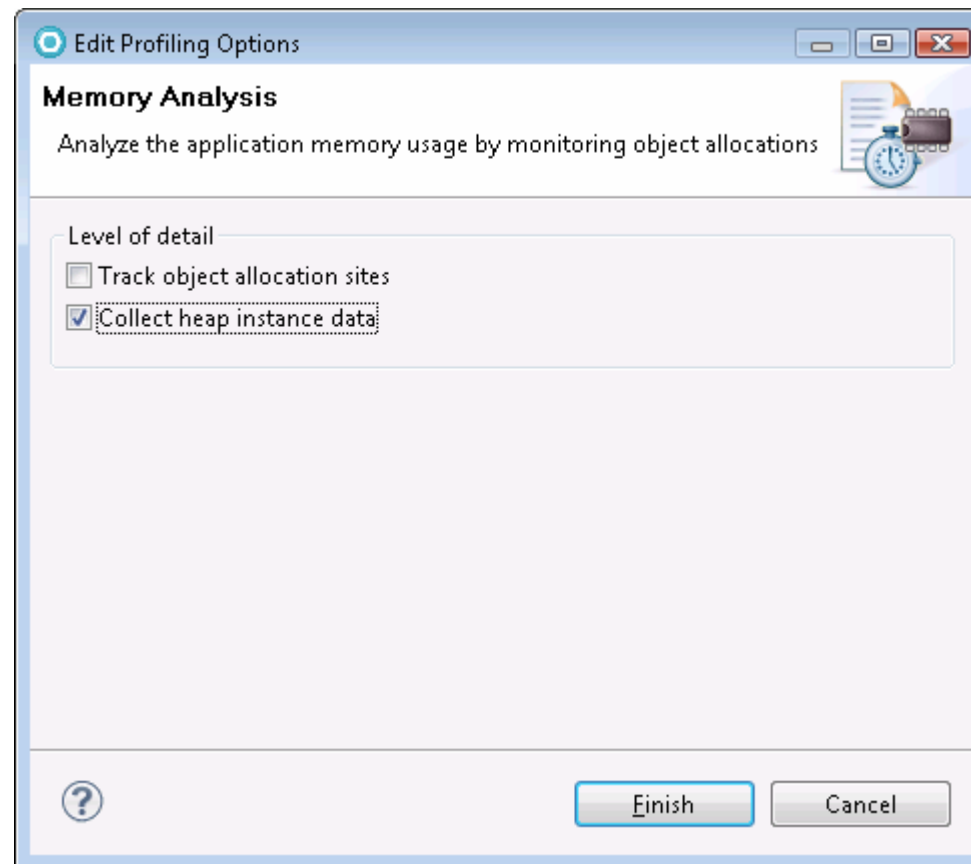
Instance Data Collection

- Choose Edit options for Memory Analysis



Instance Data Collection

- Enable instance data option



Instance Data Collection

- Inspect Collected Object Data
 - ▶ Import/export to XML file for sharing/comparison

Object Allocations x

Memory Analysis - live.instance.test.Main at tptp-jayhawk [PID: 3544]

Memory Statistics

Filter: No filter. Click [here](#) to set filter

>Class Name	Package	Live Instan...	Active Size ...	Total Insta...	Total Size (...)	Avg. Age
▲ Keystore	live.instance.test	2	32	2	32	0
▲ Keystore (id=418)			16		16	
▲ Keystore\$Entry (id=422)			24		24	
String key = Key 1;						
String value = Value 1;						
▷ Keystore (id=430)			16		16	
▲ Keystore\$Entry	live.instance.test	151	3624	151	3624	0
▲ [0 .. 99]						
▲ Keystore\$Entry (id=422)			24		24	
String key = Key 1;						
String value = Value 1;						
▷ Keystore\$Entry (id=487)					24	
▷ Keystore\$Entry (id=488)			24		24	

Instance of live.instance.test.Keystore\$Entry (id=422), size=24

Press F2 to focus

IBM Support Assistant

- Single access point for finding and downloading support tools for IBM products
- Also includes many tools for problem determination, including:
 - ▶ JVM Health Center
 - ▶ Thread and Monitor Dump Analyzer
- Access from RAD launch-pad

