

# Rational® Application Developer Performance Tips

## Introduction

This article contains a series of hints and tips that you can use to improve the performance of the Rational® Application Developer. This article updates a [previous article](#) that was published in March of 2006. The screen captures used in this article were taken with Rational Application Developer version 7.5.4.

Many of the tips in this paper apply to other members in our family of products, including Rational Software.

## *Summary*

The tables below summarize the performance tips outlined in this article including when to consider using a tip and what performance improvements are made by using the tip.

- **Consider** - The relative importance of the tip.
- **Improvement Type** - Indicates if the tip improves the speed or lowers memory usage. Note, if your system is thrashing, lowering memory usage will also greatly improve the speed.
- **Preference** - Indicates if this item is changed in the **Windows > Preferences** dialog.

Table 1: Summary of performance tips.

Tip	Consider	Area Improved			Improvement Type		Preference
		Startup	Build	General	Time	Memory	
<a href="#">Auto Build &amp; Refresh Workspace</a>	Sometimes						
<a href="#">Binary Projects</a>	Always						
<a href="#">Capabilities</a>	Rarely						
<a href="#">Closing Projects</a>	Always						
<a href="#">Defragmenting</a>	Always						
<a href="#">Do not Install Unneeded Features</a>	Always						
<a href="#">Early Startup</a>	Rarely						
<a href="#">Fresh Workspaces</a>	Sometimes						
<a href="#">JVM Tuning</a>	Sometimes						
<a href="#">JVM Tuning - Shared Classes</a>	Rarely						
<a href="#">Label Decorators</a>	Rarely						

Table 2: Summary of performance tips continued.

Tip	Consider	Area Improved			Improvement Type		Preference
		Startup	Build	General	Time	Memory	
<a href="#">Links Builder</a>	Rarely						
<a href="#">Publishing and Annotations</a>	Always						
<a href="#">Quick Diff</a>	Rarely						
<a href="#">Remote Test Server</a>	Always						
<a href="#">Restarting Projects (instead of the Server)</a>	Always						
<a href="#">Running Server in Debug mode</a>	Always						
<a href="#">Server Startup Options</a>	Always						
<a href="#">Server Startup Options (Admin Console)</a>	Always						
<a href="#">Restarting Rational Application Developer</a>	Always						
<a href="#">Task Manager</a>	Sometimes						
<a href="#">Validation</a>	Always						
<a href="#">Updates</a>	Always						

## Preferences

Rational Application Developer has a large number of preferences (**Windows > Preferences**) that allow you a great deal of control over how Rational Application Developer operates. This section discusses how some of these preferences affect performance.

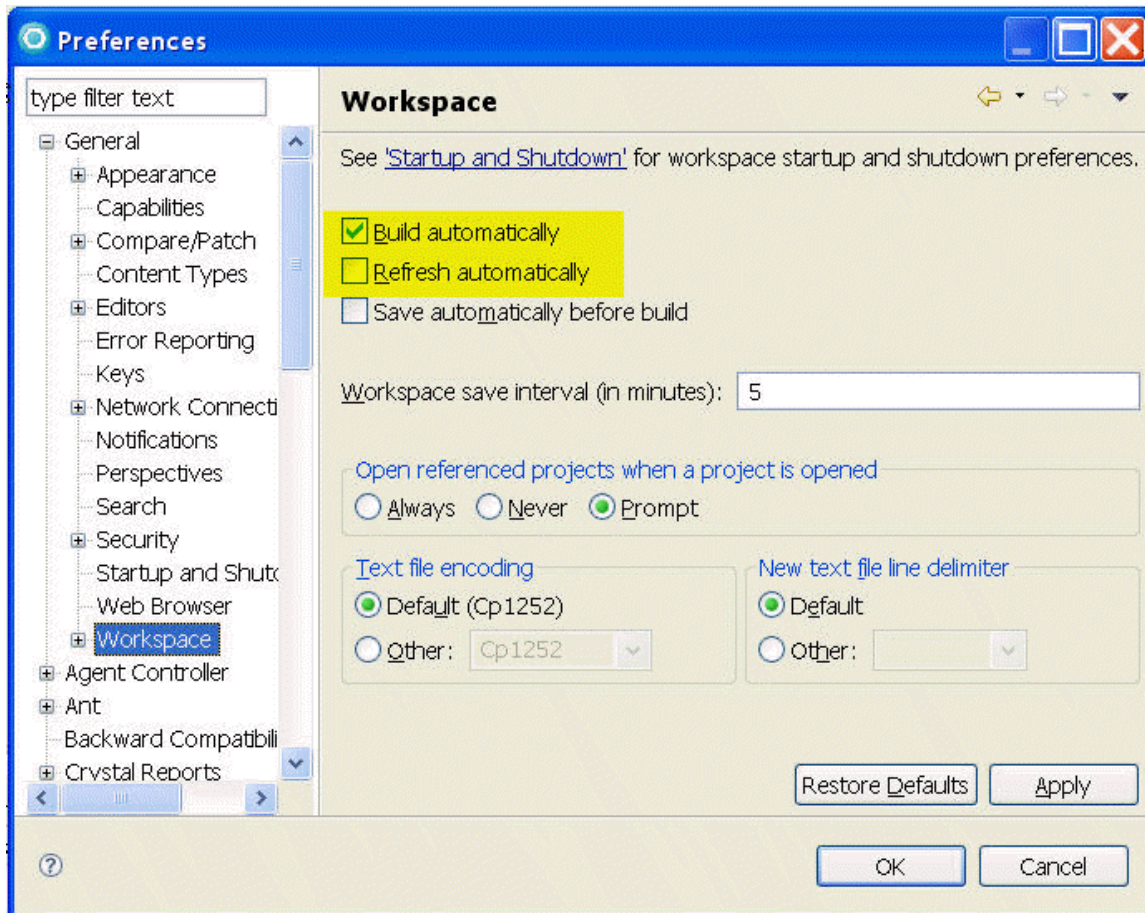
## Build and Refresh workspace automatically

When auto build is turned on, it means that every time you save a file, an incremental build occurs. This includes building (or compiling) the file as well as building any dependent files. The auto build function also runs validation. Usually this is quick; however, if you find that it takes a long time to save your files, you can disable the auto build function. Turning auto build off will improve the performance of your save operations but you will experience slower builds and error information will not be up-to-date. There are pros and cons to disabling auto build, the choice depends on your personal preference. Personally, I do my development with auto build turned on. You may want to experiment with this option to see which setting suits your personal work style.

If you do turn auto build off, you will need to do a **Build All** (Ctrl+B) periodically. This will build all of the changed items and their dependents. This is still an incremental build, where the "increment" is bigger and you can select when it is performed.

Auto refresh checks to see if outside processes have changed any of the files in the workspace. I recommend that you leave auto refresh turned off.

Figure 1. Build automatically is selected by default and Refresh automatically is cleared by default.

