Equinox / OSGi on the Server

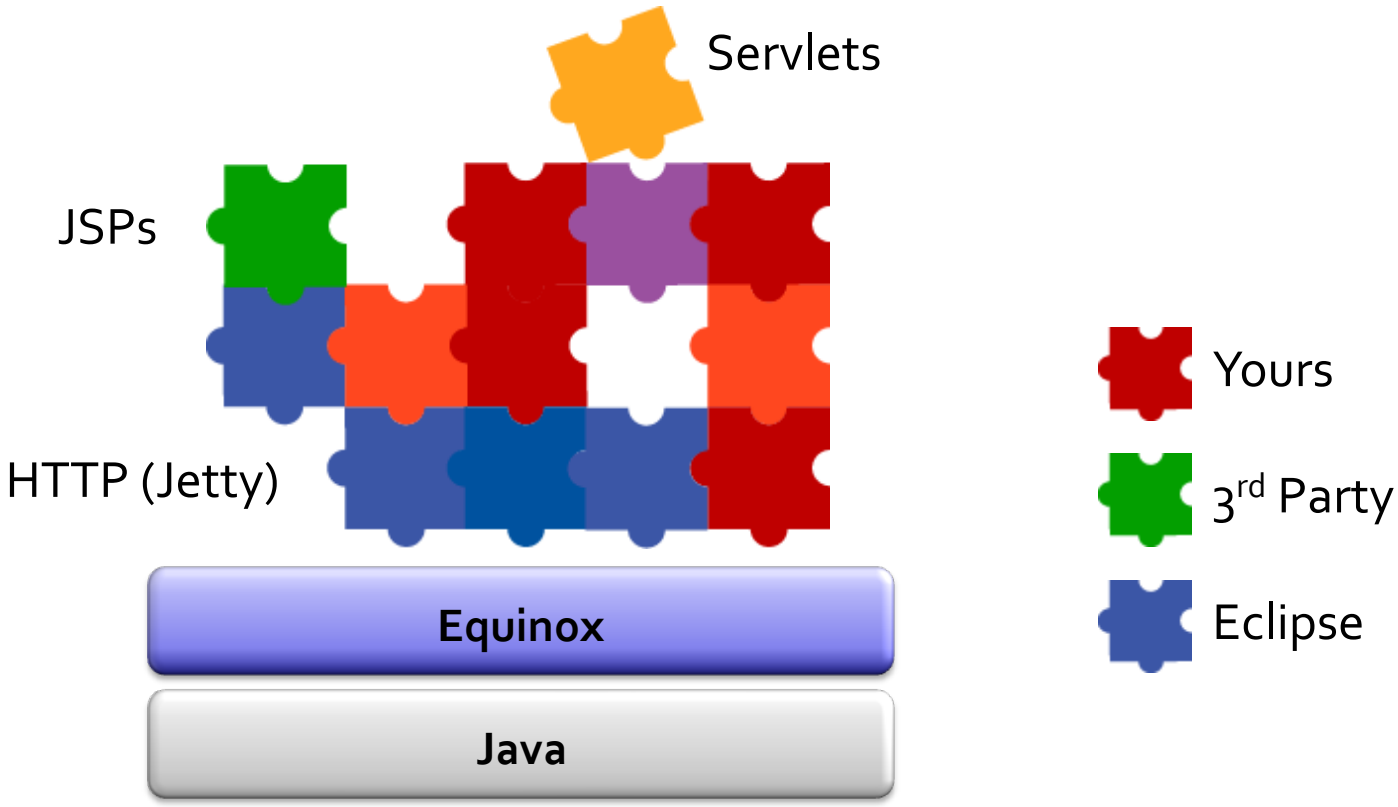
Jeff McAffer
CTO EclipseSource
RT PMC Co-Lead

