Server-Side Eclipse

Bernd Kolb
b.kolb@kolbware.de

Martin Lippert
akquinet agile GmbH
martin.lippert@akquinet.de

Outline

Eclipse everywhere

- Old fashioned:
  - Eclipse is a nice Java-IDE
- Well established:
  - Eclipse is a well-known framework for developing Rich-Client-Applications (see Lotus Notes and many more…)
- But:
  - Most applications don’t have just a rich client
  - Some applications don’t even have a rich client

What‘s next?

- Server-Side Eclipse:
  - Use Eclipse-Equinox as platform for server-side applications

- Why?

Why?

- Modules via OSGi
  - Declared dependencies, versioning, public vs. private APIs, updating, services, …
  - Strong dynamic component model
- Building flexible architectures via Extension-Points
  - Platform-based development, component model, extensibility
- Reuse of components on Clients and Server
- Reuse of Build-In Eclipse Features
  - Adapters
  - Update
  - …

Many interested parties…

- Interested Eclipse projects…
  - Equinox
  - Rich Server Platform
  - Rich AJAX Platform
  - Eclipse Component Framework
  - Corona Project
  - …
- IBM WAS 6.1 is based on OSGi
- Spring
- Many Apache Projects (Harmony, Geronimo, …)

Different opportunities

- Pure OSGi – Application
  - The puristic way
- OSGi inside a app/web-server
  - The standard way where you need isolation for your app
  - The standard way where you have no control over the app/web-server
- App/web-server within OSGi
  - The recommended way where you have control over the app/web-server

Additional things out of the box

- Equinox Extension-Registry
  - Highly scalable extension mechanism provided by Eclipse (Extension-Points and Extensions)
- Spring
  - Standard framework for building lightweight JEE applications
- …

Pure OSGi

- Descriptor for a bundle

```
Bundle-Name: Simpleosgi Plug-in
Bundle-SymbolicName: de.kolbware.samples.simpleosgi
Bundle-Version: 1.0.0
Bundle-Activator: de.kolbware.samples.simpleosgi.Activator
Import-Package: org.osgi.framework;version="1.3.0"
```

- Implementation

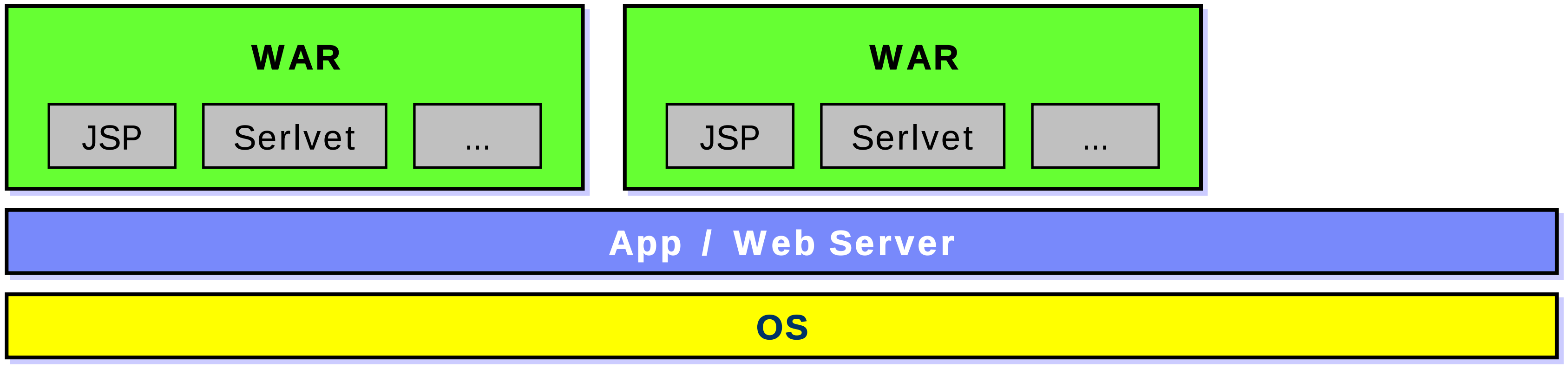
```
public class Activator implements BundleActivator {

    public void start(BundleContext context) throws Exception {
        System.out.println("Hello World!!");
    }

    public void stop(BundleContext context) throws Exception {
        System.out.println("Goodbye World!!");
    }
}
```