Agenda

- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- Tuning the JVM
- Tuning WebSphere Applications
- Questions

Diagnostic Data

- Data from the JVM
 - ▶ Javacore – thread information
 - ▶ Heapdumps – heap details
 - ▶ Verbose GC logging – garbage collection data
- Data from agents connected to the JVM
 - ▶ Health Center
 - ▶ Rational Agent Controller
 - ▶ More detailed information, higher accuracy: thread, heap details, execution flow
 - ▶ More overhead than Javacore or Heapdumps
- Data from the Application Server: PMI

General Strategy

- Try to isolate issues using data that's easily available or obtainable:
 - ▶ Javacores, heapdump files
 - ▶ Use wsadmin to interactively send events to JVM MBean

- If problem not resolved with this level of information
 - ▶ Narrow area of investigation and use filtering combined with more detailed data collection agents

- There is often more than one tool that could be used

- Measure, measure, measure!

Crashes

Problem: The untimely death of an application

- View WebSphere server logs

- Use Thread and Monitor Dump Analyzer
 - ▶ Javacore file from the application
 - ▶ Identifies where the threads last executed

Thread And Monitor Dump Analyzer

- Input - javacore files
- Provides information about threads and monitors at the point the javacore was created

Thread Detail : javacore.20100323.153858.2828.0001			
Name ▲	State	NativeID	Method
Attach handler	Runnable	0x15bc	com/ibm/tools/attach/javaSE/IPC...
Bundle File Closer	Waiting on condit...	0x14c0	java/lang/Object.wait(Native Meth...
Concurrent Mark Hel...	Waiting on condit...	0x15c8	NO JAVA STACK
Finalizer thread	Waiting on condit...	0x140c	NO JAVA STACK
Framework Event Dis...	Waiting on condit...	0x15a0	java/lang/Object.wait(Native Meth...
Gc Slave Thread	Waiting on condit...	0x15c0	NO JAVA STACK
JIT Compilation Thread	Waiting on condit...	0x14a8	NO JAVA STACK
Java indexing	Waiting on condit...	0xbdc	java/lang/Object.wait(Native Meth...
Provisioning Event Di...	Waiting on condit...	0x1788	java/lang/Object.wait(Native Meth...
Signal Dispatcher	Runnable	0x1564	com/ibm/misc/SignalDispatcher....
Start Level Event Dis...	Waiting on condit...	0x153c	java/lang/Object.wait(Native Meth...
State Data Manager	Waiting on condit...	0x9c0	java/lang/Thread.sleep(Native Me...

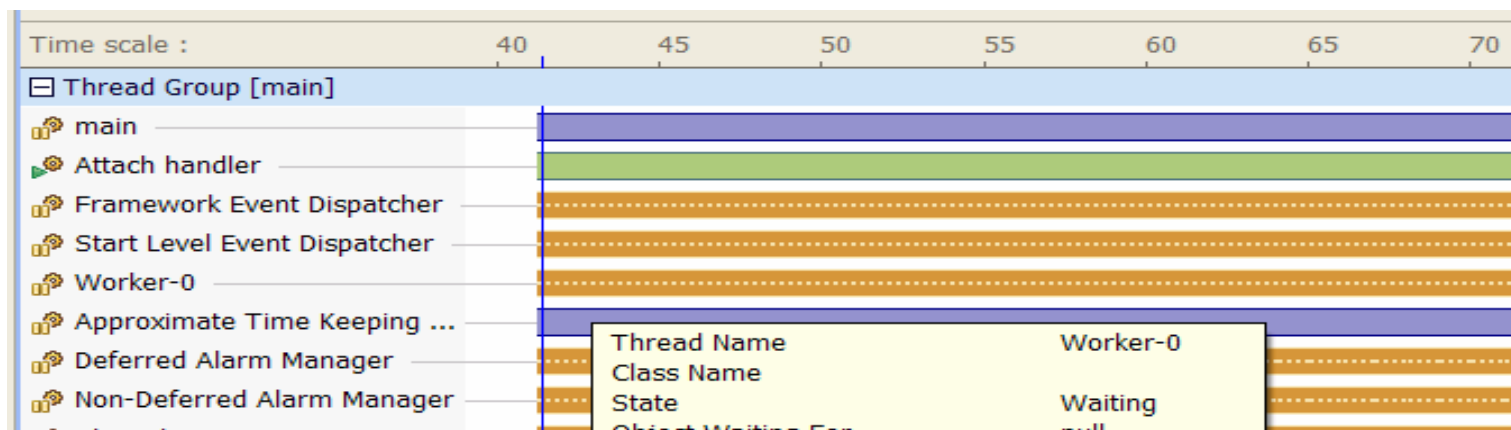
Monitor Detail : javacore1015934.1126598087.txt