3½ Ways

- Solo
- Bridged
- Native
- Implementation Detail

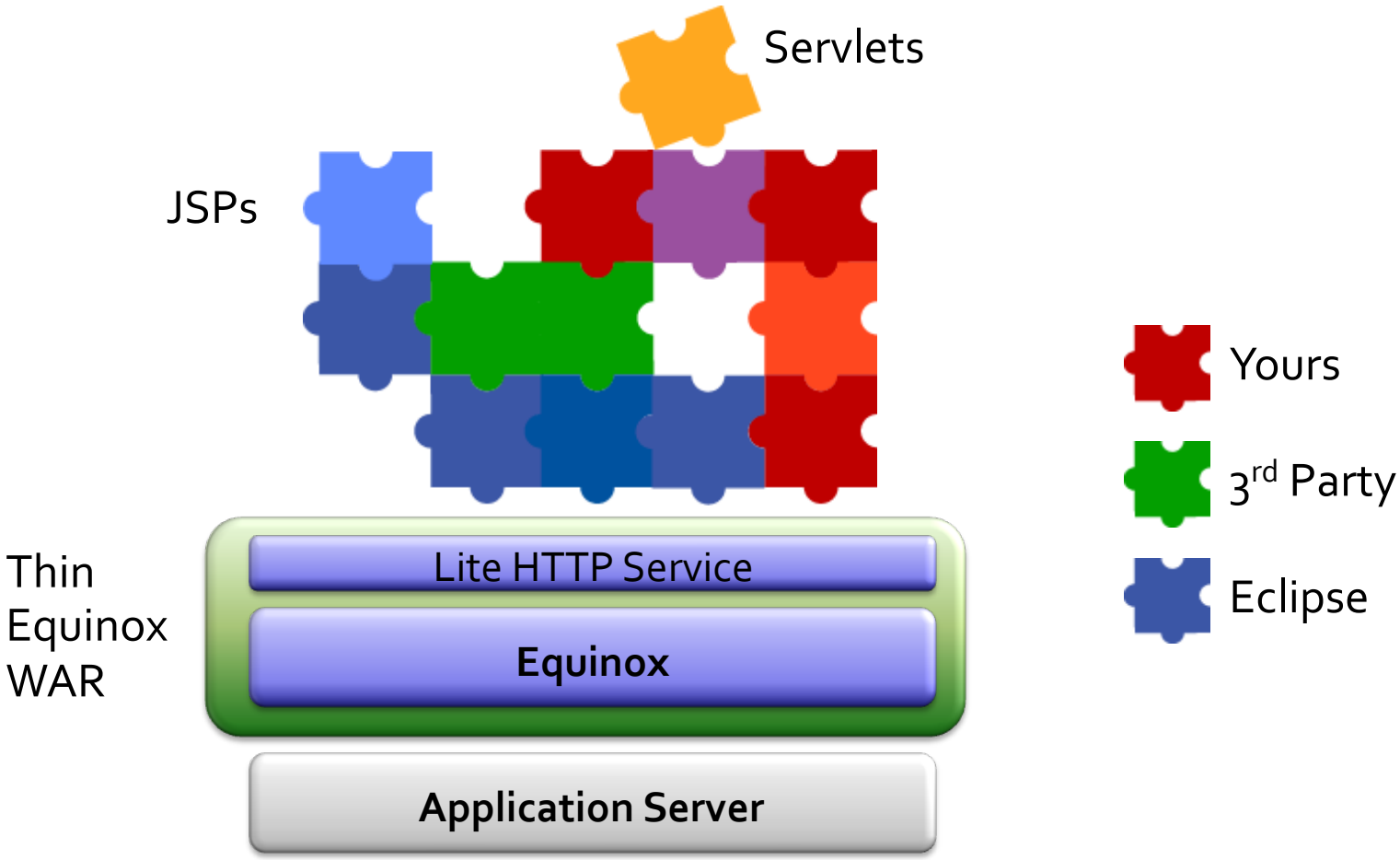
EclipseSource

Solo



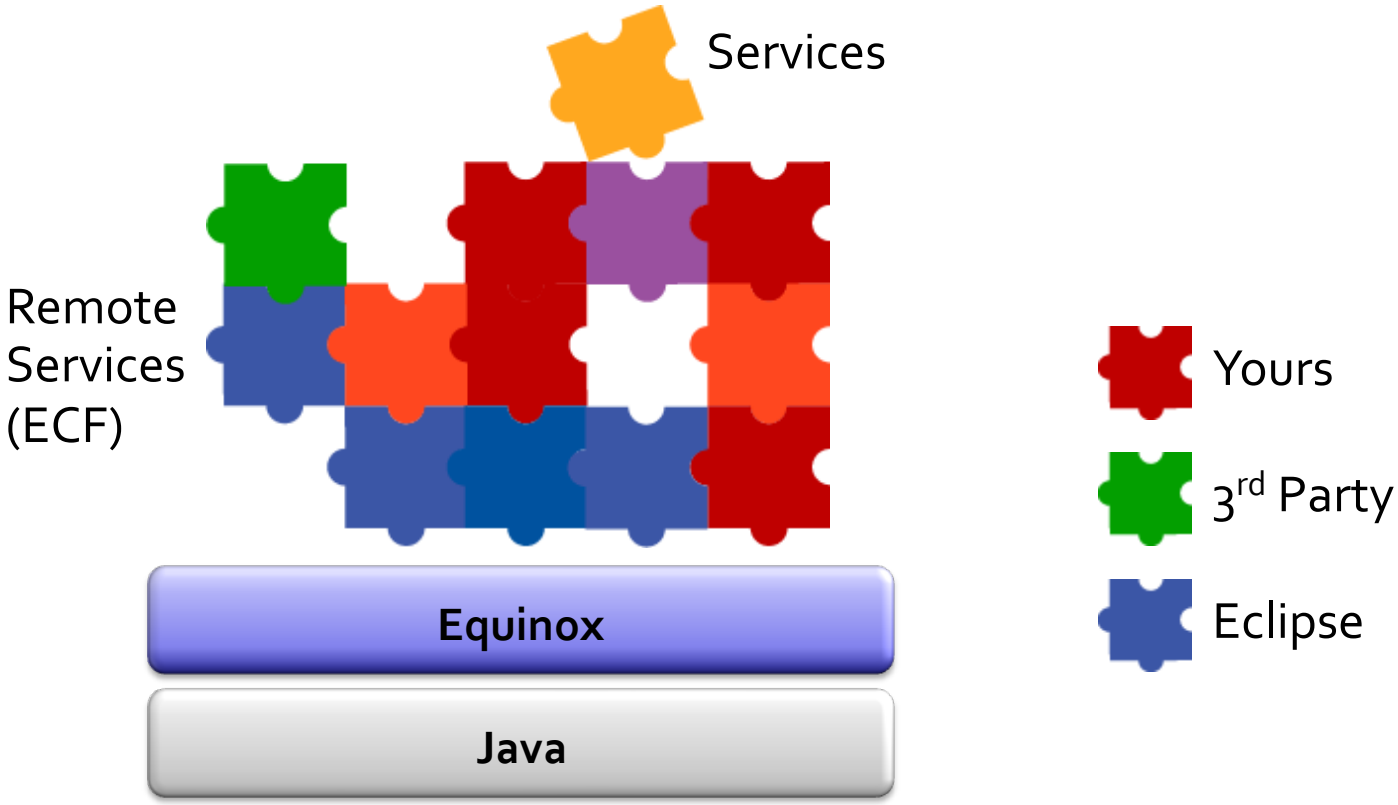
EclipseSource

Bridged



EclipseSource

Native



EclipseSource

Implementation Detail

- Implementation detail
- Bring all the power of OSGi modularity
- Without the programming model
- Initial adoption step prior to full acceptance

EclipseSource

HTTP Service

HTTP Service

- Basis for Solo and Bridged scenarios
- Standard is servlet 2.1 container
- Equinox supplies
  - Basic – small and simple
  - Advanced – based on Jetty
- Servlets, JSPs, static content, …

EclipseSource

Concepts

- Content—The bytes served in response to requests.
- Location—HTTP-based content is accessed via URLs.
- Context—All requests are processed in a context.