The App/Web Server Case

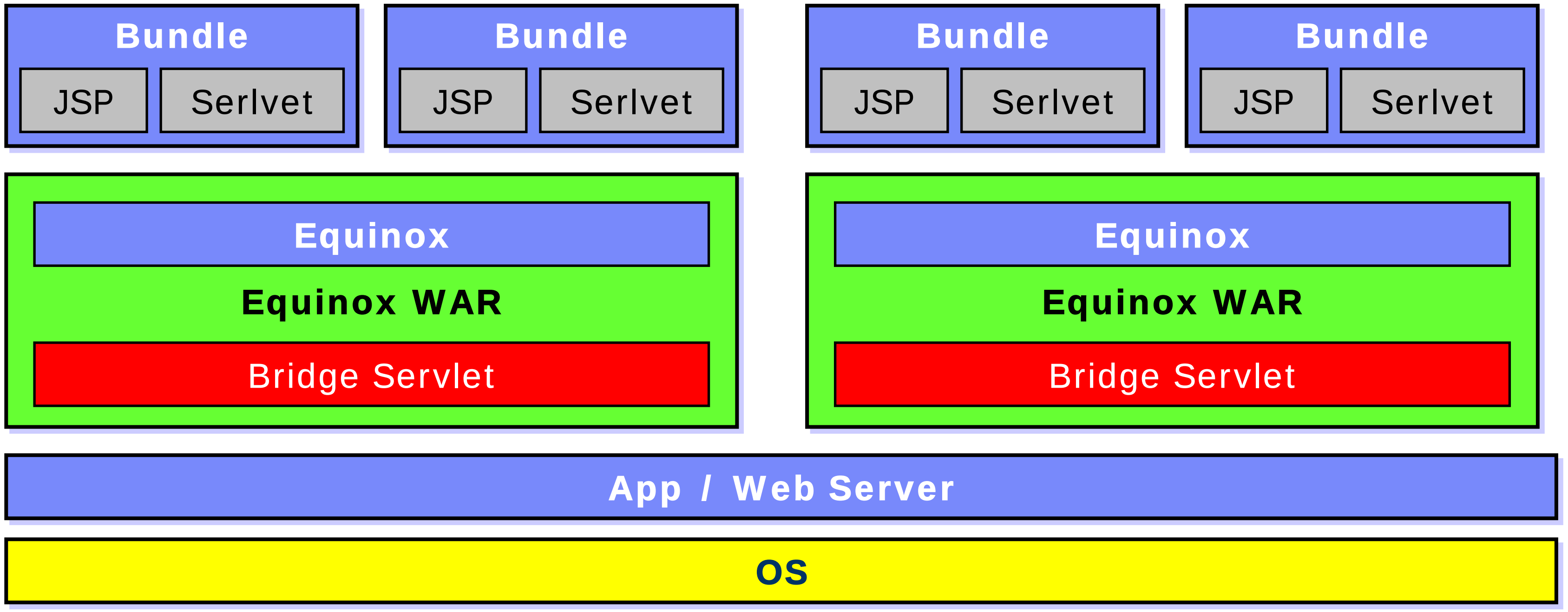
- The traditional server-side application
  - Comes as WAR file
  - Lives inside the app/web-server



OSGi inside a Web-container

- The Equinox incubator project developed a Servlet-Bridge
- The OSGi container is bundled inside a WAR-file
- The Servlet inside the Servlet-Bridge forwards the requests to your servlets
- Servlets and resources can be contributed via an extension point