Thread Name : Alarm : 3
 State : Waiting on monitor
 Waiting for Heap Lock
 Waiting for Monitor Lock on Heap lock
 Java Stack
 at com.ibm.ejs.util.FastHashtable.elements(FastHashtab le.java:179)
 at com.ibm.ejs.container.StatefulBeanReaper.sweep(Stat efulBeanReaper.java:262)
 at com.ibm.ejs.container.StatefulBeanReaper.alarm(Stat efulBeanReaper.java:236)
 at com.ibm.ejs.util.am._Alarm.run(_Alarm.java(Compiled Code))
 at com.ibm.ws.util.ThreadPool\$Worker.run(ThreadPool.ja va(Compiled Code))

Hangs & Deadlocks

Problem: App is completely unresponsive

- Use Rational Application Developer
 - ▶ Complete hangs - run under debugger and pause after unresponsive point
 - ▶ Hangs/deadlocks
 - Thread Analysis for more detailed data
 - Thread statistics and Threads Visualization contention



- Use Thread and Monitor Dump Analyzer

Memory Leaks / Excessive Memory Use

Problem:

- ▶ Slowdown due to increased garbage collection
- ▶ Reduced number of concurrent sessions
- Use Rational Application Developer
 - ▶ Use Memory Analysis
 - ▶ Identify offending object types
 - ▶ Restart analysis, tracking object allocation sites if understanding where they're being allocated is needed

Class Name	Package	Live Inst...	Active Size...	Total Instances	<Total Siz...	Avg. Age
byte[]	(default package)	230	171808	1457	963528	0.08
char[]	(default package)	91	68544	301	114848	0.04
int[]	(default package)	394	21112	731	50664	0.48
long[]	(default package)	3	2232	3	2232	0.67
short[]	(default package)	2	1568	2	1568	0.5
boolean[]	(default package)	9	1456	12	1552	0.75
int[][]	(default package)	0	0	3	120	0

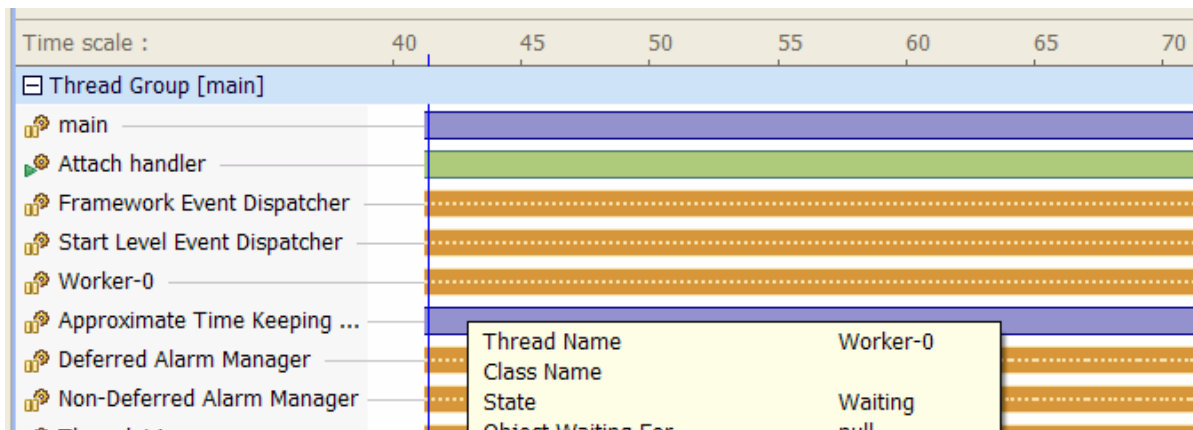
- Consider reducing session timeout to reduce longevity of session objects

