The Equinox Project

Thomas Watson, IBM Lotus
Simon Kaegi, IBM Rational
Agenda

- Introduction to OSGi Technology and Equinox
- Server Side Equinox
- Equinox Application Model
Introduction to OSGi Technology
The Dynamic Module System For Java™ Platforms

- Specification developed by the OSGi Alliance, a non-profit organization [http://www.osgi.org](http://www.osgi.org)
- Equinox is one of several open source implementations
  - Implements several of the OSGi R4.1 specifications
    - Core Framework, HttpService, PerferenceService, EventAdmin …
    - The runtime for Eclipse which gives Eclipse its power
      - extensible, modular, dynamic …
- Benefits of OSGi technology
  - Avoids Jar Archive (JAR) file hell, supports multiple versions
  - Reuse of common components
  - Simplifies multi-team projects with componentization
  - Extensive tooling support in Eclipse
  - Many providers of the core technology
  - High adoption rate, several key open source implementations
Introduction to OSGi Technology
The Dynamic Module System For Java™ Platforms

- The Framework is split up into different layers
  - *Execution Environment* – the VM
  - *Module Layer* – Module system for the Java Platform
  - *Lifecycle Layer* – Dynamic support
  - *Service Layer* - Service orientated
The OSGi Framework – Execution Environment

- Execution Environment
  - The VM used to launch the Framework
  - The OSGi specification originated on the J2ME platform
  - Framework implementations can scale down to small devices and scale up to large server environments
The OSGi Framework – Module Layer

- Module system for the Java Platform
  - Enforces visibility rules
  - Dependency management
  - Supports versioning of *bundles*, the OSGi modules
- Sophisticated modularity framework
  - provides for class space consistency for bundles
  - supports multiple versions of packages and bundles
The OSGi Framework – Lifecycle Layer

- Lifecycle Layer provides API to manage bundles
  - Installing
  - Starting
  - Stopping
  - Updating
  - Uninstalling
- All dynamically supported at runtime
The OSGi Framework – Services Layer

- Provides an in-VM service model
  - Services can be registered and consumed inside a VM
  - Again all operations are dynamic
  - Extensive support for notification of the service lifecycle
Bundle, the unit of modularization

- **Self-describing module**
  - Roughly equivalent to a JAR file
  - Bundle manifest describes the bundle to the Framework, Identity, version, dependencies, exports etc.

- **Deployment package for the Framework**
  - Java classes, embedded JAR files, native code libraries, and resources
  - Lifecycle, bundles can use a BundleActivator to execute code when activated and deactivated
  - All operations are dynamically managed, install, update, uninstall, start and stop

- **Bundles Collaborate through**
  - Modular layer – Java *package* sharing
  - Service layer – Inter VM object sharing
  - Others – Eclipse Extension Registry, Spring-OSGi, Declarative Services etc.
Hello and Goodbye World Example
What is it?

- “Enabling technology” for using the Eclipse runtime in server-side applications.

New Features in Europa

1) Servletbridge
   - use the OSGi / Eclipse runtime inside of Java Enterprise Servers.

2) Replacement for Tomcat / Eclipse Hybrid
   - avoid some tough class loading problems

3) Enhanced OSGi Http Service Support
   - declarative support for the Servlet API and Java ServerPages
Servletbridge

Key benefits:
- Allows use of OSGi technology for web application development while leveraging existing server-side infrastructure.
- “One” component framework for client and server.

Production Ready:
- System portability verified across a wide variety of application servers and Java VMs
- Tomcat (4.1, 5.5)
- BEA Weblogic (8.1, 9.2)
- IBM Websphere (6.0, 6.1)
- Sun JRE (4.0)
- BEA JRockit (1.4, 5.0)
- IBM J9 (1.4, 5.0)
- JBoss AS (4.0)
- Mortbay Jetty (5.1, 6.0, 6.1)
- Oracle OC4J 10g (10.1.3)
- Sun JRE (1.4, 5.0)
Replacement for Tomcat / Eclipse Hybrid

- Previously developed in the Platform for Eclipse Help / User Assistance
  - Based on a customized version of Tomcat 3.2. Introduced dependencies on many older 3rd party libraries and required maintenance.
  - Never really meshed nicely with OSGi modularity support.
  - Common desire from other projects for Servlet application support

