

Extend WTP Server Tools for your application server

Tim deBoer

deboer@ca.ibm.com

Gorkem Ercan

gercan@acm.org

What is the Eclipse Web Tools Platform?

- Top-level project at www.eclipse.org
- Provides tools to build applications for standards-based Web and Java runtime environments
- Consists of two subprojects:
 - Web Standard Tools (WST)
 - HTML, XML, ...
 - J2EE Standard Tools (JST)
 - JSP, EJB, ...
- <http://www.eclipse.org/webtools/index.html>

Where does Server Tools fit in?

- Sub-component of WTP
- Server Tools provides support for:
 - Targeting applications to a specific server
 - Adding & removing projects from servers
 - Publishing applications to a server
 - Starting & stopping servers
 - Implementations for specific servers:
 - Tomcat, JBoss, ...

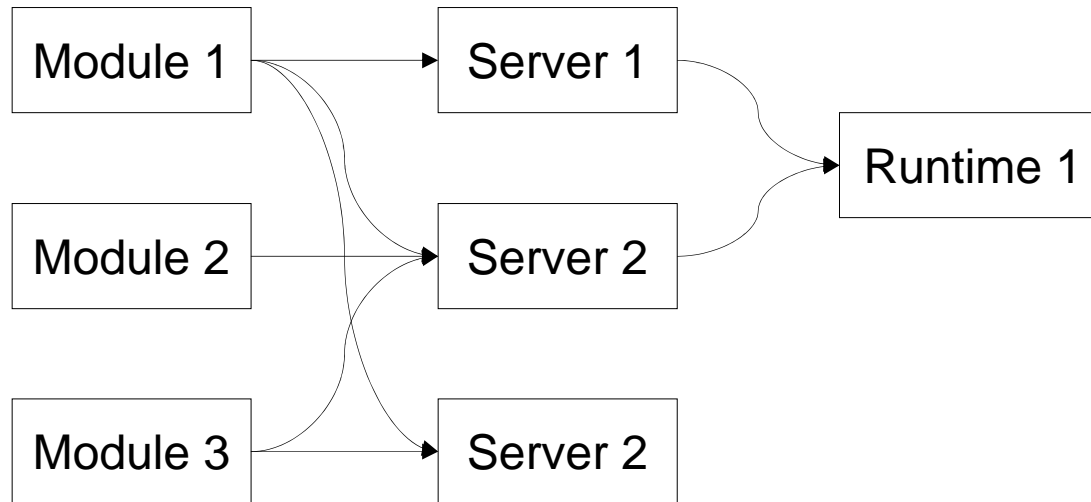
Server Tools Components

- The server tools framework supports any server, not just J2EE
- Support in both of the WTP subprojects:
 - **wst.server**
 - Server Tools framework (.server.core)
 - Server Tools UI (.server.ui)
 - **jst.server**
 - J2EE server tools (.server.*)
 - Generic J2EE server framework (.server.generic.*)
 - Tomcat, JBoss support, ...

Users of Server Tools APIs

- Server Providers
 - Add support for additional servers
 - E.g. Tomcat, JBoss
- Module Providers
 - Add additional module types and Run on Server support
 - E.g. J2EE Tools
- Client App Providers
 - Provide clients for Run on Server
 - E.g. Web browser
- Client Users
 - Use API to configure and launch servers, check runtime target, etc.
 - E.g. Web Services, DD editors

Model Overview



Modules

- A module is content that can be deployed to a server
- Typically a project or folder (e.g. Web module) within the workspace, but can consist of anything

- Extension Points:
 - moduleTypes
 - Define a new type of module
 - moduleFactories
 - Provide factory for creating and discovering modules of a specific type
 - Provides module delegates with a specified interface

Runtimes

- A runtime is an installed server on the local hard-drive
 - Executables, Jar files, etc.
 - Used for build-time compilation, validation

- Extension points:
 - runtimeTypes
 - Define a new type of runtime and delegate class
 - runtimeLocator
 - Automatically locate new runtimes on disk
 - runtimeTargetHandler
 - Change what happens when a project (containing modules) is targeted to a particular runtime
 - Modify classpath, validation, etc.

Servers

- A server is an instance of (handle to) a real server
 - Add & remove modules
 - Publish modules
 - Usually supports starting & stopping
- Often based on a local runtime

- Extension points:
 - `serverTypes`
 - Define a new type of server and delegate classes
 - Handles publishing, starting & stopping server, etc.

Run on Server support

- Run > Run on Server menu item allows users to quickly choose/create a server and run module
 - Allows user to choose or create a server
 - Starts server, publish
 - Launches client application (e.g. Web browser)
- To enable on a selection:
 - Adapt object to ILaunchable to make Run menu appear (via Eclipse debug support)
 - ModuleArtifactAdapter extension point provides enablement support
 - Adapt object to IModuleArtifact
- Each server provides support via launchableAdapter ext. point
- Clients (e.g. Web browser) can add support via clients ext. point

UI Support

- Provided by org.eclipse.wst.server.ui
- Servers view for creating and configuring servers
- Preferences and property pages, etc.

