
WebSphere Portal Application
Development Best Practices using
Rational Application Developer

Agenda

- RAD Best Practices
- Deployment Best Practices
- WSRP Best Practices
- Portlet Coding Best Practices
- Portlet JSP Best Practices
- Portlet Packaging Best Practices
- Portlet Configuration Best Practices
- Portlet Session Management Best Practices
- Portlet Internationalization Best Practices
- Resource Serving Best Practices
- Portlet Performance Best Practices
- Portlet Security Best Practices
- Portlet Documentation Best Practices
- JSF portlets Best Practices

RAD Best Practices

Install Options

- IBM Installation Manager simplifies and optimizes install
  - Minimize install footprint by only installing options you need
  - Shell-share with other Rational products to help more easily manage the desktop
  - Enterprise install capabilities for simple maintenance and provisioning of your own standardized install configurations (create response files for what install configuration you need)

Flexible Help Options

- Install documentation based on needs of your environment
  - Available on the world-wide web
  - No footprint on local workstation
  - Available on internal web server
  - No footprint on local workstation
  - Available for download on each workstation

Basic Hygiene

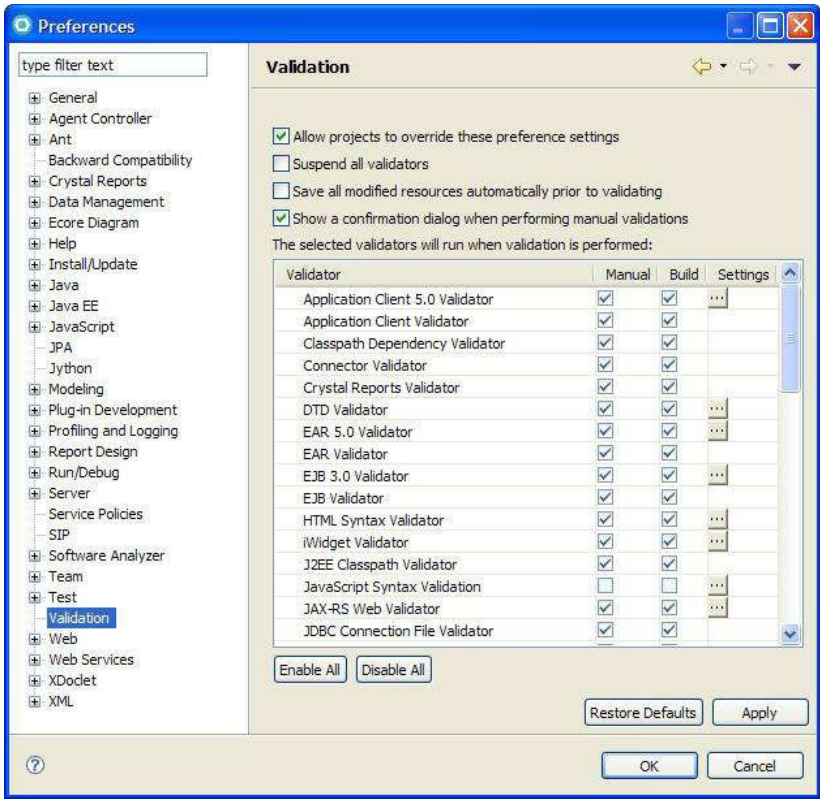
- Close unused/unneeded projects
- Close unused editors and views
- Use multiple workspaces
  - Don't have one big workspace with everything
  - Better to have special-purpose workspaces

Use Binary Modules (AKA Shared EARs)

- Reduces the number of source projects in the workspace
  - Most projects kept in binary form
- Basic approach
  - Start with EAR file containing complete application, including source code.
    - Typically produced by a nightly or weekly build
  - Developer imports this EAR as a base
  - Developer imports from source control the projects to be changed
    - Source projects build against the binary modules in the EAR

Tune Validation

- Lots of different types of validation
- Validation can be “tuned”
- Much more tuning has been added in RAD 7.5
- Turning off validation will reduce error checking



Java EE 5 Annotation Scanning

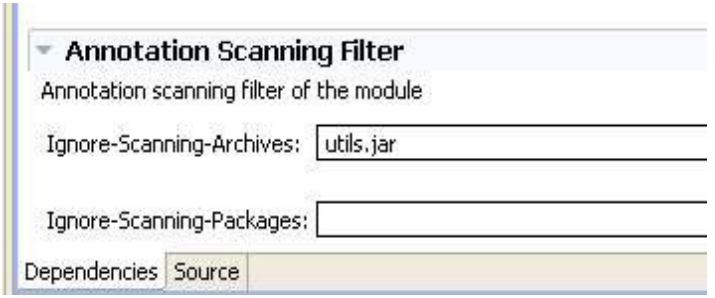
- Speed up application install
  - Provide a list of archives, or utility JAR files, that do not contain annotations
  - Can provide a list of Java Packages that do not contain annotations

Annotation Scanning System Property

- For archives
  - com.ibm.ws.amm.scan.context.filter.archives
- For packages
  - com.ibm.ws.amm.scan.context.filter.archives

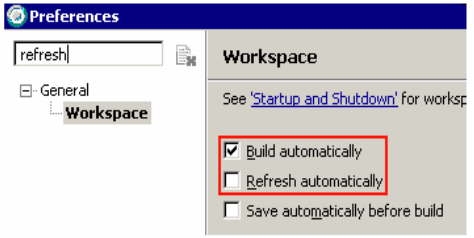
Annotation Scanning Property Files

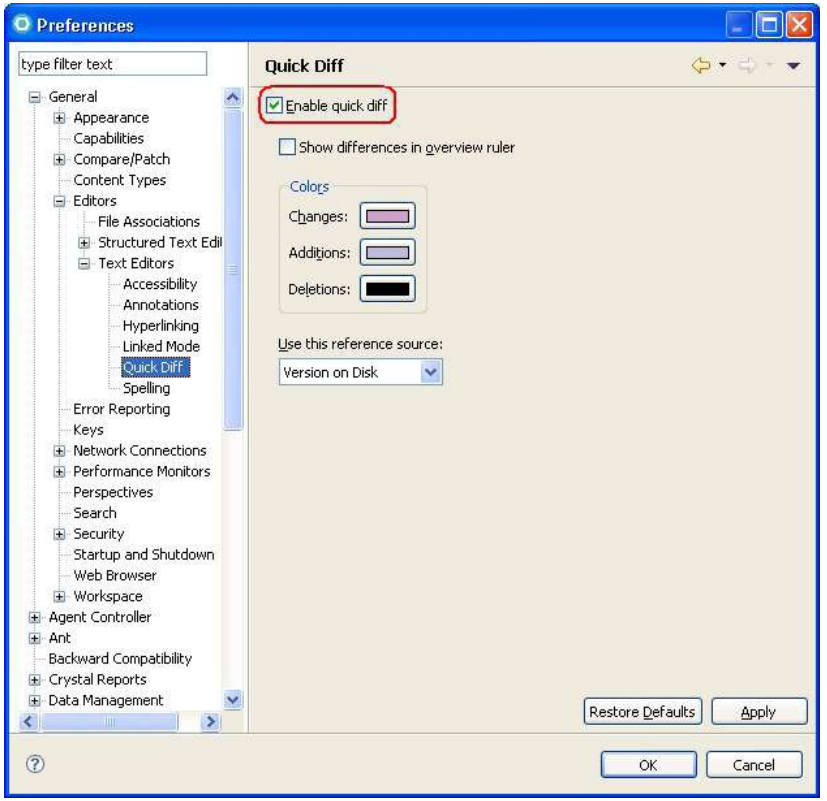
- For archives, Ignore-Scanning-Archives
- For packages, Ignore-Scanning-Packages
- File amm.filter.properties in was_home/properties
- File amm.filter.properties in profile_home/properties
- META-INF/MANIFEST.MF in EAR
- META-INF/MANIFEST.MF in module



Consider...