Demo

Bottlenecks – Overly Aggressive Locking

Problem: slow responsiveness and throughput of web apps

- Potential cause: multithreaded code blocks protect more than they need
- Use Rational Application Developer
 - ▶ Thread Analysis for more detailed data
 - ▶ Thread statistics and Threads Visualization to view overlapping contention areas over time
 - ▶ Investigate long lock areas



Demo

Bottlenecks – Inefficient Code Paths

Problem: Slow responsiveness and throughput of web apps

- Potential cause: redundant calls or long running code
- Use Rational Application Developer
 - ▶ Execution time analysis for detailed metrics
 - ▶ Look for long cumulative times and base times

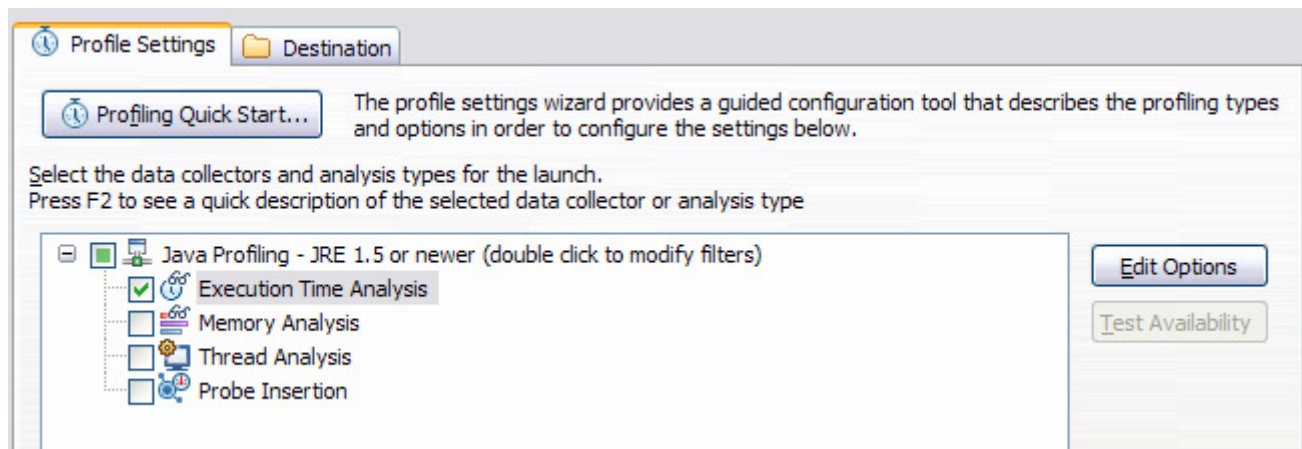
Thread name	<Percent Per Thread	Cumulative...	Min Time	Avg Ti...	Max Time	Calls
main[9696]	100.00%	2.381987				
↳ TestSuite(java.lang.Class)	26.88%	0.640376	0.640376	0.640376	0.640376	1
↳ addTestMethod(java.lang.reflect.Met	25.93%	0.617642	0.000046	0.051470	0.616993	12
↳ createTest(java.lang.Class, java.	25.89%	0.616682	0.616682	0.616682	0.616682	1
↳ -clinit(-)	9.25%	0.220281	0.220281	0.220281	0.220281	1
↳ ctaticInitializer/java lang Stri	0.77%	0.018303	0.018303	0.018303	0.018303	1

Demo

Custom performance profiling – Collecting data you want

Problem: Create customized profiling data from your application. Key metric or aspect of your application you want to easily capture and repeat.