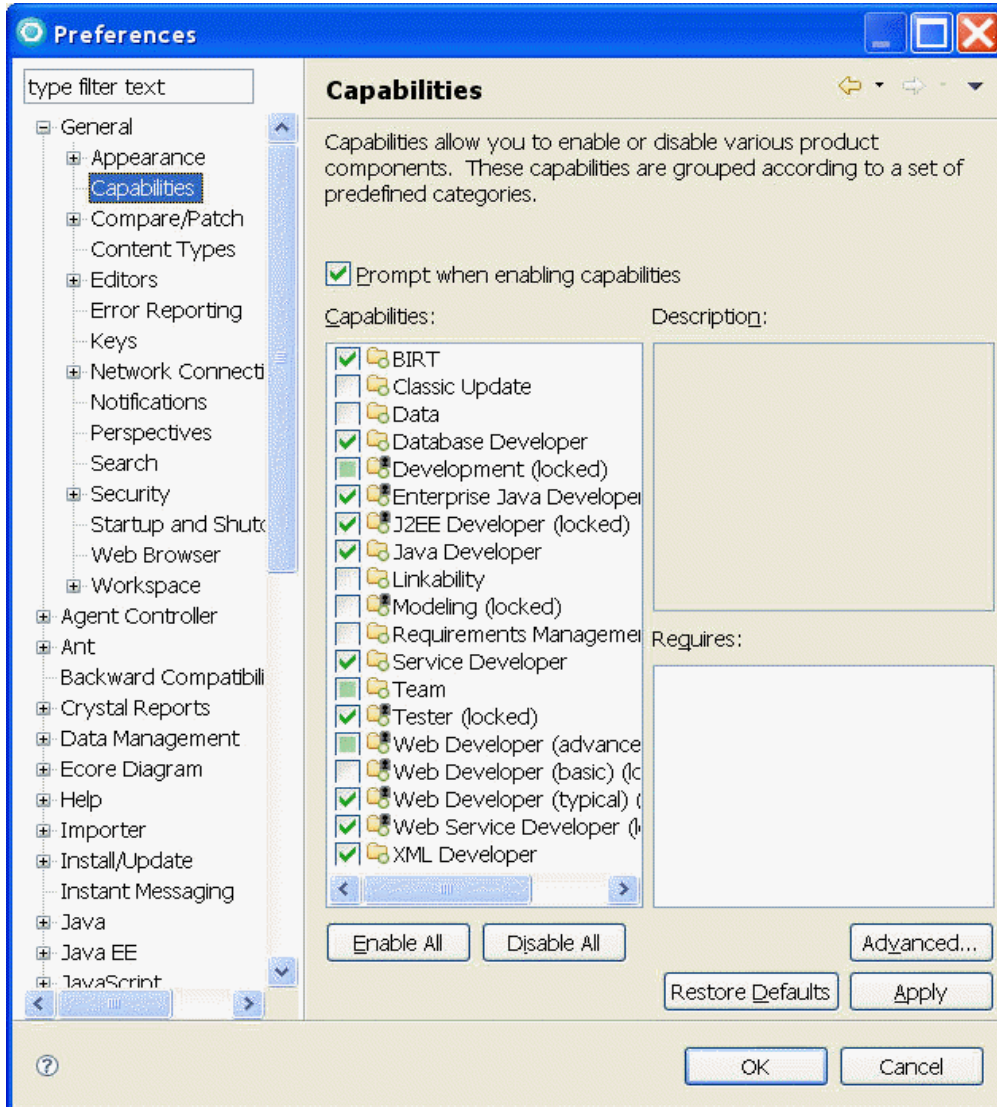


## Capabilities

You can control which capabilities are enabled. By turning off some of the capabilities, you will remove certain user interface items. This in turn may prevent you from accidentally invoking certain functions. Since Rational Application Developer uses lazy loading, any time you invoke a new function, it takes more memory to hold the classes that implement the new function.

If you are not using those functions, turning off the capability will not make any difference to performance, other than to prevent you from accidentally invoking a function.

Figure 2. Modifying Capabilities

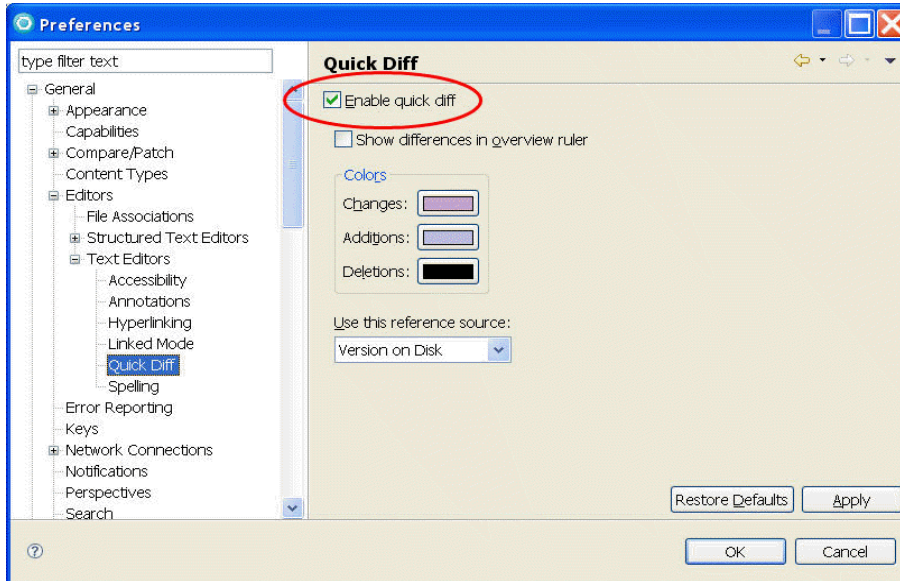


## Quick Diff

Quick diff displays a line in the margin to indicate what has changed. Disabling quick diff will improve performance slightly while editing files.



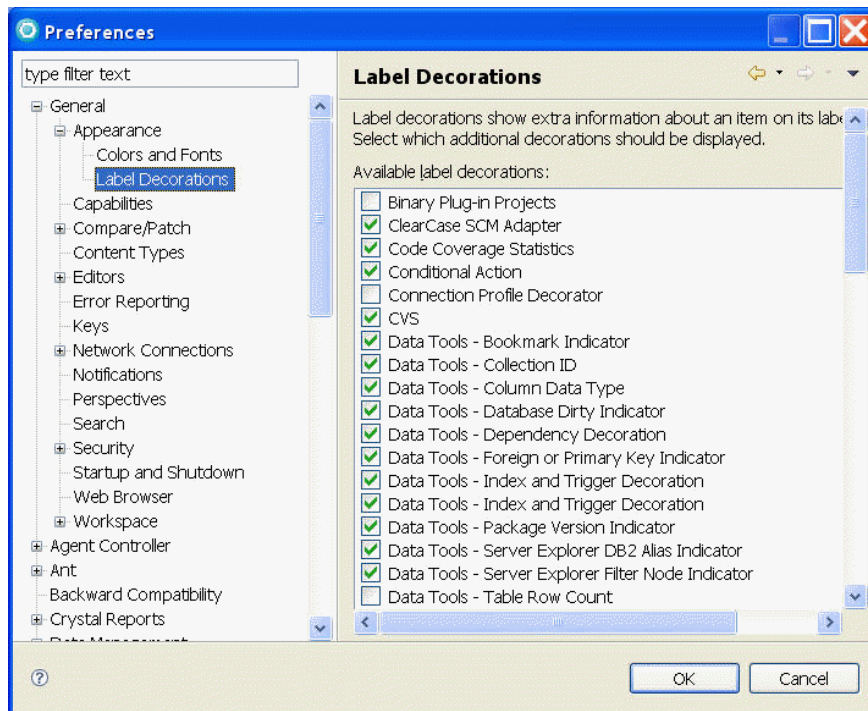
Figure 3. Quick Diff Options



## Label Decorators

Disabling all or some of the label decorations will improve performance slightly.