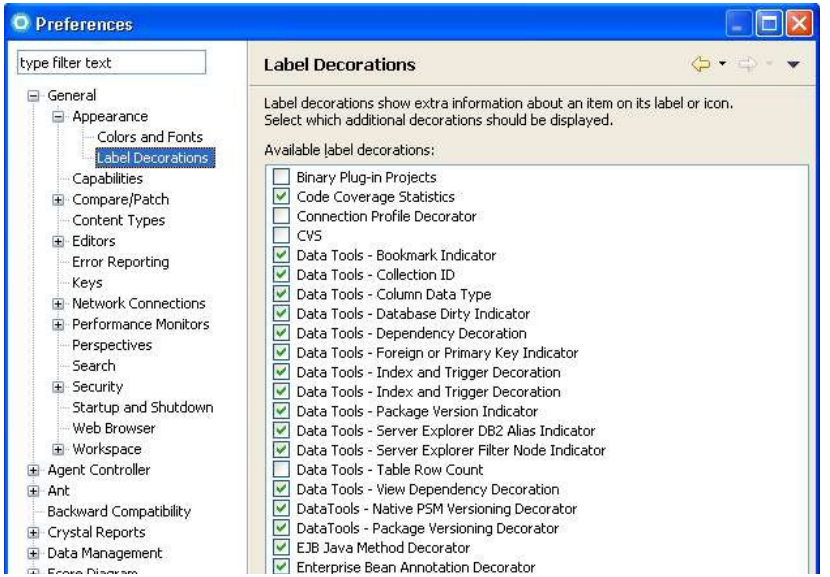
- Turning off “Build automatically”
- Turning off “Refresh automatically”
- Sharing preferences across individual workspaces
- Turn off Quick Diff

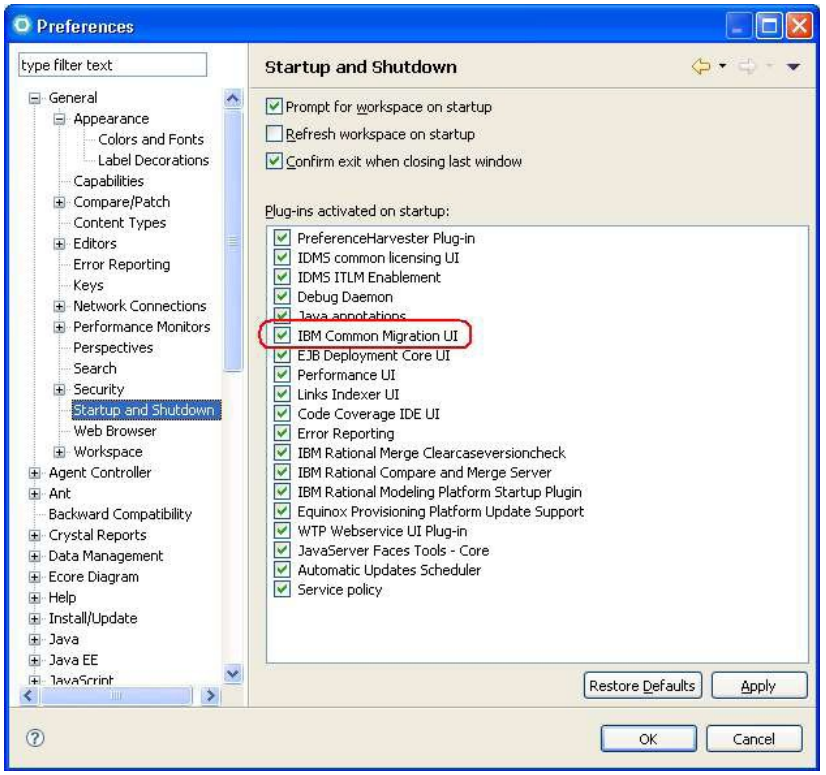




Consider...

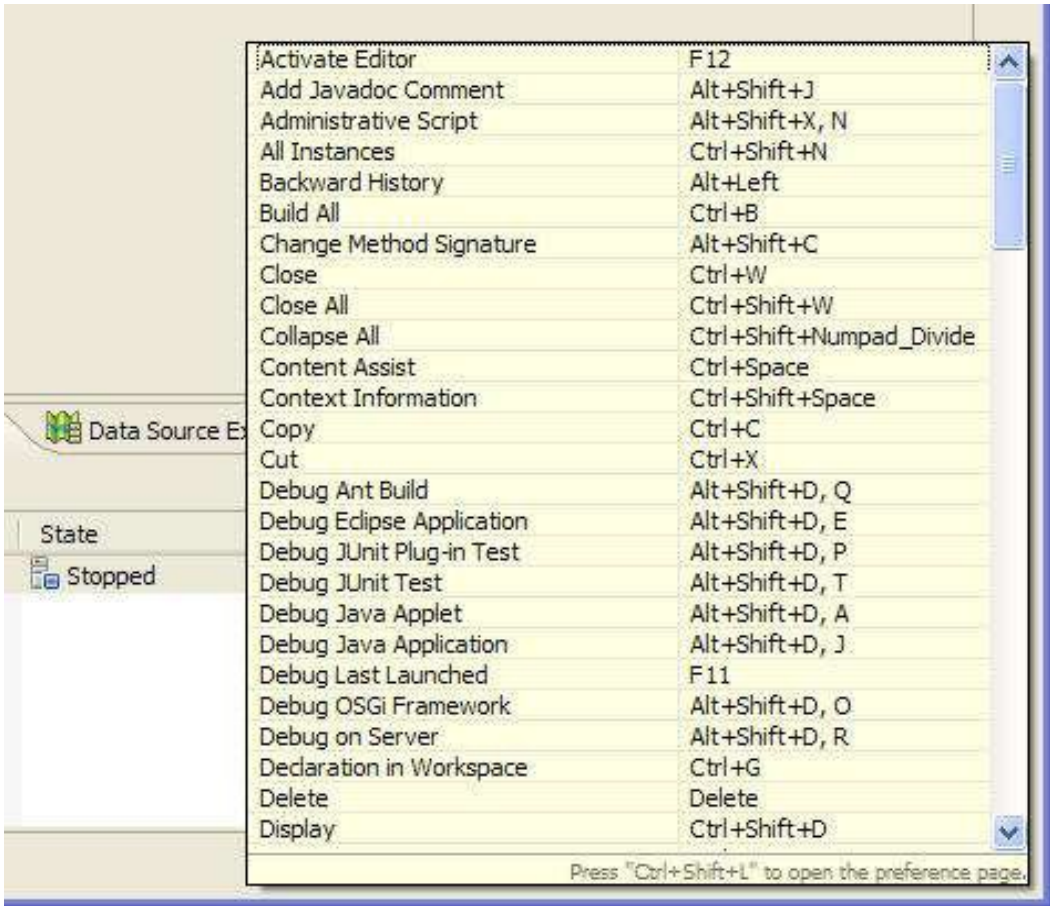
- Turn off label decorators
- Disabling Links Indexer
- Tuning Early Startup