- Use Rational Application Developer
 - ▶ Write your own java code fragment
 - ▶ Insert your code through the Probe Insertion profiling



Demo

Agenda

- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- **Tuning the JVM**
- Tuning WebSphere Applications
- Questions

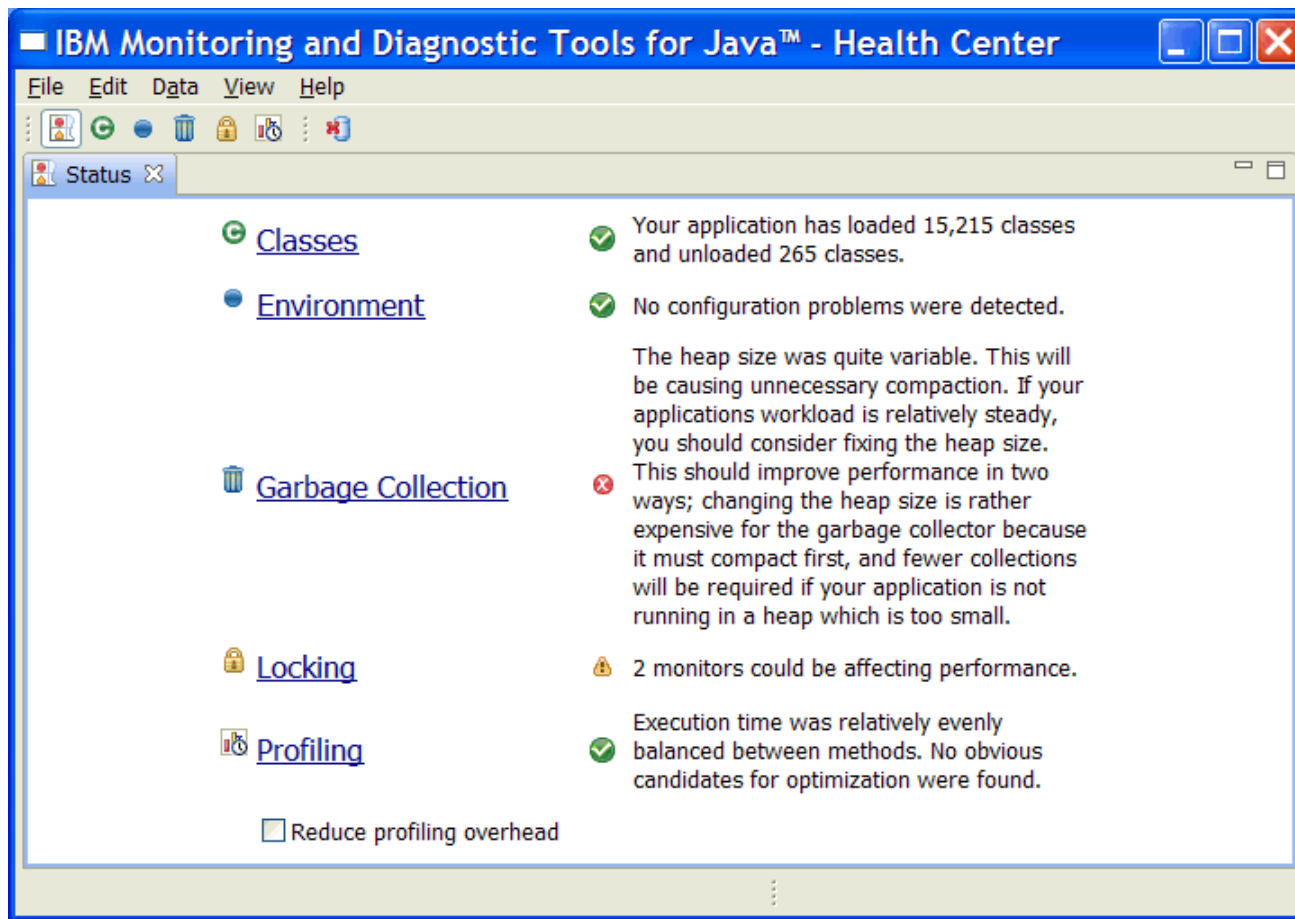
JVM Tuning – Garbage Collection Strategy and Heap size

