Modular web applications based on OSGi

Jochen Hiller
March, 18th 2008
The OSGi HttpService …

• is lightweight
• is restricted to Servlet Spec 2.1
  - no filters, no listeners, no welcome-files
  - no JSP support
• requires registration at HttpService
  - servlets / resources
  - programmatically
  - via extension point if using Equinox
• Equinox provides techniques to add missing functionality
  - see org.eclipse.equinox.http.helper classes from Simon Kaegi
  - but: complex, additional effort required
A solution: “Web Application Service”

- The `web.xml` provides all configuration information

- Parse `web.xml` to …
  - register all servlets and filters from contributing bundle
  - support specified welcome-files
  - add JSP support by default
  - add all mime-types, support default mime-types

- Map web application to a context (alias in HttpService terms)
- Resource lookup from contributing bundle
- Binds to all OSGi HttpServices

- The technique: register wrapper servlets
- See [https://bugs.eclipse.org/bugs/show_bug.cgi?id=162132](https://bugs.eclipse.org/bugs/show_bug.cgi?id=162132)
Web Application Service: The Service API

```java
package org.eclipse.equinox.webapp.service;

public interface WebAppService {
    public Object registerWebApp(
        String alias,
        Bundle bundle,
        String bundleResourcePath,
        String webXml,
        Dictionary options)
        throws WebContextException;
    public void unregisterWebApp(Object handle);
}
```
Web Application Service: As Extension point

<!– web application service as extension point -->
<extension
  id="webapp"
  name="DemoWebApp"
  point="org.eclipse.equinox.webapp.registry.webapp">
  <webapp
    alias="/"
    path="/WebContent">
  </webapp>
</extension>
Web Application Service: As Extender pattern

• Follows Extender Pattern as proposed by OSGi
• Bundle Listener observes all bundles coming / leaving the platform
• Trigger file is `/WEB-INF/web.xml`
• Context to register is bundle symbolic name
  • May be configured through `/WEB-INF/osgi-web.xml`
What are the benefits?

• Easy development of JavaEE based web applications based on OSGi
• Simplified deployment of existing web applications (WAR) to OSGi
• OSGi can act as a **lightweight** web container
• Implementation is not dependent on OSGi runtime implementation
Demo: Deploy Tomcat examples

• Tomcat 5.5.x example applications
  ♦ servlet-examples
  ♦ jsp-examples

• Changes required:
  ♦ Create `/META-INF/MANIFEST.MF`
  ♦ Added `/WEB-INF/classes` and `/WEB-INF/lib/* .jar` to bundle classpath
  ♦ May be automated using `bnd` tool from Peter Kriens
Limitations, Plans

• And what is missing?
   Taglib support is missing
   Listeners not yet supported
   Declarative security not supported
   Implementation under development, API may change
   Documentation, Tutorials
   More testing (e.g. compatibility with servlet bridge, other OSGi implementations)

• Further ideas:
   Equinox specific framework extension to directly support loading WAR files
   Align with Enterprise OSGi and RFC 66 activities
Questions?

• Contact me:
  ✷ jo.hiller@googlemail.com

• Incubator project at:
  ✷ http://sourceforge.net/projects/sse-examples
  ✷ CVS, webapp-incubator