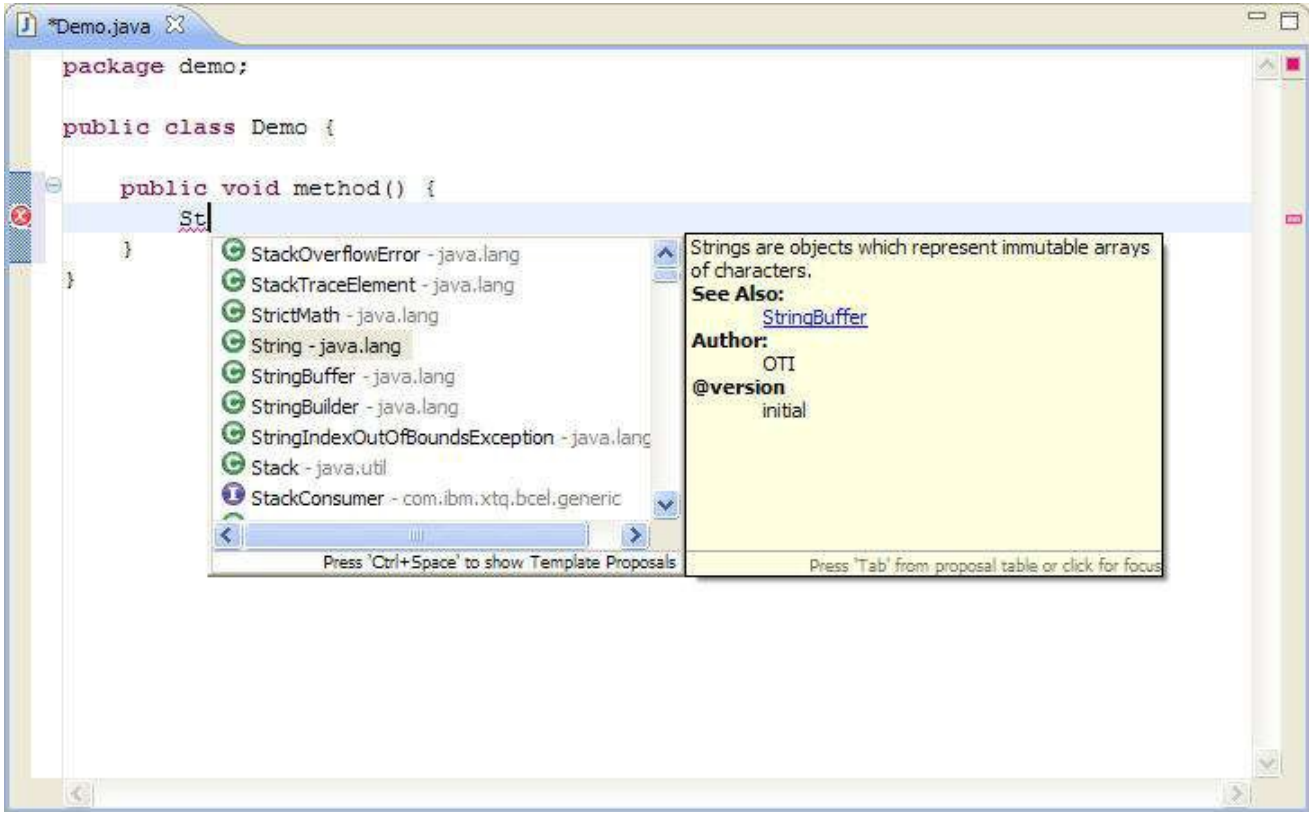
Tricks and Tips

- Show Key Assist (Ctrl + Shift + L): Use this to get a list of all key bindings in eclipse



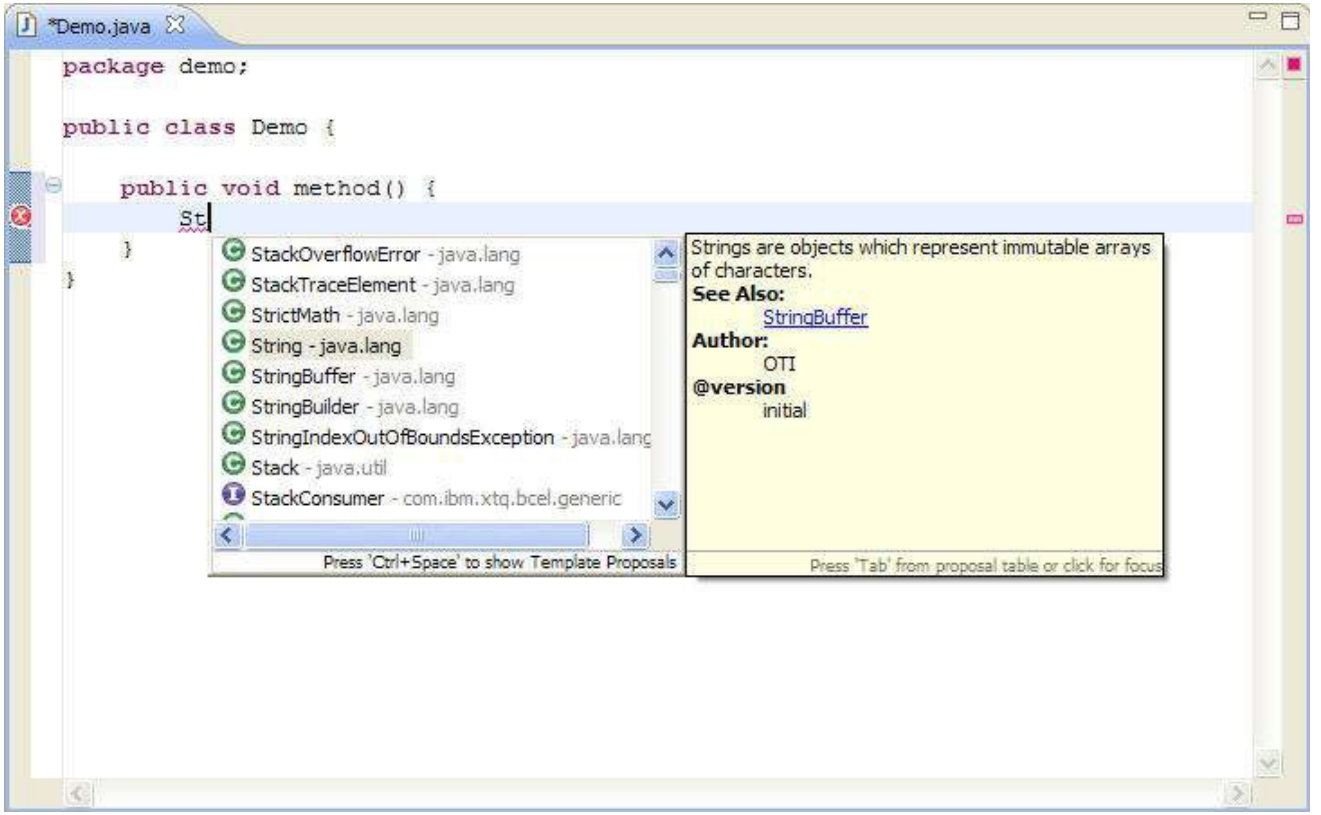
Tricks and Tips

- Content Assist (Ctrl + Space): Use this to provide context sensitive content completion



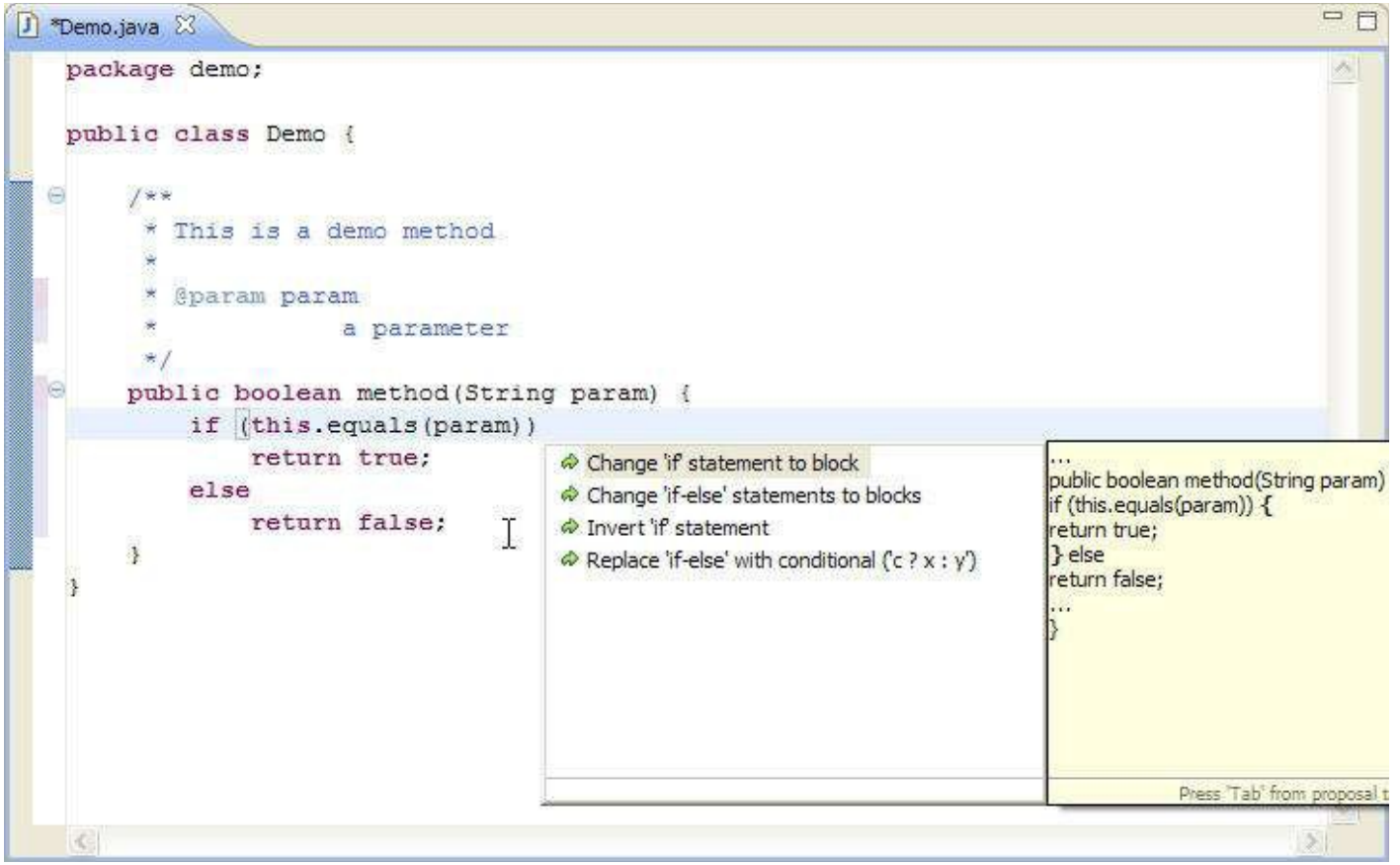
Tricks and Tips

- Word Completion (Alt + /): Type a letter or two and then simply press Alt-/ to cycle through all words starting with those letters.