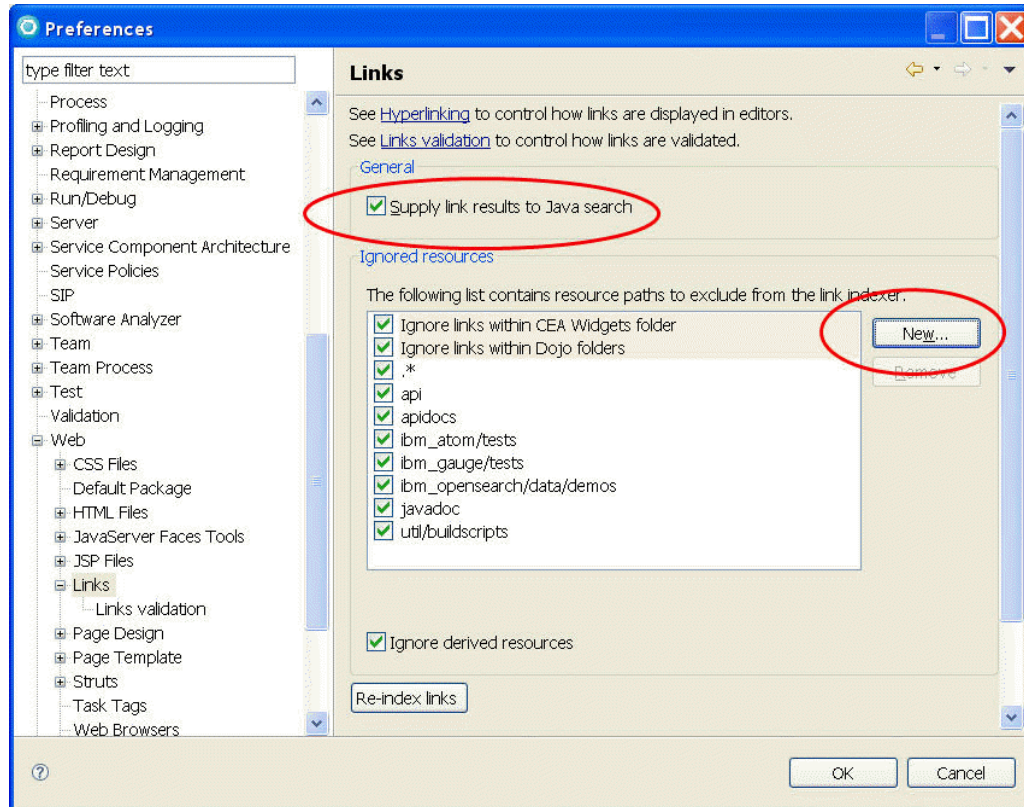
Figure 4. Label Decorations



## Links Builder

The links builder keeps track of hyperlinks. If you have a lot of HTML or JSP files in your workspace, there can be a lot of links to monitor. Disabling links builder will improve performance; however, in general I do not recommend disabling links builder, since many other functions (for example refactoring) depend on having accurate link information.

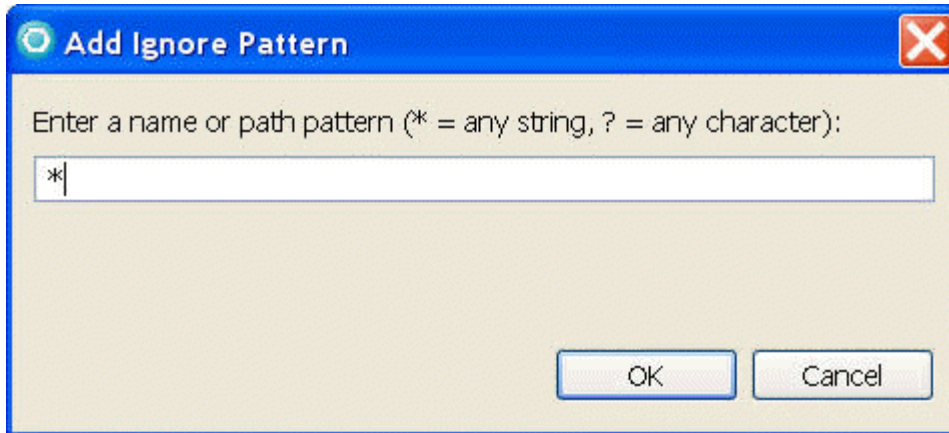
Figure 5. Links Builder



To disable links builder, add a new ignored resource, with the pattern `*`.