EclipseSource

HttpService

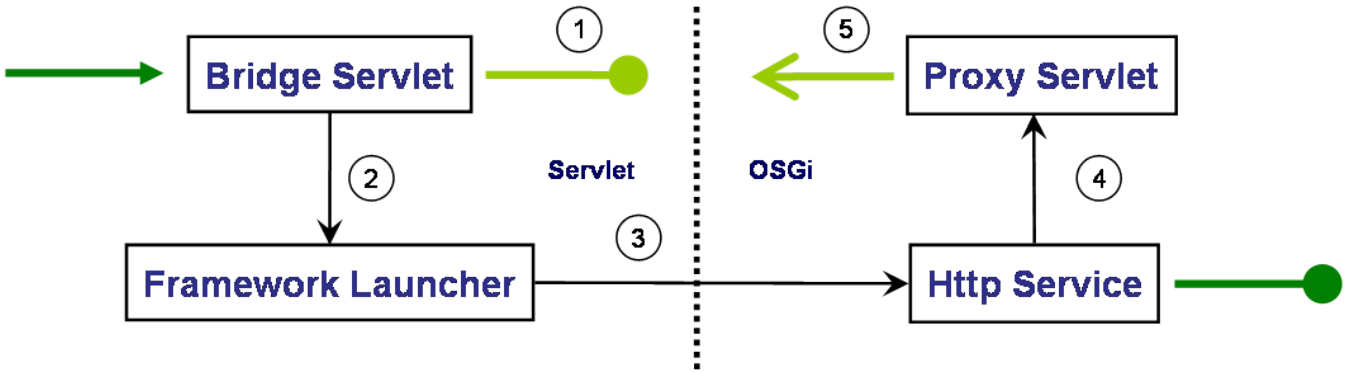
```
public interface HttpService {

  public HttpContext createDefaultHttpContext();

  public void registerResources(String alias, String name,
     HttpContext context) throws NamespaceException;

  public void registerServlet(String alias, Servlet servlet,
     Dictionary initparams, HttpContext context)
     throws ServletException, NamespaceException;

  public void unregister(String alias);

}
```

EclipseSource

EclipseSource

AppServer Integration

Bridge Request Flow



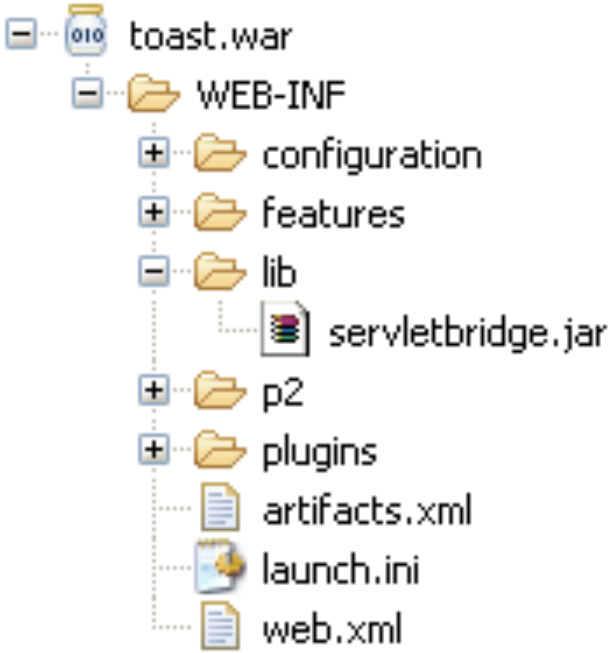
EclipseSource

Bridge Bundles

- **org.eclipse.equinox.servletbridge**—The bridge itself. Runs *under* the framework rather than *in* it. It is packaged as a bundle for consistency and to ease workflows.
- **org.eclipse.osgi**--The OSGi framework.
- **org.eclipse.equinox.http.servletbridge**—Servlet delegate within the application server to capture servlet requests.
- **org.eclipse.equinox.http.servlet**—Defines and registers an HttpService instance.
- **org.eclipse.equinox.servletbridge.extensionbundle**—Fragment that exports the packages javax.servlet, javax.servlet.http, and javax.servlet.resources from the underlying application server.

EclipseSource

WAR Contents



EclipseSource

Web.xml

```
<web-app id="WebApp">
 <servlet id="bridge">
  <servlet-name>equinoxbridgeservlet</servlet-name>
  <display-name>Equinox Bridge Servlet</display-name>
  <servlet-class>
   org.eclipse.equinox.servletbridge.BridgeServlet
  </servlet-class>
  <init-param>
   <param-name>commandline</param-name>
   <param-value>-console</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
  <servlet-name>equinoxbridgeservlet</servlet-name>
  <url-pattern>/*</url-pattern>
 </servlet-mapping>
</web-app>
```

EclipseSource

EclipseSource

Remote Services

RFC 119

- Distribution
  - Remote message calls
  - Marshaling
- Discovery
  - Discover services published remotely
- Transparent and explicit messaging

EclipseSource

ECF and 119

- ECF has had remote services for some time
- 119 standardizes the markup / properties
- Current implementation based on early draft
- Shipping as part of Galileo
- Based on the notion of *containers*