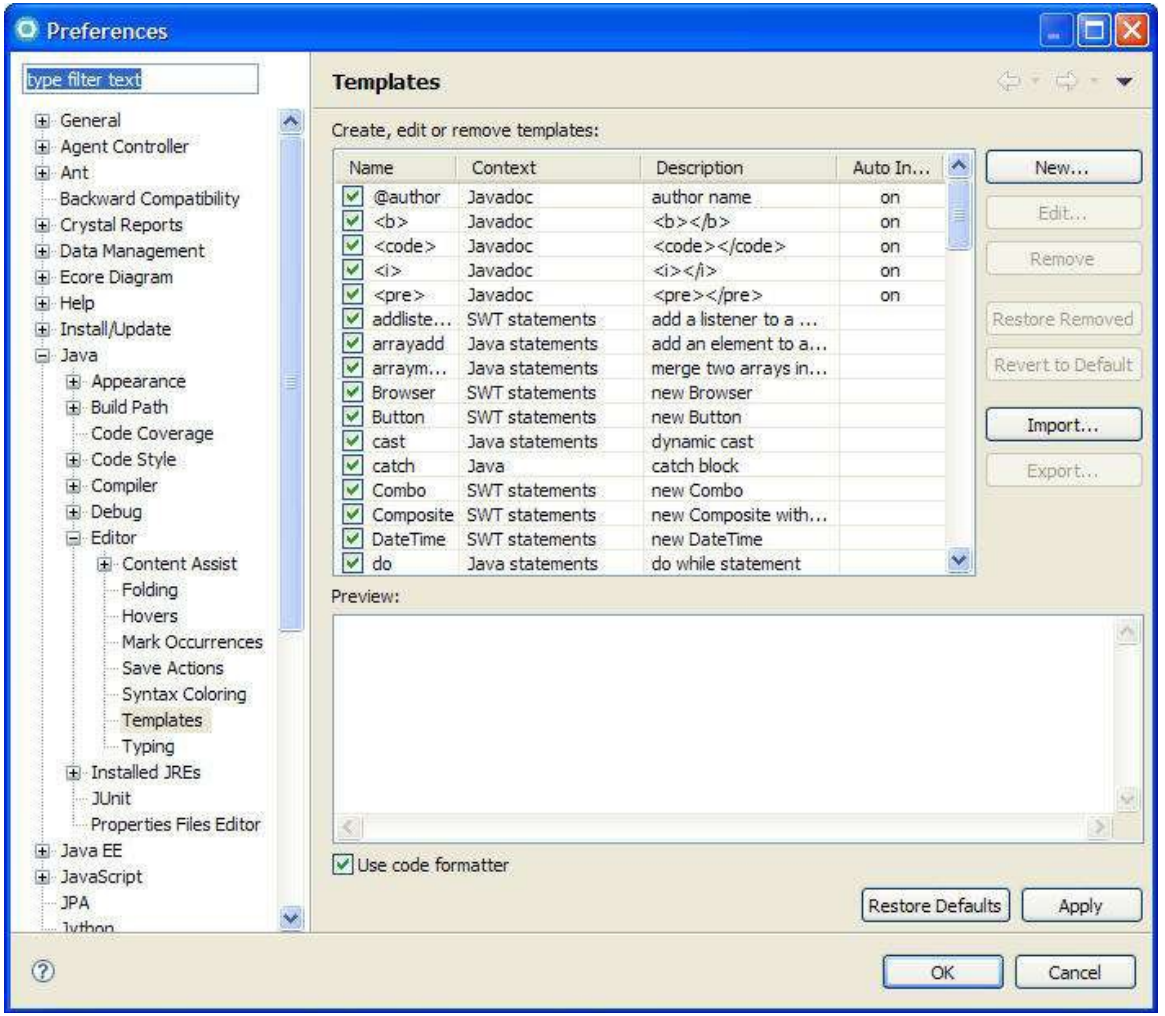
Tricks and Tips

- Quick Assist (Ctrl + 1): Use this to assist you to perform local code transformations.



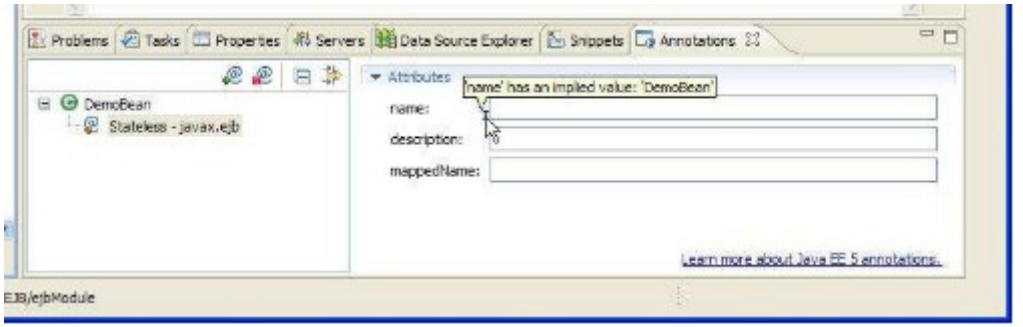
Tricks and Tips

- Editor Templates
  - a structured description of coding patterns that reoccur in source code.
  - used to fill in commonly used source patterns.
  - Templates are inserted using content assist (Ctrl+Space).



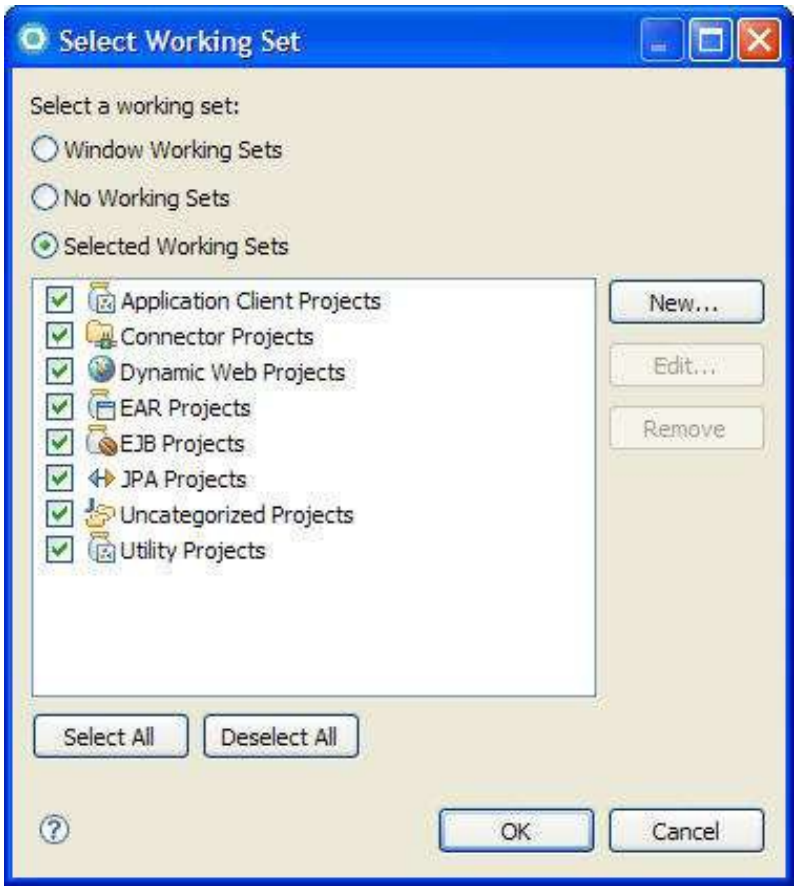
Tricks and Tips

- Annotations View - Add or delete annotations, and to modify the attribute values of annotations



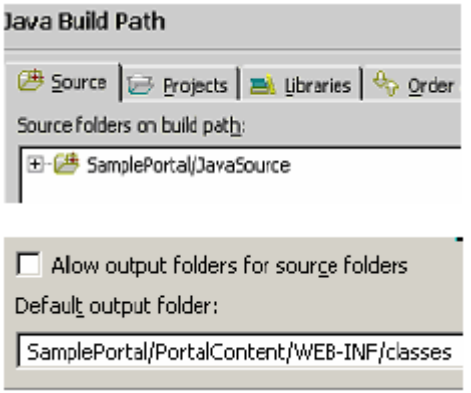
Tricks and Tips

- Working Set
  - categorize resources across projects into a contextually relevant representation
  - sub-set of files/classes/folders/ projects that represent a certain developer workflow.