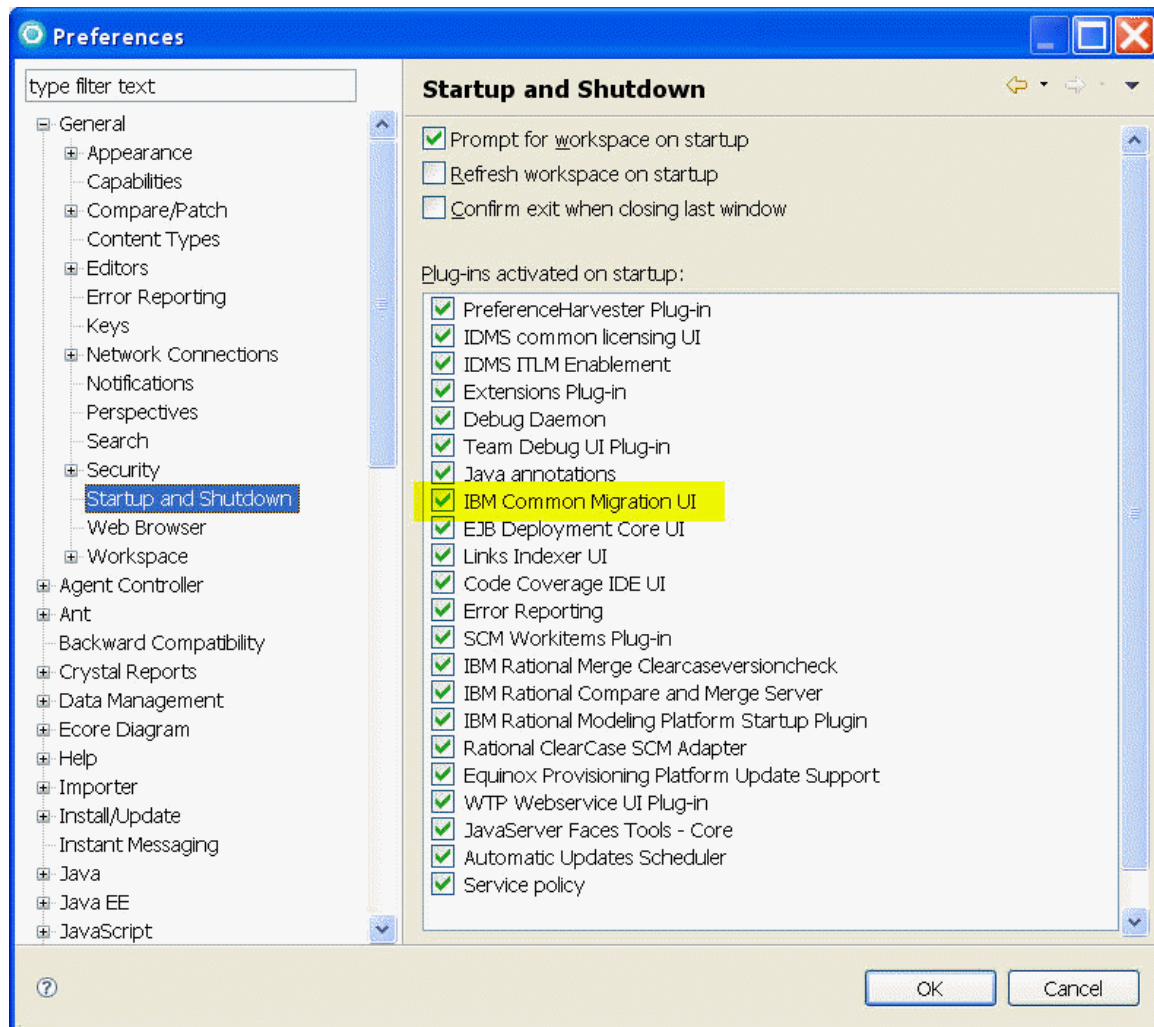
Figure 6. Adding a New Ignored Resource



## Early Startup

These preferences control which plug-ins start automatically as part of startup. I would not recommend disabling any of the plug-ins. However, if you have a large workspace and you have migrated all of your resources, and you would like to improve your startup times, you can disable **IBM® Common Migration UI**. This plug-in scans most of your files as it tests for resources that need to be migrated.

Figure 7. Disabling Startup Plug-ins



## Validation

This setting is one of the big hitters. Running all of the validators all of the time is a reasonably expensive operation. If you are finding that your builds are taking too long, you could turn off all or some of the validators.

When a validator is turned off, it means that you will not have the benefit of any error, warning, or informational messages that that validator might have produced. A quick way to turn off all the validators is to select **Suspend all validators**.

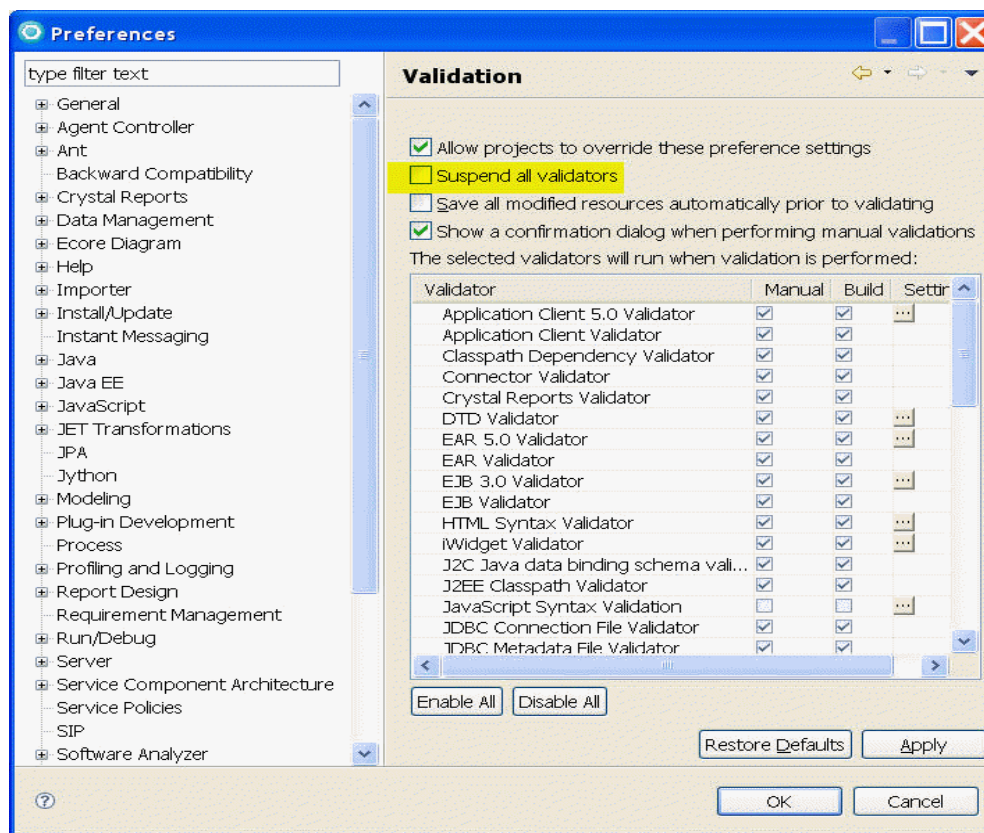
Another common approach is to turn off all of the validators at build time, but leave them enabled for manual validation, and then do periodic manual validations. A lot of validator tuning options were added in version 7.5. Explore these new options to become familiar with them.

In particular:

- The JSP Compilation Validator negatively impacts performance.
- The XML Validator can negatively impact performance if it references DTDs or XML schemas on external servers.

A slow external server will slow down the validator, which in turn will slow down the build. One way to speed up this validator is to move remote DTDs or XML schemas to your local system, and then update the XML catalog to reference these local copies. (Search for "XML catalog" in the online help for information on how to set this up).

*Figure 8. Suspend Validators*



## Other Rational Application Developer Tips

### Binary Projects

As your application grows it may not be practical or desirable to keep your entire application in source form in your workspace. For example, if you have large workspace with many projects, it is unlikely that you will need to change every project. Keep the

projects that you intend on changing in source form, and the projects that you are not changing in binary form.

To learn more about this topic, you can search the online help for "binary projects." You can also review the article [Using binary modules to optimize Rational Application Developer in a team environment](#).

## **Closing Projects**

As a general rule you should not have projects in your workspace that are not required. They cause extra overhead when building, validating, and indexing your workspace. Close or delete projects that are not required. It is better to have multiple workspaces, each with a set of related projects, than one big workspace that has everything.

## **Defragmenting**

Rational Application Developer is a very large product; therefore, to improve startup time, defragment your hard disk after installing Rational Application Developer. It is also a good idea to defragment your hard disk before installing Rational Application Developer.

Please be aware that if you have not defragmented your disk in a while, the defrag operation could take a long time to run on a large disk.

## **Do not Install Unneeded Features**

For convenience reasons many people install all of the optional product features. I recommend that you install only what you will use in your application development and uninstall features that you no longer require.

Rational Application Developer uses lazy loading as much as possible so that plug-ins are not activated unnecessarily. However, even unactivated plug-ins carry a book keeping (i.e. memory) cost, and some features can activate plug-ins (for example indexers) in anticipation of being used, if they are installed.

## **Fresh Workspaces**

Periodically starting with a clean workspace and importing your source may improve performance. This is easy to do when your projects are stored in a source control system (which is a best practice in its own right).

Over time metadata (for example log files) accumulates in the workspace .metadata directory negatively impacting performance. You can clear the .metadata directory to remove obsolete metadata.