- Extension points:
 - images
 - Provide images for runtimes, servers, etc.
 - editorPages and editorPageSections
 - Provide sections and pages for the server editor
 - wizardFragments
 - Provide pages to appear when servers are created

Generic Server Introduction

- Extension to WTP server tools
 - RuntimeTypes
 - ServerTypes
- Design has its roots from Lombok
- Community already familiar with its use
- A special server and runtime that can adjust behavior
 - Server type definition files determine behavior

Server type definition file

- XML based meta information
- Validated against an XSD
- Introduced using “*org.eclipse.jst.server.generic.core.serverdefinition*” extension
- Virtually two parts
 - Properties
 - Derived information
- Properties are variables that users provide values using server tooling UI
- Derived info is information used by the generic server to perform server tooling functionality

Server type definition file example

```
.  
.br/><property id="serverRootDirectory"  
label="Application Server Directory:"  
type="directory"  
context="runtime"  
default="/your_server_root/appservers/jboss-3.2.3" />  
.br/><start>  
  <class>org.jboss.Main</class>  
  <workingDirectory>${serverRootDirectory}/bin</workingDirectory>  
  <programArguments>-c ${serverConfig}</programArguments>  
  <vmParameters></vmParameters>  
  <classpathReference>jboss</classpathReference>  
</start>
```

Making sense of server type definition files

- `<classpath>` : define a classpath used by other elements
- `<start>`: information for starting a server(classpath,class,vmarguments, etc.)
- `<stop>`: information used for stopping a server
- `<port>`:port(s) to start server on
- `<project>`: classpath to provide when creating a project for this runtime
- `<module>`: information for each supported modules, such as publisher and type
- `<publisher>`: data used by different publishers when publishing to this server
- `<property>`:define variable data to be collected from user

Using metadata for UI

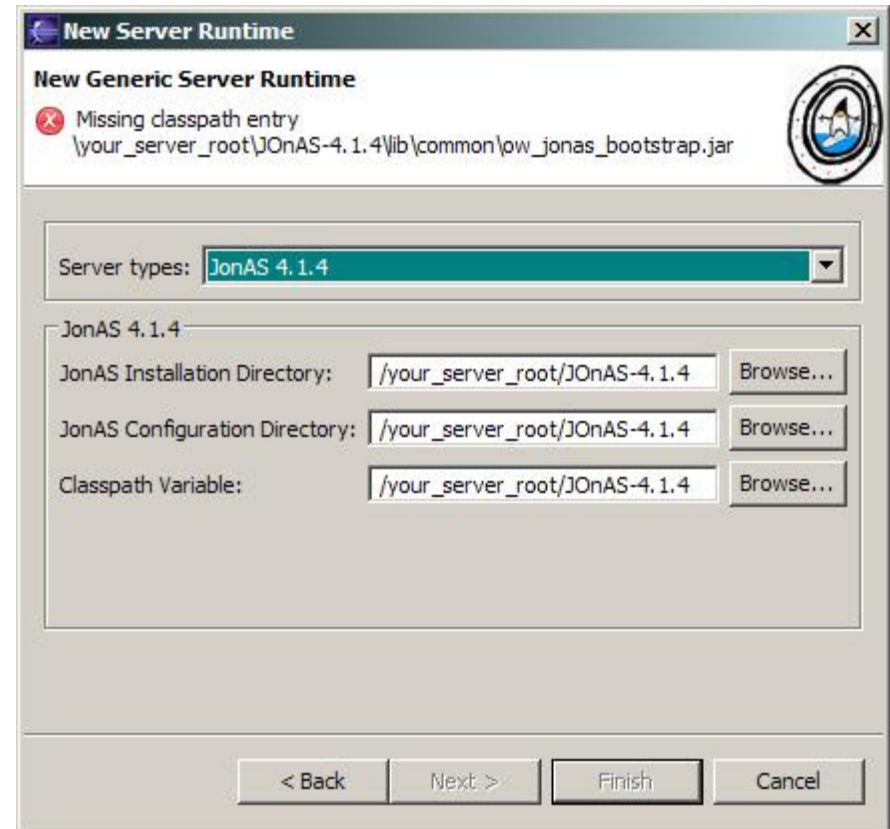
- Property type determines the type of widget used
- Context determines whether this is a server or runtime property
- Currently 4 types are supported
 - Directory
 - String
 - Boolean
 - File

UI example

```

<property id="jonasRoot"
label="JonAS Installation Directory:"
type="directory"
context="runtime"
default="/your_server_root/JOnAS-4.1.4"
/>
<property id="jonasBase"
label="JonAS Configuration Directory:"
type="directory"
context="runtime"
default="/your_server_root/JOnAS-4.1.4"
/>
<property id="classPath"
label="Classpath Variable:"
type="directory"
context="runtime"
default="/your_server_root/JOnAS-4.1.4"
/>

```



Generic publishers

- Handles publishing modules to servers
- Only part where you may need to code
- Introduced using *org.eclipse.jst.server.generic.antpublisher* extension point
- Extend GenericPublisher class
- It is optional you may choose to use an existing publisher
- ANT build file based publisher is available part of the core package
- More general publishers to come...

Shortcomings

- Server runtime discovery is not supported
- Runtime validation is limited
- No remote server support
- Incremental deployment is possible with a specialized publisher

Demo

Introducing an application server using Generic server tooling

Help Needed

- We're not done yet!
- If you are planning on using or building on WTP, we can use your help
 - Support for new server types
 - Defining and refining API
 - Testing
 - JUnit tests
 - Feedback

Thank you

Questions
&
Comments