Tricks and Tips

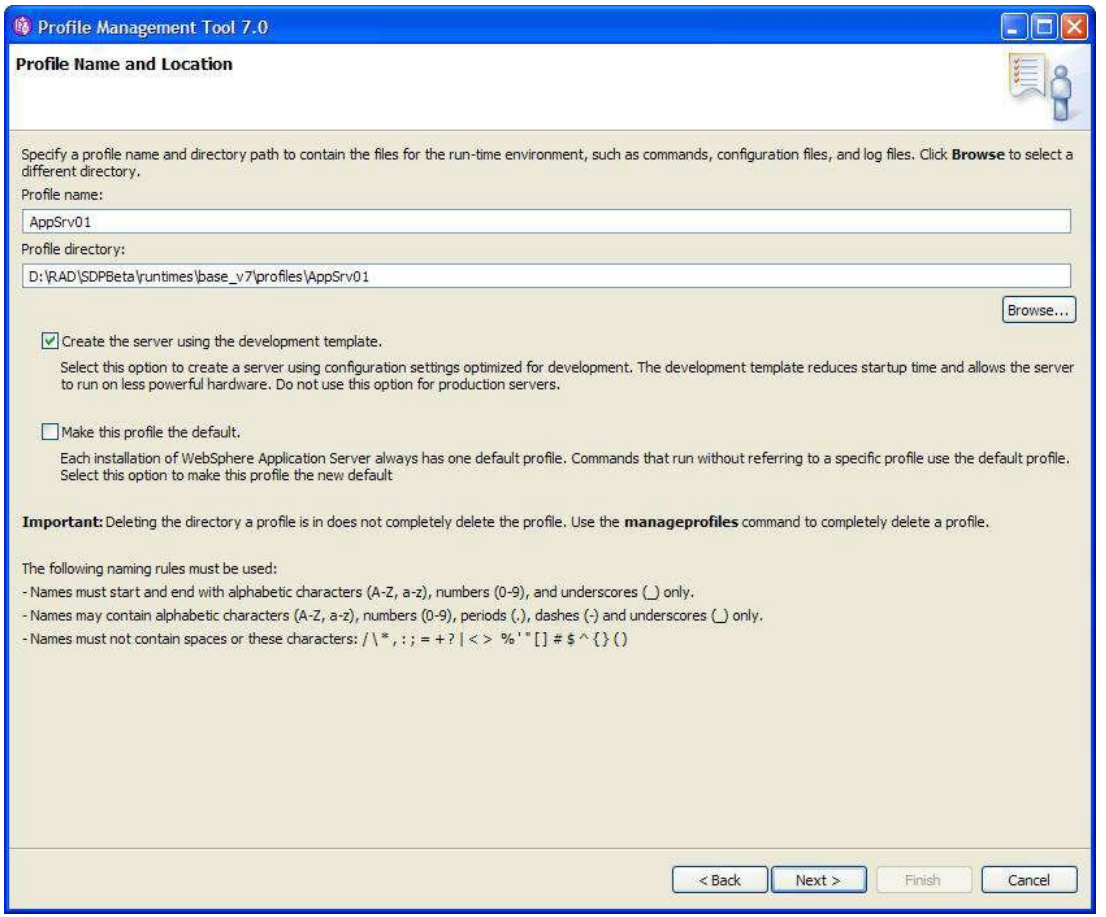
- Hot swapping
  - Set build folder to <*project_name*>/WebContent/WEB-INF/classes for portlets
  - ensure that projects from the local workspace are used for deployment (for the local server when you use Run on Server)



Application Deployment Best Practices

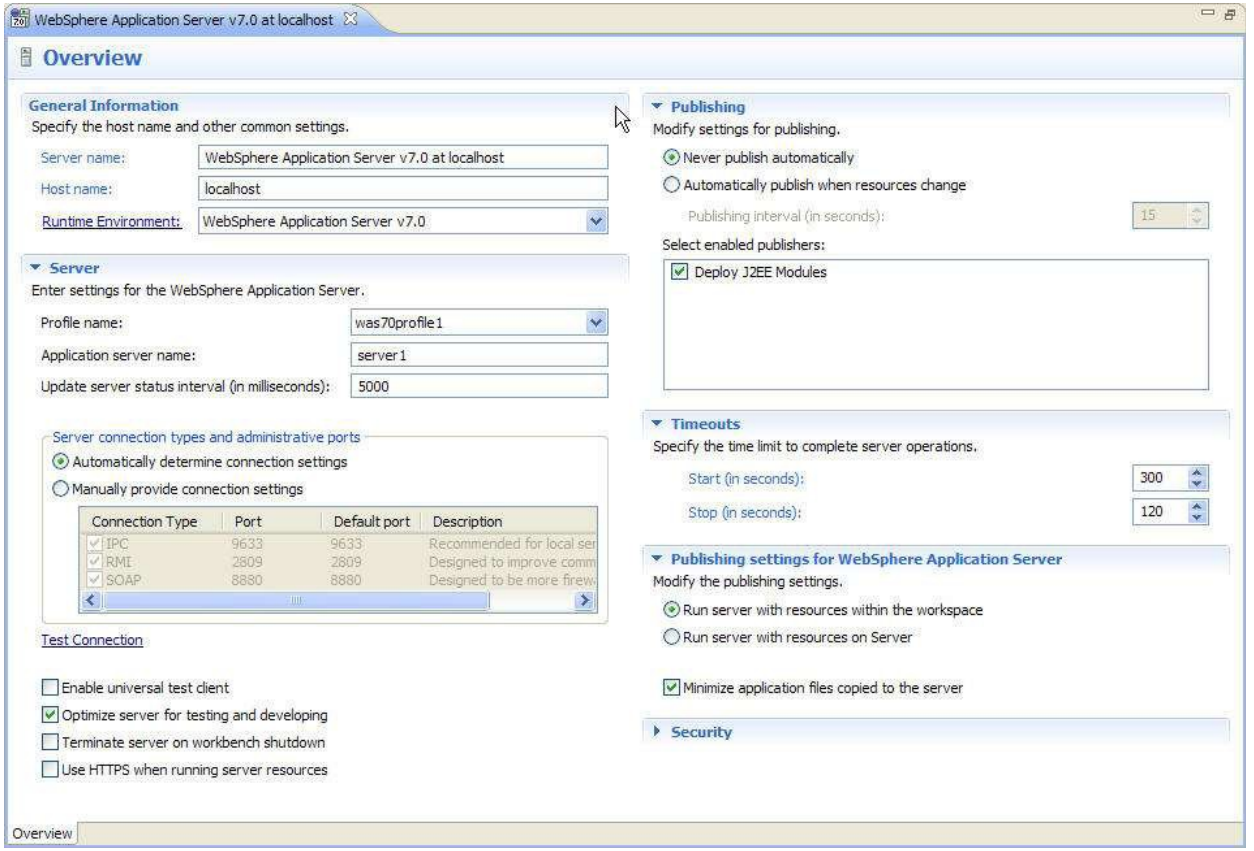
Use WAS Developer Profile

- WebSphere Test Environment already uses this
- WebSphere Profile designed for developers
  - Faster startup time
  - Faster application restart