To start with a clean workspace:

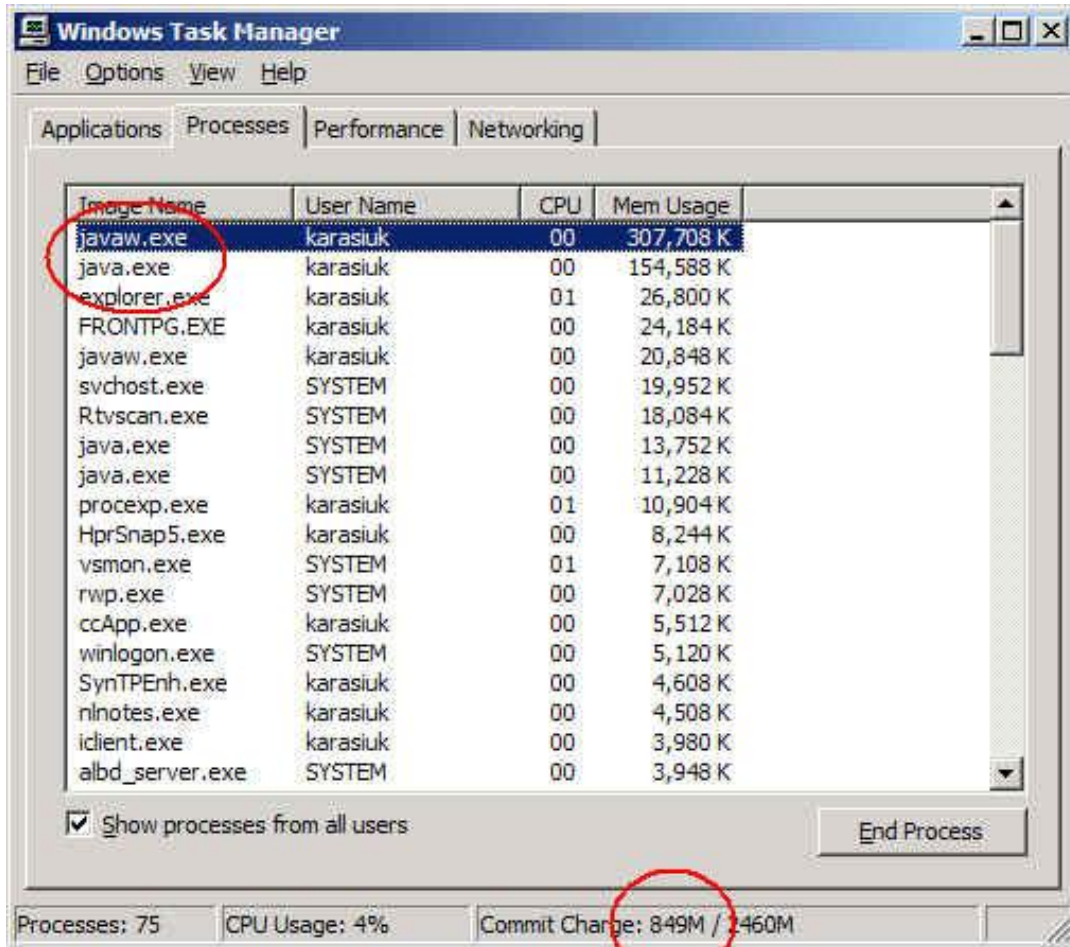
1. Make sure that you have everything checked into your source control system.
2. Export a Team Project Set for the projects in your workspace.
3. Export your preferences.
4. Close Rational Application Developer.
5. Delete your workspace directory.
6. Start Rational Application Developer.
7. Import your preferences.
8. Import your Team Project Set.

**Important:** A number of resources are not stored under source control including: preferences, key bindings, customizations, filters, working set definitions, breakpoints, VM configurations, coding templates, editor settings, decorators, and CVS repository locations. If you have changed many of these resources, you may not want to clean your workspace.

## **Task Manager**

Task Manager can help you determine if performance issues are memory based or processor based. In the case of memory based issues, Task Manager can help you determine which processes are using memory.

Figure 9. Windows Task Manager



On the Processes tab, sort by Mem Usage to determine which processes are using the most physical memory. The Rational Application Developer image name is javaw.exe. The WebSphere® test server image name is java.exe. Note the distinction between javaw and java.

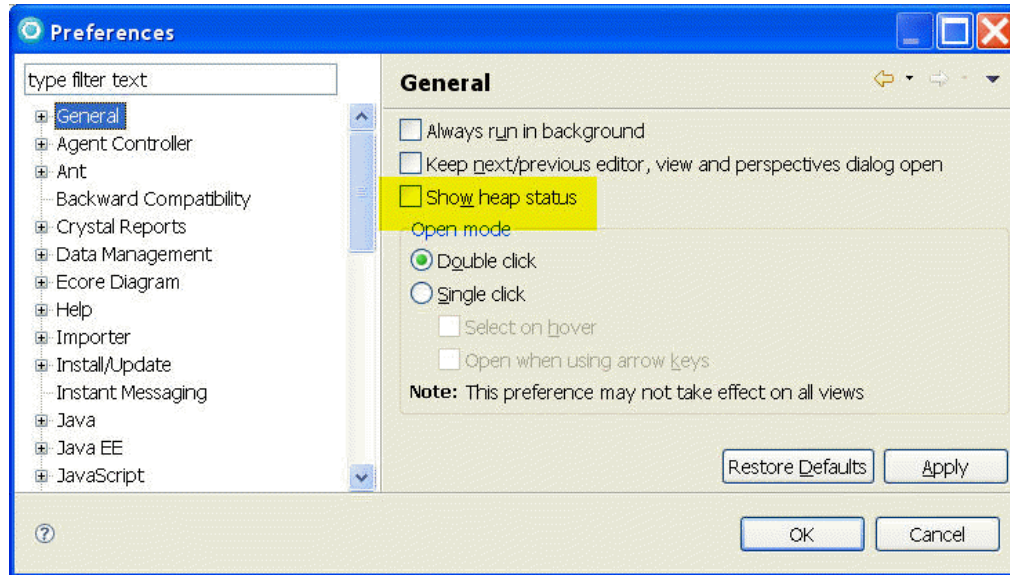
At the bottom of the window is the Commit Charge number. If the Commit Charge is higher than the amount of physical memory that you have on your system, then your system may be thrashing. In this case, reduce the amount of memory that is required or get more memory to improve performance.

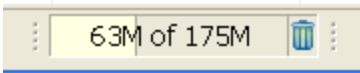
Memory is a resource that needs to be conserved, Task Manager will show you where your memory is allocated.

You can also see how much memory is being used by turning on the **Show heap status** preference in Rational Application Developer.



Figure 10. Heap Status



This will add an indicator  to your status bar that shows you both the used heap (in this example 63 MB) and the total heap (in this example 175 MB).

## Remote Test Server

If your computer is too small to fit Rational Application Developer, WebSphere test server, and possibly a database manager, consider setting up a remote test server. If you have access to a second system you can run the test server on the second system to free up resources on your development machine. Refer to [Setting up a Remote WebSphere Test Server for Multiple Developers](#) for more information. The article was written for specifically for WebSphere Studio Application Developer, but the ideas can be applied to Rational Application Developer and the WebSphere test server.