- For Europa the Platform now uses a Jetty based implementation of the OSGi Http Service.
  - Change happened late in the Eclipse 3.3 development cycle. Tomcat hybrid was still shipped in Europa (but has been removed in 3.4)
  - API is no longer internal (e.g. org.osgi.service.http.HttpService) and can be used freely by other projects
Enhanced OSGi Http Service Support

OSGi Http Service API provides for code-based registration and unregistration of Servlets and resources functionally similar to web.xml

- registerServlet (String alias, Servlet s, Dictionary initparams, HttpContext c)
- registerResource (String alias, String path, HttpContext c)
- unregister (String alias)

Enhancements in the Equinox implementation:

- Declarative support via the Extension Registry
- JSP 2.0 support
- Continued work to improve the Http Service’s Servlet API support
  - File extension support for URI mappings (e.g. /*.jsp)
  - ServletConfig.getServletName()
  - ServletContext.getResourcePaths()
Http Service Extension Registry Support

org.eclipse.equinox.http.registry

“Declarative” alternative to using the HttpService directly. Offers the following extension-points:

- **httpcontexts** – supports creation of a basic parameterized HttpContext or user-defined HttpContext
- **servlets** – semantically equivalent to HttpService.registerServlet(...)
- **resources** – semantically equivalent to HttpService.registerResource(...)

```
web.xml servlet declaration
<servlet>
  <servlet-name>test servlet</servlet-name>
  <servlet-class>my.TestServlet</servlet-class>
  <init-param>
    <param-name>testParam</param-name>
    <param-value>test param value</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>testservlet</servlet-name>
  <url-pattern>/test</url-pattern>
</servlet-mapping>

servlets extension
<servlet>
  <init-param>
    <param-name>servlet-name</param-name>
    <param-value>testservlet</param-value>
  </init-param>
</servlet>

<servlet>
  <init-param>
    <param-name>testParam</param-name>
    <param-value>test param value</param-value>
  </init-param>
</servlet>
```
Java ServerPages Support

**org.eclipse.equinox.jsp.jasper**

```java
<< public JspServlet(Bundle bundle, String bundleResourcePath, String alias) >>
```

- Used with the Http Service’s “registerServlet”.
- Provides full JSP 2.0 and tag library support

**org.eclipse.equinox.jsp.jasper.registry**

```xml
<extension
    point="org.eclipse.equinox.http.registry.servlets">
    <servlet
        alias="/myPath/*.jsp"
        class="org.eclipse.equinox.jsp.jasper.registry.JSPFactory:/bundlePath"/>
</extension>
```

Server Side Example
Hello and Goodbye World
Equinox Application Model

- Equinox Application Container
  - Based on the OSGi Application Admin Service Specification
  - Registers OSGi services for each application installed
    - ApplicationDescriptor represent each installed application
    - ApplicationHandle represent each instance of a running application
  - Services control the applications
    - Launch, Destroy, Schedule, Lock etc.
  - Manage multiple applications at the same time
    - Allow multiple agents to control
    - Locally, remotely etc.

- Eclipse Applications
  - Defined by extensions to org.eclipse.core.runtime.applications
  - Declare ID, Name, Cardinality, Thread requirements, Application class etc.
  - Many types of applications (UI, headless, server, client etc.)
Equinox Application Model

- Equinox Application Container
  - Applications extension point
  - registers
  - contributes
  - OSGi Service Registry
  - creates
  - Application Bundle
    - Application extension
      - plugin.xml
        - ID
        - Name
        - Cardinality
        - Thread
        - Application Class
  - launches
  - destroys
  - Administrator

The Equinox Project © Copyright 2007 IBM Corp.; made available under Creative Commons Att. Nc Nd 2.5 license
Hello World Application
Equinox Application Model - Conclusion

- Standards based application container
  - Application Admin Service specification
- Backwards compatible with old eclipse applications
  - IPlatformRunnable still supported (replaced by IApplication)
  - Supports applications that must be singletons
- Greater flexibility
  - Allow applications to run on threads other than main
  - Allow multiple instances of applications to run at the same time.
Full Demo
More Information

Project hub:  
http://www.eclipse.org/equinox/

Newsgroup:  
news://news.eclipse.org/eclipse.technology.equinox

Dev Mailing List:  
equinox-dev@eclipse.org
Legal Notices

- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both

- Other company, product, or service names may be trademarks or service marks of others