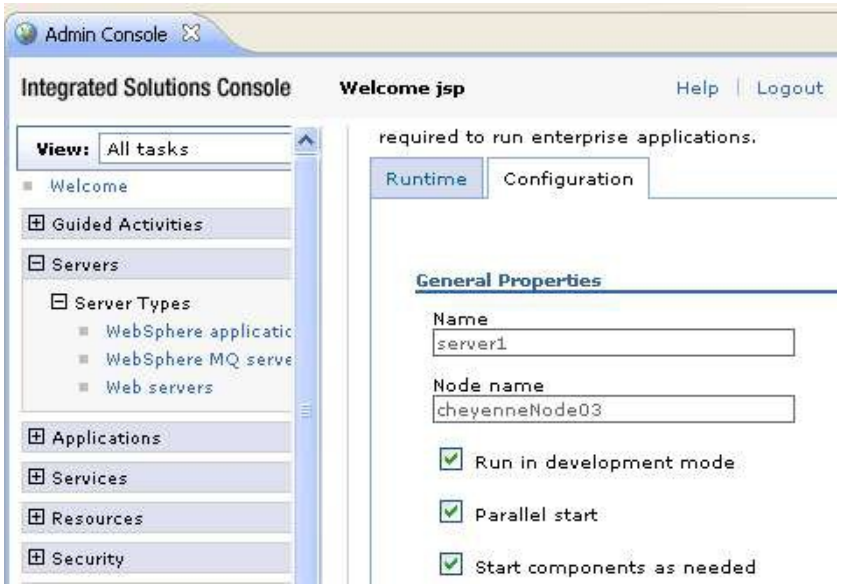
Tune WAS Options

- Be aware of different server options
- Can also run server on a different machine



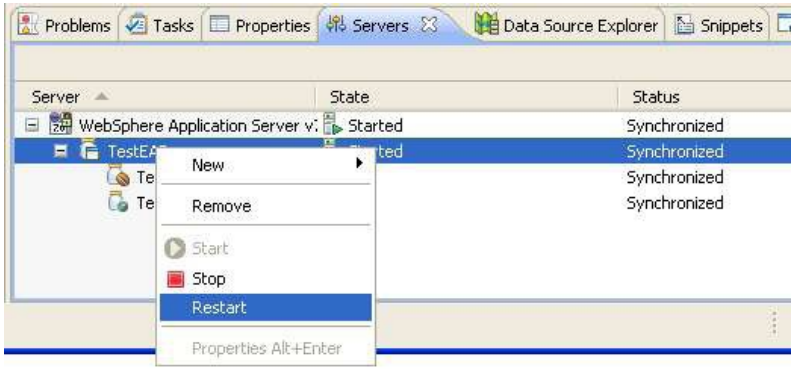
Server Startup Options

- Ensure that these are selected:
  - Run in development mode
  - Parallel start
- Start components as needed

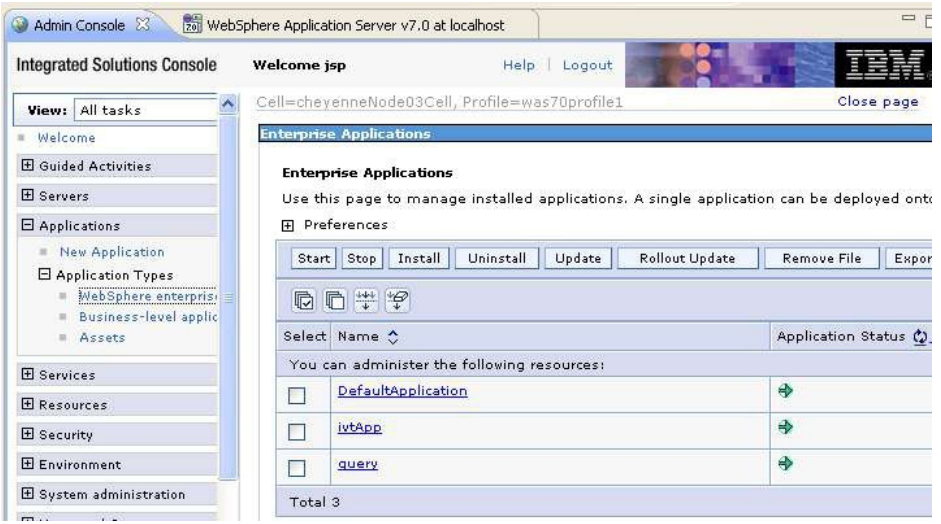


WAS basic hygiene

- Restart applications instead of the server



- Remove un-needed applications like sample applications
  - DefaultApplication
  - IvtApp
  - query

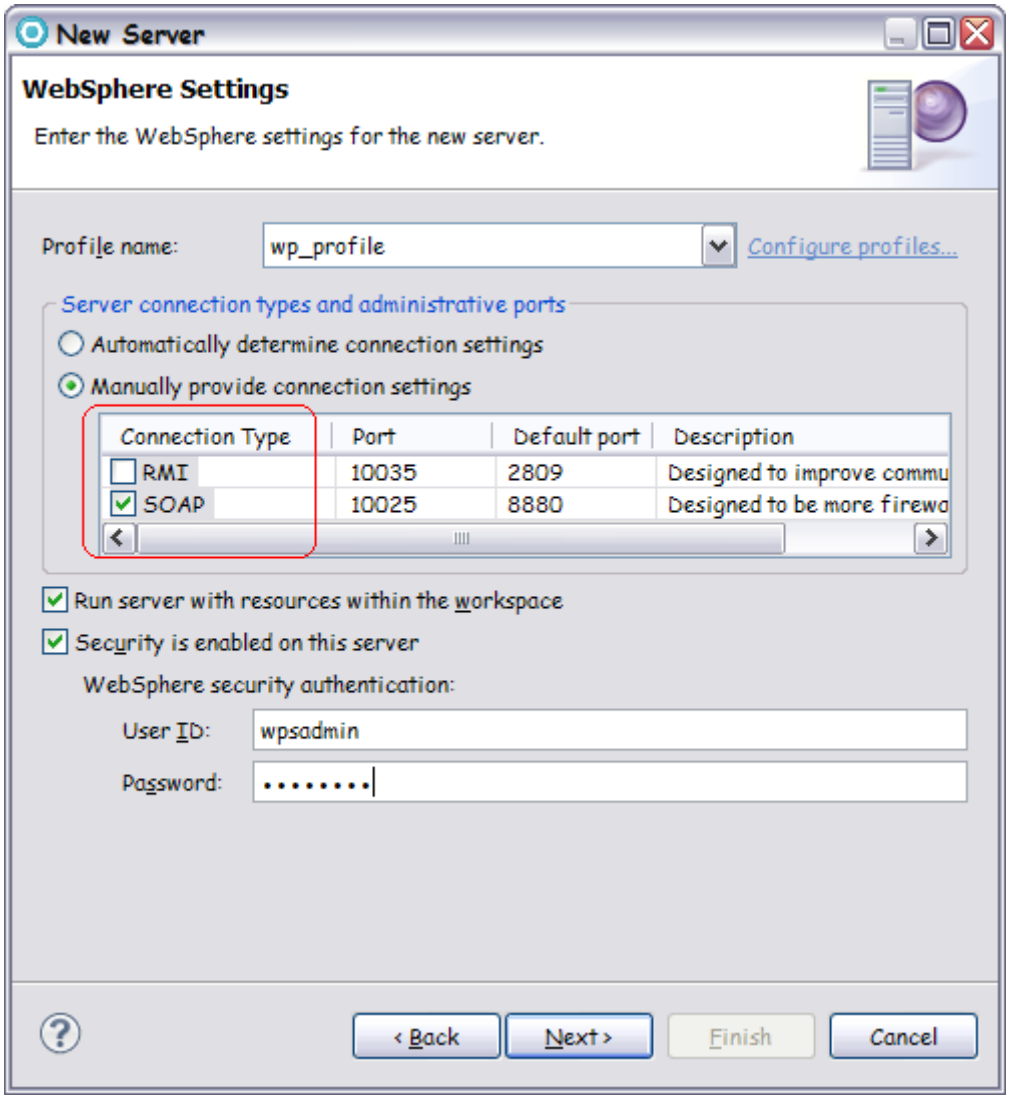


WebSphere Portal Installation

- Install the server using “Base” option
  - Deploys a base set of portal features
  - Allows selective enablement at any time later
  - More efficient and speedy portal behavior