EclipseSource

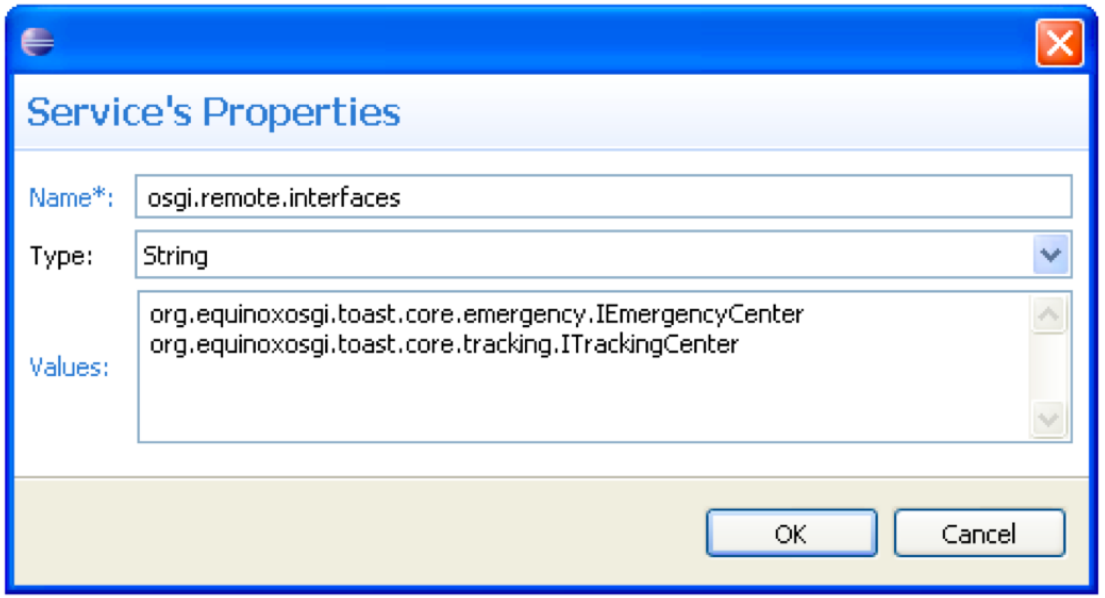
Service Host

- Hosts service
- Publishes services marked as “remote”
- Marked using service properties

```
containerManager.getContainerFactory().
    createContainer("ecf.generic.server",
        new Object[] {"ecftcp://localhost:3282/server"});
```

EclipseSource

Remote Service Properties



* Property name/structure likely to change

EclipseSource

Service Client

- Setup ECF container
- Remote services just show
- Implicit or explicit messaging

EclipseSource

Service Discovery

- SLP
- Zeroconf / Bonjour
- File-based
- Remote services appear in client service registries

EclipseSource

Coming…

EclipseSource

Web Container (RFC 66)

- "Web Application Bundle"
- Includes a "web.xml"
- Same deployment model and same features as current web applications
- ...but is a real bundle that can use OSGi Services
- Syntactic sugar? Tasty?

EclipseSource

JNDI and OSGi Integration (RFC 142)

- Current JNDI implementations use Context Class Loader for lookup.
- Clever Idiom?

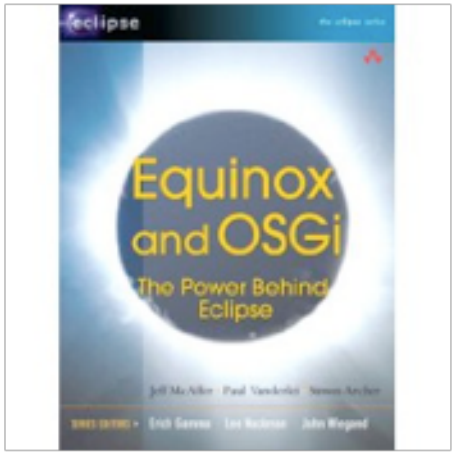
```
ClassLoader original = Thread.currentThread
    ().getContextClassLoader();
try {
    Thread.currentThread().setContextClassLoader
    (BridgeServlet.class.getClassLoader());
    // Do JNDI Calls here
} finally {
    Thread.currentThread().setContextClassLoader
    (original);
}
```

- JNDI Service will provide a more sensible "bundle" scoped lookup.

EclipseSource

Summary

Resources



OSGi and Equinox book
http://equinoxosgi.org

Toast @ Eclipse
http://wiki.eclipse.org/Toast

EclipseSource