- Potential Issues:
 - ▶ Pause times – server unable to process requests due to GC pause times
 - ▶ Factors affected GC frequency and size/number of session objects supported
 - Garbage collection policies
 - Heap size parameters

 - Use Health Center – live recommendations for GC strategy and heap size
 - ▶ Execute a reasonable portion of your application first
- Demo**
- Use Garbage Collection and Memory Visualizer
 - ▶ Enable verbose GC output
 - ▶ Provides more detailed tuning recommendations

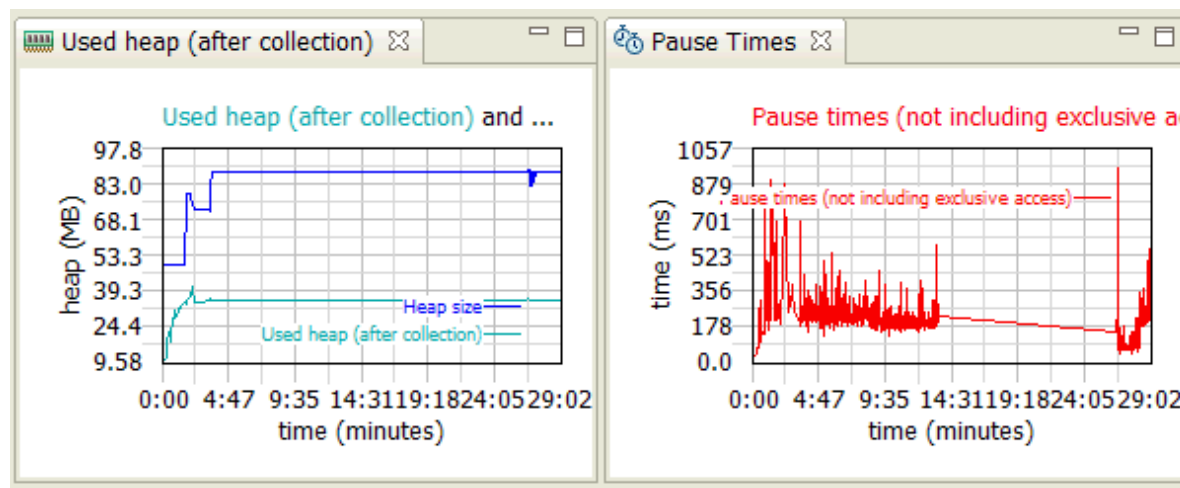
Health Center

- From IBM Support Assistant (ISA) via the RAD launchpad
- Whole application health
 - ▶ lightweight profiling, heap size graphs, classes loaded, GC information



Health Center

- Requires application be started with the Health Center agent
- Configurable filter
- Tuning suggestions based on live analysis
 - ▶ GC
 - Call stacks
 - Lock contention



Agenda

- Governance of Code
 - Creating your own rules - customizing rule templates
 - Advanced Rule creation
 - Development process integration
 - Best Practices
- Code coverage and Unit test optimization
 - Overview
 - Development process integration
- Profiling
- Problem scenarios: What should I use?
 - ▶ Crashes, hangs
 - ▶ Memory leaks
 - ▶ Execution bottlenecks
- Tuning the JVM
- Tuning WebSphere Applications
- Questions

WebSphere Application Server Tuning

- Server configuration can affect the performance of your application

- Analysis Tools
 - ▶ Enable logging of key PMI data via admin console
 - Open tivoli view – showing various tabs that can be monitored
 - Start logging activity for scenarios you want to optimize for
 - Use the performance advisor to produce tuning recommendations
 - Retry the scenario with configuration changes

Demo

Tivoli Performance Viewer

- Embedded in the WAS Administrative Console
- Allows monitoring of WAS overall health using the Tivoli Performance Viewer
 - ▶ Look at PMI (Performance Monitoring Infrastructure) data
 - System resources
 - WebSphere pools, queues
 - Application data (e.g. servlet response times)

[Tivoli Performance Viewer](#) > server1

Use this page to view and refresh performance data for the selected server, change user and log settings, and view summary reports and information on specific performance modules.