Server connection protocols in RAD

- SOAP recommended for remote and local servers



Multiple profiles on WPv7

- Newly introduced feature in WPv7
- Allows for creating and maintaining multiple profiles to create multiple independent portal instances

Automatic publishing

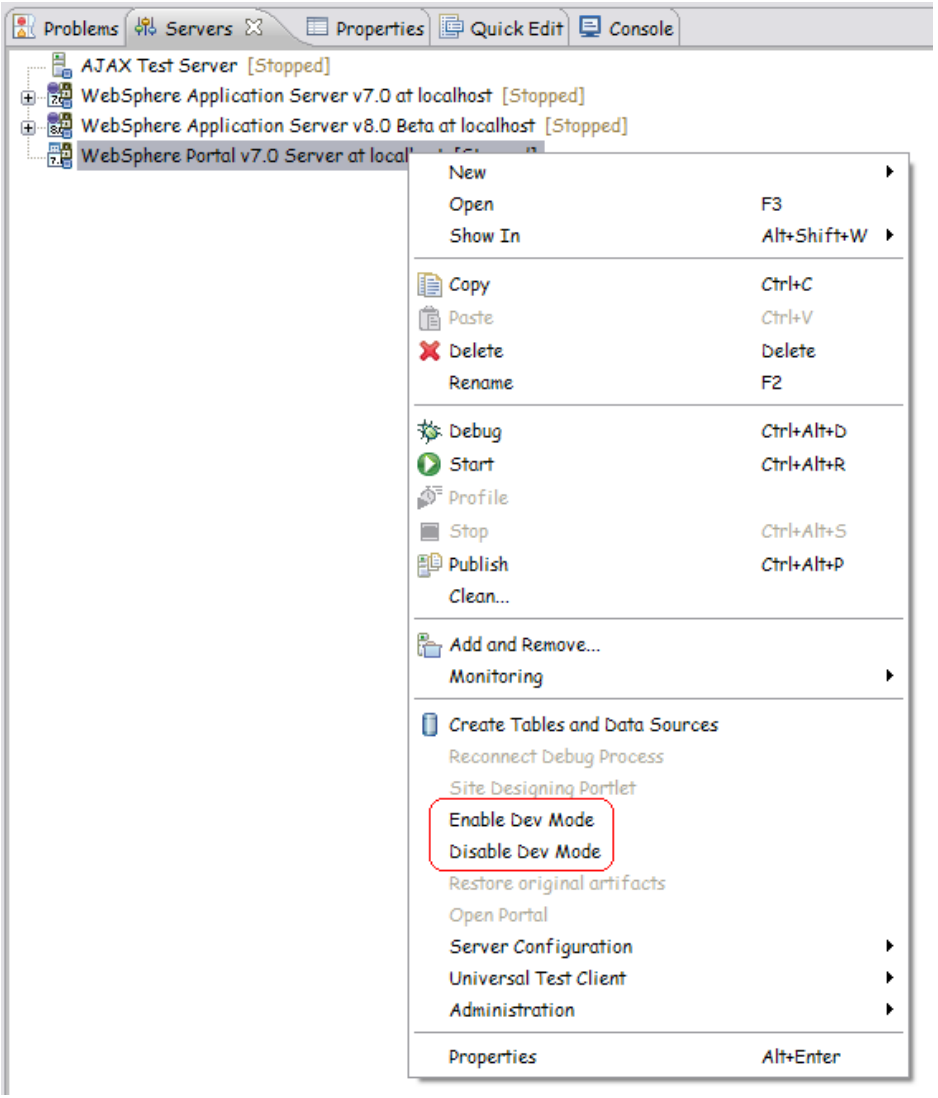
- Use option “Never publish automatically”
- Use “Override default settings” if required to publish the changes whenever they are made but use it with care

Enable hot method replace in debug mode

- Enables hot replace and facilitates just-in-time compilation in debug mode.
- Speeds up the application deployment and also improves server startup time.

Enable developer mode

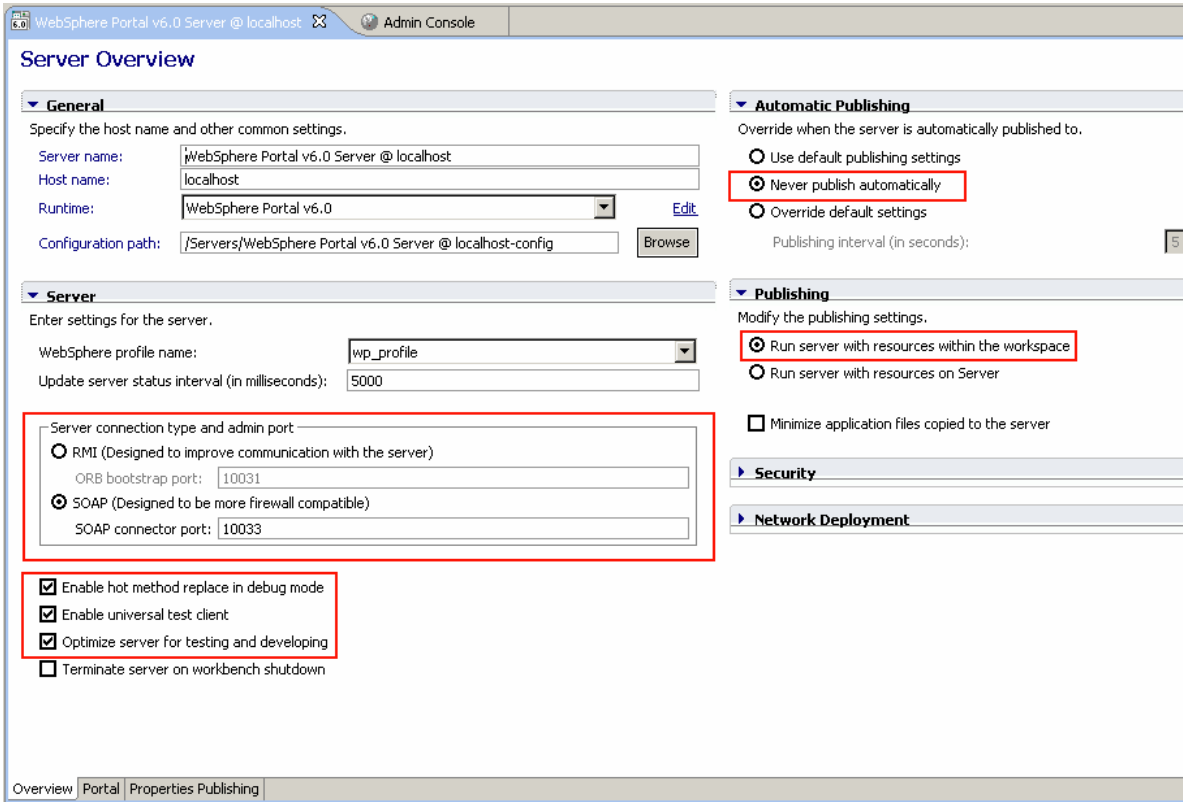
- Efficient and productive use of server
- Portlets and web applications are not activated on start-up
  - Activated on first access
- Switches JVM heap size to maximum
  - Reduces amount of garbage collection during startup
- Not recommended for production environments



Remote server use

- Consider using remote server if
  - Desktop machine has less memory
- Can cause concurrency issues if multiple users are using same remote server

Recommended server editor settings in RAD



Remote Portlet (WSRP) Best Practices

Consider WSRP when …

- Looking for portlet development which is faster to develop and test
- The Portlet requires huge loads for CPU and/or memory and is not really suited as a co-located application
- Portlets are owned and controlled by independent teams
  - Infrastructure and version management is not possible
  - Flexibility is key
  - Cross organisation usage would be in this category as well
- Portlet quality and level of testing does not match the quality and SLA of the overall Portal
  - Run the portlet remotely for some time to proof readiness and then move local
  - Typically called "quarantine Portal"
- If the remote connection is very chatty, or not existent
  - WSRP can consolidate a set of web services calls into a single WSRP call
  - Allows for co-location of the Portlet, if only local java api is available on remote server

DO NOT consider WSRP when …

- Access to IBM extensions in portal is required like
  - Portal SPIs and APIs is needed
  - Credential vault, personalization features
  - This can still be done when REST APIs etc are used,..but typically using local Java yields higher developer productivity
- The portlets are very lightweight for example hello world, feed readers, AJAX heavy portlets
- It is not worth the admin overhead and network latency