Running multiple local test servers in addition to Rational Application Developer on one system will consume a lot of memory. At most, run one local test server.

## Restart Rational Application Developer

As you use more functions, more classes are loaded into memory. Rational Application Developer uses lazy loading, so the code for a particular function is not loaded until it is required. For example, if you do not usually use the XML editor, but in this session you edited one XML file, the XML editor is loaded only when it is needed. However, once it is loaded it will not be un-loaded. The only way to free this memory is to close and restart

Rational Application Developer. You should close Rational Application Developer at least once a day.

If you find a memory leak, please contact us. We take memory leaks very seriously and would be eager to investigate it.

## JVM Tuning

There are a number of articles, that you can read, that discuss parameters used to tune the JVM. The main parameter that is usually mentioned is the **-Xmx** parameter. This parameter is used to limit the amount of memory that the Java™ heap is allowed to allocate. The default for Rational Application Developer is set to 1 GB. It is set to this value so that Rational Application Developer can handle very large and complex workspaces. You can try reducing this to something smaller, although I would not recommend going below -Xmx512M.

**Important:** Rational Application Developer will fail with OutOfMemoryError if -Xmx is set too low.

You can also experiment with the parameters **-Xmaxf** and **-Xminf**. These parameters control how the heap is expanded and contracted.

The defaults are:

- -Xmaxf0.6
- -Xminf0.3

The numbers are percentages. **-Xmax0.6**, instructs the JVM to compact (shrink) the heap if the amount of free space exceeds 60%. Setting this to a smaller number will make your heaps smaller at the expense of doing more garbage collections. For example, **-Xmaxf0.4** will cause heap compaction if the free space exceeds 40%.

The **-Xminf** parameter is used to control the minimum free space. The setting **-Xminf0.3** instructs the JVM to expand the heap, if after doing a garbage collection it does not have at least 30% free space.

If your system has memory problems you can experiment with **-Xmaxf0.4 -Xminf0.2**, or even the more aggressive **-Xmaxf0.2 -Xminf0.1**. The advantage of using these parameters over the -Xmx parameters is that you will not get an artificial OutOfMemoryError error message.

If you want to change any of these parameters, edit the **eclipse.ini** file. This file is found in the Rational Application Developer installation directory.

## Shared Classes

Rational Application Developer uses shared classes to improve startup time. This is enabled by specifying the **-Xshareclasses** directive in the eclipse.ini file. If you need to save memory, you can remove this directive as well as the **-Xscmx96m** directive; however, removing these directives will increase your startup times. For example, on my laptop, a warm start of Rational Application Developer takes 11 seconds and uses 249 MB of memory (working set). When I remove the shared class cache, the startup time increased to 14 seconds and memory dropped to 209 MB.

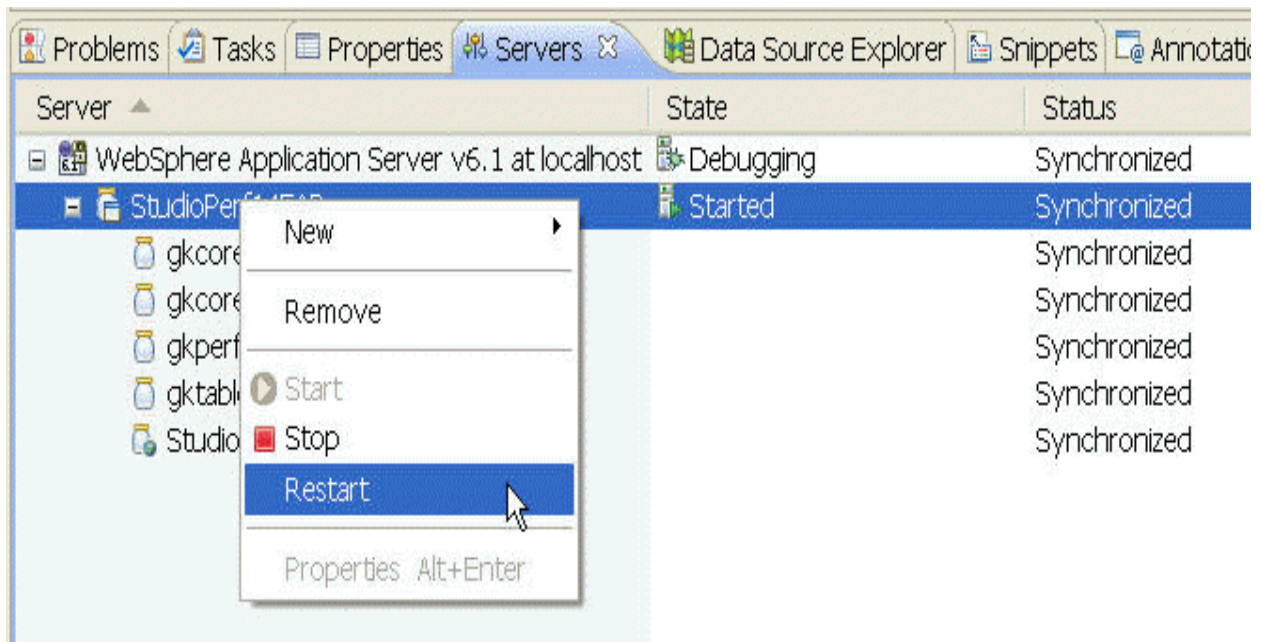
As an alternative to removing the directives, you can lower the -Xscmx parameter. For example, using -Xscmx48m cuts the shared class cache in half. If you have multiple copies of Rational Application Developer running at the same time (most people do not), then the shared class cache will save memory as well as improve your startup times.

## Test Server Tips

### Restarting Projects (instead of the Server)

It is considerably faster to restart a project instead of restarting the server.