Refresh View Module(s)

- server1
 - Advisor
 - Settings
 - Summary Reports
 - Servlets
 - EJBs**
 - EJB Methods
 - Connection Pool
 - Thread Pool
 - Performance Modules

EJBs Summary Report

[More information about this page](#)

Stop Logging

Name	Application	Method Calls	Avg Resp Tim
Catalog	PlantsByWebSphere_EAR#PlantsByWebSphereAjax_EJB.jar	34	127.324
Inventory	PlantsByWebSphere_EAR#PlantsByWebSphereAjax_EJB.jar	410	9.229
ShoppingCart	PlantsByWebSphere_EAR#PlantsByWebSphereAjax_EJB.jar	5	9.4

Tivoli Performance Advisor

- Get tuning recommendations from the performance advisor on various customizable server configurations

Severity	Message
Alert	TUNE0221W: Data for memory session ...
Config	TUNE5003W: The JVM maximum heap siz...
Config	TUNE5012W: The size of the minimum ...
Config	TUNE5042W: Enable servlet caching f...

Message

TUNE5003W: The JVM maximum heap size is unusually small. Typically, the maximum heap size is greater than or equal to 256.

Severity

Config

Description

If the maximum heap size is too small, the Java Virtual Machine (JVM) does not have enough room to manage the heap efficiently. Performance degrades and the application can fail.

User Action

To change the JVM heap size, open the administrative console and click Application Servers > server > Java and Process Management > Process Definition > Java Virtual Machine. See the information center for more information on tuning JVM.

Detail

Currently, the initial heap size is 49 MB and the maximum heap size is 255 MB.

Conclusions

- Finding problems early in development saves money
- Code Coverage from Rational Application Developer running on a server
 - ▶ Verify what get's touched incrementally
 - ▶ Combine with unit tests or BVT to automate and track
- Many tools available to aid in problem determination:
 - ▶ Understanding what the various diagnostic data contains helps guide problem determination strategy for particular problems
 - Different tools operate on different input data
 - ▶ Detailed and accurate data collection incur overhead
 - Identify focus areas to filter data collection
 - ▶ WebSphere Application Server contains embedded performance monitoring
 - Useful Java EE centric performance data
 - Advisor provides server tuning suggests for you applications

Additional RAD information

- RAD for WebSphere
<http://www-01.ibm.com/software/awdtools/developer/application/index.html>
- RAD Wiki
<http://www.ibm.com/developerworks/wikis/display/rad/Home>
- RAD v8 Redbook (this and others available via the RAD Wiki)
<http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg247835.html>
- What's New in RAD 8.0.3
<http://www-01.ibm.com/support/docview.wss?uid=swg27018924#803>
- WebSphere Application Migration Toolkit
http://www.ibm.com/developerworks/websphere/downloads/migration_toolkit.html

QUESTIONS



www.ibm/software/rational



www.ibm/software/rational

© Copyright IBM Corporation 2011. 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.

Export Collected Object Data

- The data, such as in the example below, can be exported and used for comparison

```
<?xml version="1.0" encoding="UTF-8" ?>
<heap-instance-data version="1">
  ...
  <complex-obj tid="418" parent-tid="429" class="Keystore" package="live.instance.test" size="16">
    <object tid="423" parent-tid="418" class="Entry:entries" package="" size="24"/>
  </complex-obj>
  <complex-obj tid="422" parent-tid="423" class="Entry:next" package="" size="24">
    <primitive type="String" name="value">Value 1</primitive>
    <primitive type="String" name="key">Key 1</primitive>
  </complex-obj>
  <complex-obj tid="423" parent-tid="418" class="Entry:entries" package="" size="24">
    <object tid="422" parent-tid="423" class="Entry:next" package="" size="24"/>
    <primitive type="String" name="value">Value 2</primitive>
    <primitive type="String" name="key">Key 2</primitive>
  </complex-obj>
  ...
</heap-instance-data>
```



www.ibm/software/rational