WSRP thumb rule …

… keep everything local until there is a reason to remote

- *simply stating everything is remote, looks nice on diagrams but is not cost effective nor best practice*

Portlet Coding Best Practices

Portlet Coding Best Practices

- Do not use instance variables
  - Portlets like servlets are singletons
  - Instance variables can create collisions across users and requests
- Pass data to the view (jsp) as bean in request object
  - Once request is complete, data gets out of scope and is cleaned up
  - Accessing bean properties is much cleaner
- Adopt good code documentation habits
- Follow JSF best practices for JSF portlets (covered in a separate section here)
- Categorize portlet state early in the design phase

Portlet Coding Best Practices

- Minimize navigational state information
  - Navigational state is stored in url
  - url length has limits
  - If navigational state is larger that a specific value, it is stored in session instead of url, thus leading to a loss of bookmark ability
  - The threshold can be customized in the StateManagerService.properties via the com.ibm.portal.state.keymanager.renderparameters.threshold constant.
- Internationalize portlets using resource bundles

Portlet Coding Best Practices

- Declare J2EE roles in portlet applications only if absolutely necessary
  - J2EE roles are separate and must not be managed by portal
- Always provide version information for portlet applications
  - Declare the version in the META-INF/MANIFEST.MF using the Implementation-Version attribute
- Use P3P user profile attributes whenever possible
- Use url encoding for resources inside portlet application WAR file

Portlet Coding Best Practices

- Do not spawn unmanaged threads from portlets
- Prepare portlets for parallel rendering
  - If taking too much time, portal may cancel portlet rendering and IOException will result. Portlet should be ready to act accordingly
- Use http GET and POST correctly
  - Render phase should provide a re-playable markup generation
  - Form submissions that change state should be handled using ActionURLs
- Use IBM extension markup property with care
  - Some portals may not provide it

Portlet Coding Best Practices

- Do not have same names for portlets and servlets within a web application
- Lookup portlet services in portlet init method
  - JNDI calls are quite expensive

Portlet JSP Best Practices

Portlet JSP Best Practices

- Include only HTML fragments in portlet jsps
  - Portlets contribute markup to a larger portal page
- Portlet jsp html fragments should be well-formed to avoid unbalanced pages
- Design the view to fit on a page with other portlets also
  - A large portlet jsp may squeeze other portlets off the page
- Use java style comments instead of HTML
  - Java code including comments are removed from rendered source
  - Thus reduces document size leading to optimization of network data

Portlet JSP Best Practices

- Use portal style classes instead of specific style-oriented attributes
- Make pages fully accessible even for users with disabilities
- URIs, HTML element name attributes, and javascript resources must be namespace encoded
  - Since many portlets exist on same page, avoid name collisions
- Prefer javascripts libraries like dojo to javascript
- Be careful with the usage of pop-ups

Portlet JSP Best Practices

- Use IFRAMES with caution
  - Undermine the portlet principle by tunneling portlet apis

Portlet Packaging Best Practices

Portlet Packaging Best Practices

- Make common functions available externally to portlets
  - Collect common classes in a jar and place it in *PortalServer/shared/app* directory
- Group together portlets operating on same backend, into one portlet application
  - Share configuration settings like server name etc

Portlet Configuration Data Best Practices

Portlet Configuration Data Best Practices

- Use portlet initialization parameters for storing static information
  - Not meant to be changed by administrators or users
- Use read-only portlet preferences for storing configuration data
  - Can only be changed by administrators in config mode
- Use writeable portlet preferences for storing customization data
  - Can be changed by users in edit mode

Portlet Configuration Data Best Practices

- Write persistent data only in action phase
  - Render phase must be re-playable to allow bookmarking and back-button feature
- Use string arrays for preference lists

Portlet Session Management Best Practices

Portlet Session Management Best Practices

- Limit the use of portlet session for storing portlet state information
  - Session management incurs cpu cycles and memory heap
  - User specific data and data that cannot be created from any other means can be put in portlet session
- Do not rely on portlet session if the portlet is to allow anonymous access
  - Unauthenticated user does not have a logout action, so the session does not invalidate until it times out, leading to memory overhead
- Always request an existing portlet session

Portlet Session Management Best Practices

- Prevent the jsp from generating temporary sessions
  - Add the JSP page directive <%@pagesession=”false”%>
- Be aware of session timeouts
  - Each portlet on a portal page may come from a different portlet application which will have different portlet session for each portlet
- Use attribute prefixing for global session scope
  - Several entities of the same portlet on the same page may result in over-writing
  - Provide a read-only portlet preference entry called session-prefix that the administrator can set in the config mode.

Portlet Session Management Best Practices

- Use no prefixing only if the data needs to be shared among all portlet application entities of one Web application.
- Use application prefixing for sharing data among all portlet entities in one portlet application.
- Use application and config param prefixing for sharing data in a portlet application, but not between different entities of the same portlet.

Portlet Internationalization Best Practices

Portlet Internationalization Best Practices

- All displayable portlet strings should be fetched from resource bundles
- All view JSPs should be language-independent.
- Be sensitive to cultural-specific formatting
  - Use locale specific in the portlet request PortletRequest.getLocale
- Define preferences names as reference names to the resource bundle
- Define the supported locales in the deployment descriptor.

Resource Serving Best Practices

Resource Serving Best Practices

- Parameters set on a resource URL are not render parameters
  - Last only for the current serveResource request
- setCacheability method on the ResourceURL should be used to improve cache-ability of markup being returned
  - It indicates that only parts of the overall state are needed via the cache level parameter
  - Results in an increased likelihood of a subsequent browser access being served from a browser/web cache

Resource Serving Best Practices

- Portal application is not involved in serving portlet fragments via serve resource so following are prohibited during resource serving
  - Eventing or shared render params
  - Setting new render params, portlet mode or window state
  - Issuing redirects
  - Changing application scoped session state
- Serving static resources like images, pdf, that are shared across portlet is not advisable to ensure portal can optimize caching of such resources
  - Resource url has the portlet ID encoded so link to same resource results in different urls

Portlet Performance Best Practices

Portlet Performance Best Practices

- Do not spawn threads
  - Since portlets are multi-threaded, spawning threads may lead to synchronization problems
- Do not use threads to access J2EE resources
  - Some OS have limited threads in process thread pools
- Avoid synchronization methods
  - May slow down portal
  - May cause deadlocks
- Avoid long running loops to ensure fast portlets

Portlet Performance Best Practices

- Use JSPs instead of XML/XSLT
  - JSPs are more efficient
  - XML/XSLT apply stylesheets on every request
- Use caching as much as possible
  - For static content valid for a period of time, enable caching
  - Define default expiration time in portlet deployment descriptor

Portlet Security Best Practices

Portlet Security Best Practices

- Use the credential vault to store sensitive data
  - Enable single sign-on to avoid multiple login challenges
- Appropriately handle data which is passed to the client
  - Consider encryption and securing senstive information
- Enable single sign-on as appropriate

Portlet Documentation Best Practices

Portlet Documentation Best Practices

- Document all context parameters and configuration parameters
  - Under portal administration, they are prone to editing and deletion
- Document how the portlet uses caching
  - Allows for portlet caching optimization
- Document portlet session requirements
  - If a valid session is required for portlet, instruct administrator not to place it on an anonymous page

JSF Portlets
Best Practices

Hosted on the public wiki

- Get access to the latest doc on our public wiki here

  http://www.ibm.com/developerworks/wikis/download/attachments/115999431/JSF_Portlet_Development_Tips_and_Suggestions.pdf?version=1

Useful Links

- RAD v8 Infocenter
  - http://publib.boulder.ibm.com/infocenter/radhelp/v8/index.jsp
- WebSphere Portal Infocenter links
  - http://www.ibm.com/developerworks/websphere/zones/portal/proddoc/index.html
- RAD Portal Tools Wiki
  - https://www.ibm.com/developerworks/wikis/display/rad/Portal+tools
- RAD Wiki
  - https://www.ibm.com/developerworks/wikis/display/rad/Home
- RAD Performance Tips
  - https://www.ibm.com/developerworks/wikis/download/attachments/113607155/radtipsv754.pdf
- Binary Modules
  - http://www.ibm.com/developerworks/rational/library/07/0619_karasiuk_sholl/

Thank You