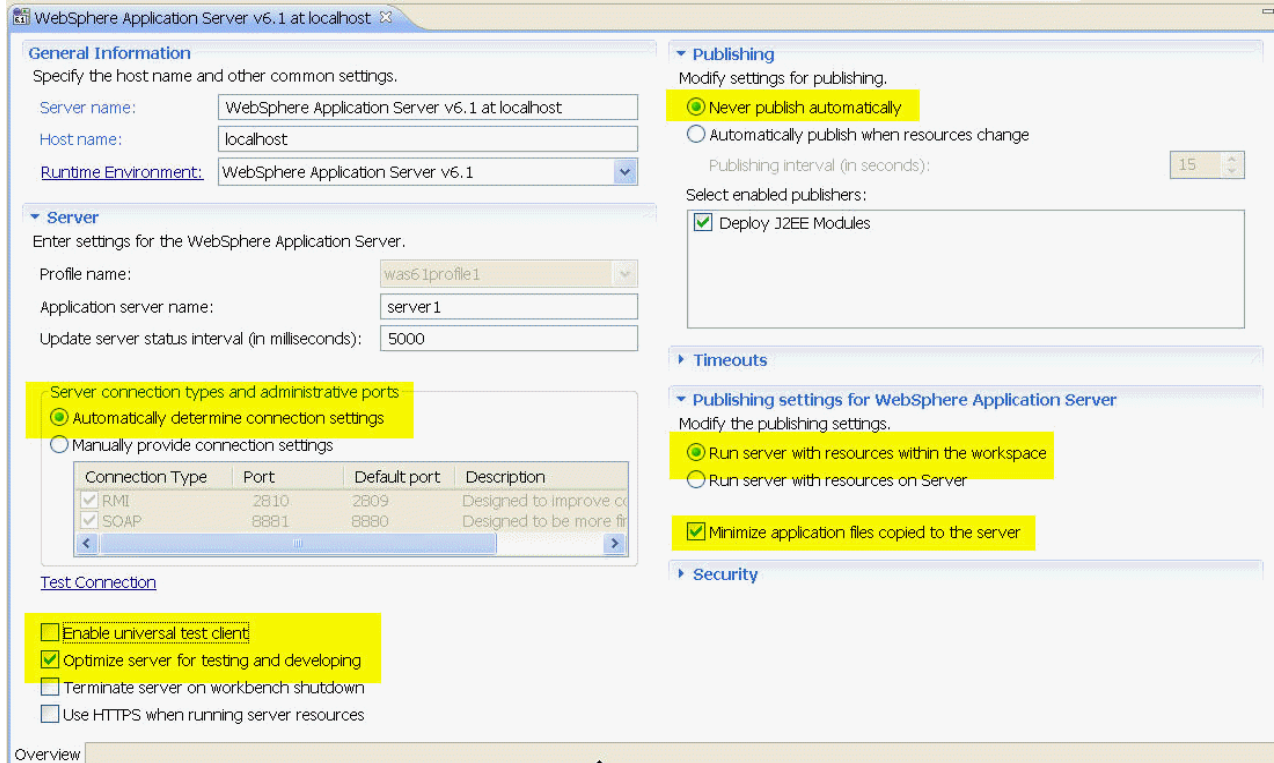
Figure 11. Restarting a Project



### Server Configuration Options

There are a number of server options that significantly affect performance.

Figure 12. Server options



To open the Server editor, double-click a server entry in the Servers view.

The highlighted items all affect performance. Most of the items in the view are set by default to give you the best performance, except for: **Publishing** and **Enable universal test client**.

If you do not need the universal test client, clear the checkbox so that it is not installed.

I recommend setting the publishing option to **Never publish automatically**. There are many changes that you can make that will update the server without having to publish. For example, if you are running with resources within the workspace (the default), then any changes to HTML or JSP files will be detected by the server without publishing. Likewise any changes to servlets are detected as part of the automatic servlet reloading functionality of the server.

The **Optimize server for testing and developing** option should always be checked. This options sets some JVM parameters that improve the startup time of the server.

## Running in Debug Mode

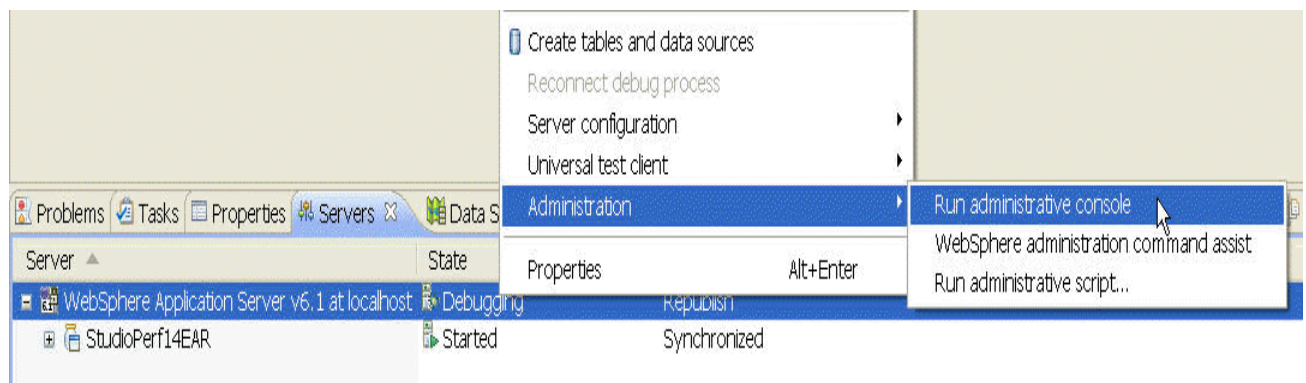
I recommend running the server in debug mode because the JVM supports hot method replace. Your Java changes will be injected into the running server, without having to do

a publish. Be aware that there are some class changes (for example the size or shape of the class changes), that can not be injected using the hot method replace functionality. If this situation happens you will get an informational message right after saving your change.

## Server Startup Options (Admin Console)

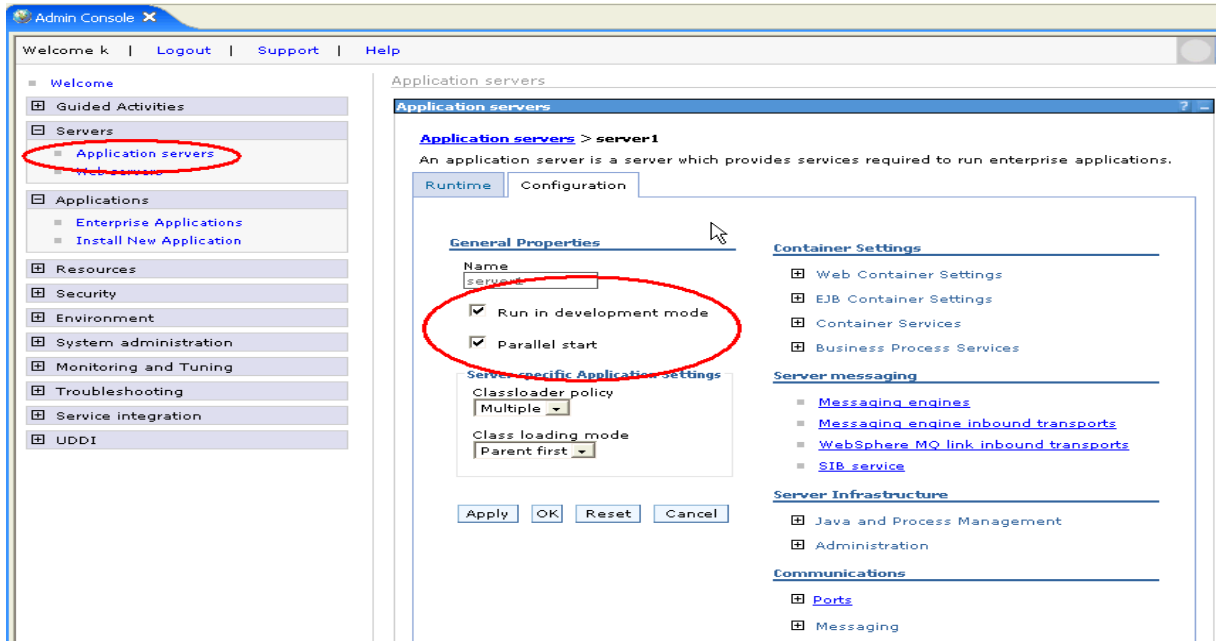
In addition to the server configuration changes that you can make inside of Rational Application Developer, there are additional changes that you can make from the WebSphere Admin Console. Once you have your server running, you can start the Admin Console from inside of Rational Application Developer by selecting the **Run administrative console** menu item from the server context menu, as shown below:

Figure 13. Administrative Console



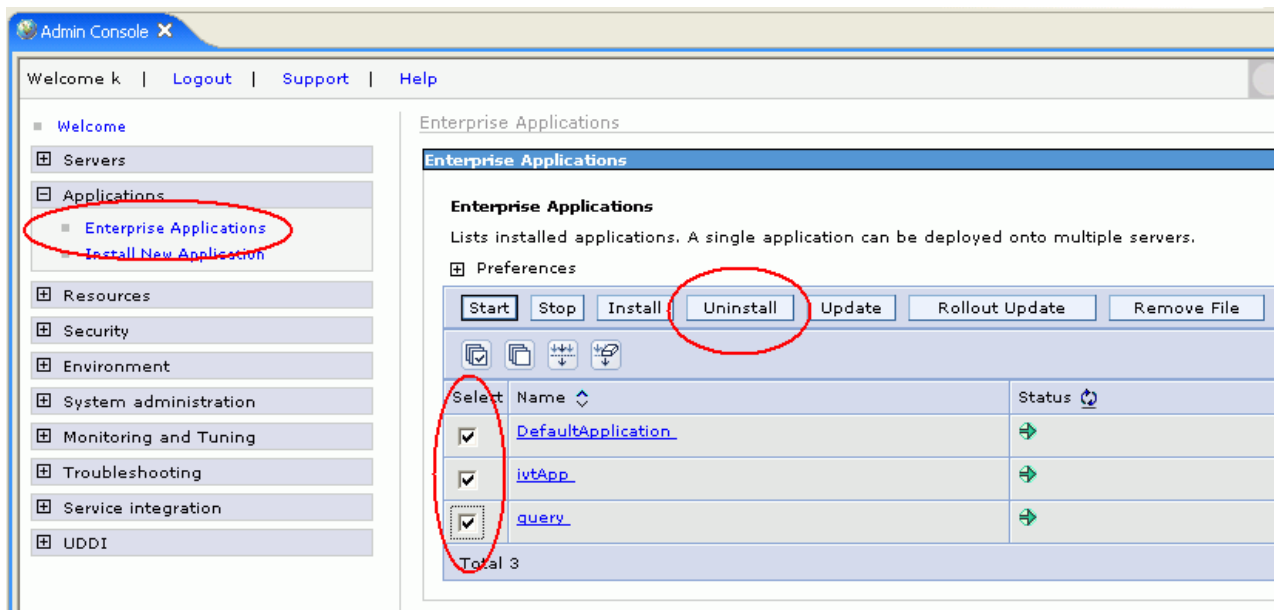
From the Admin Console, select the server (it is usually called server1), and ensure that the **Run in development mode** and **Parallel start** options are selected. This will provide even more server startup improvements.

Figure 14. Startup Improvements



While you have the Admin console running, you may want to remove any unneeded applications from the installed applications list. There are a couple of small sample applications (DefaultApplication, ivtApp, and query) that are installed by default. These can be removed safely.

Figure 15. Removing Unneeded Applications



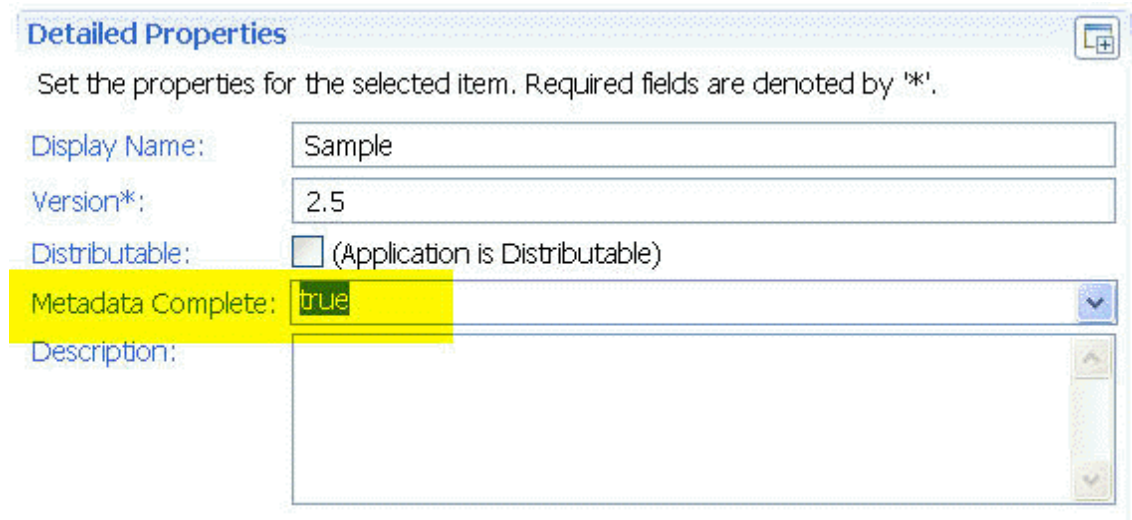


## Publishing and Annotations

One of the exciting features in WebSphere Application Server V7 is support for Java EE 5, specifically using annotations as part of the programming model. It takes time to parse the new annotations. If you know that your application does not contain annotations, there are some settings that you can use that will dramatically reduce the amount of time that it takes to publish (or republish) your application. This tip applies to Dynamic Web Modules at the 2.5 level.

If you know that your module does not contain any Java EE annotations, set the Metadata Complete property to true in the web.xml deployment descriptor. This will signal to WebSphere Application Server that it does not need to scan the classes and JAR files for annotations.

*Figure 16. Signaling WAS to not scan for annotations*



The screenshot shows a 'Detailed Properties' dialog box with the following fields:

- Display Name: Sample
- Version\*: 2.5
- Distributable: ☐ (Application is Distributable)
- Metadata Complete: true (highlighted in yellow)
- Description: (empty text area)

If your project contains annotations, WebSphere Application Server has some directives that instruct the server not to scan certain JAR files or package names. Read about the directives `com.ibm.ws.amm.scan.context.filter.archives` and `com.ibm.ws.amm.scan.context.filter.packages` in [Java virtual machine custom properties](#) for more information.

## Updates

We continue to make performance improvements to Rational Application Developer, as new service releases become available. I recommend always moving to the latest fix pack.

# Conclusion

This paper offers several tips for improving the performance of Rational Application Developer.

If you know of other tips, not included in this article, please contact me at [karasiuk@ca.ibm.com](mailto:karasiuk@ca.ibm.com).

# Other Resources

Here are some links that you may find useful:

- [Eclipse Performance page](#)
- [Optimizing Multi-Project Builds Using Dependent Project JARs in WebSphere Studio Application Developer](#)
- [Setting up a Remote WebSphere Test Server for Multiple Developers](#)
- [Tune Eclipse's startup performance with the Runtime Spy: Getting Started](#)
- Tune Eclipse's startup performance with the Runtime Spy: Winning the Shell Game

## *About the author*



**Gary Karasiuk** is a performance analyst at the IBM Toronto Laboratory. You can reach Gary at [karasiuk@ca.ibm.com](mailto:karasiuk@ca.ibm.com).