OSGi inside a Web-container



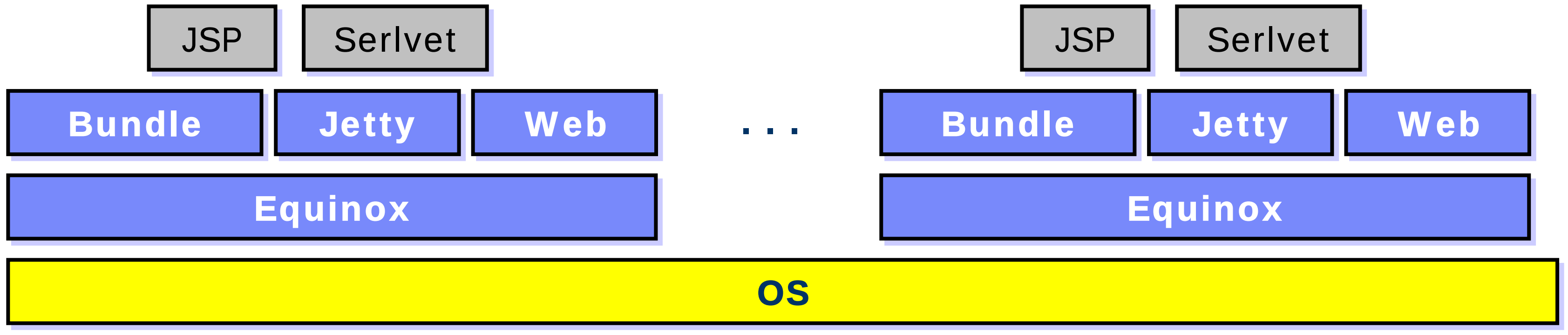
OSGi inside a Web-container

- **Demo**

Web Server in an OSGi container

- The OSGi container starts up normally
- The Web Server is wrapped into an OSGi bundle
- A third Plug-In publishes extension-points to register web-apps

Web Server in an OSGi container

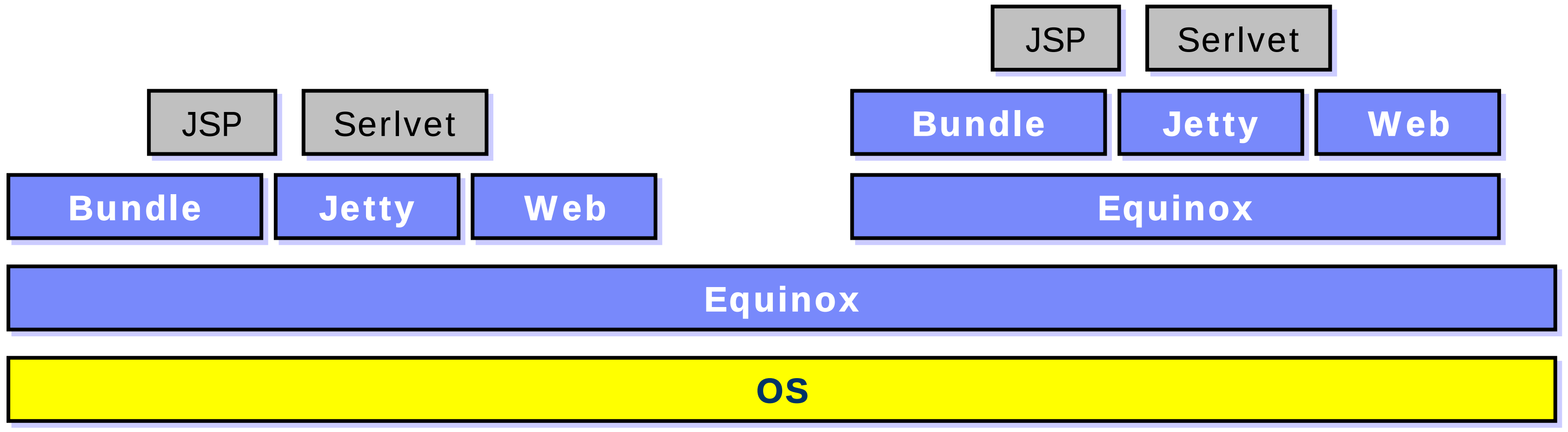


Web Server in an OSGi container

- **Demo**

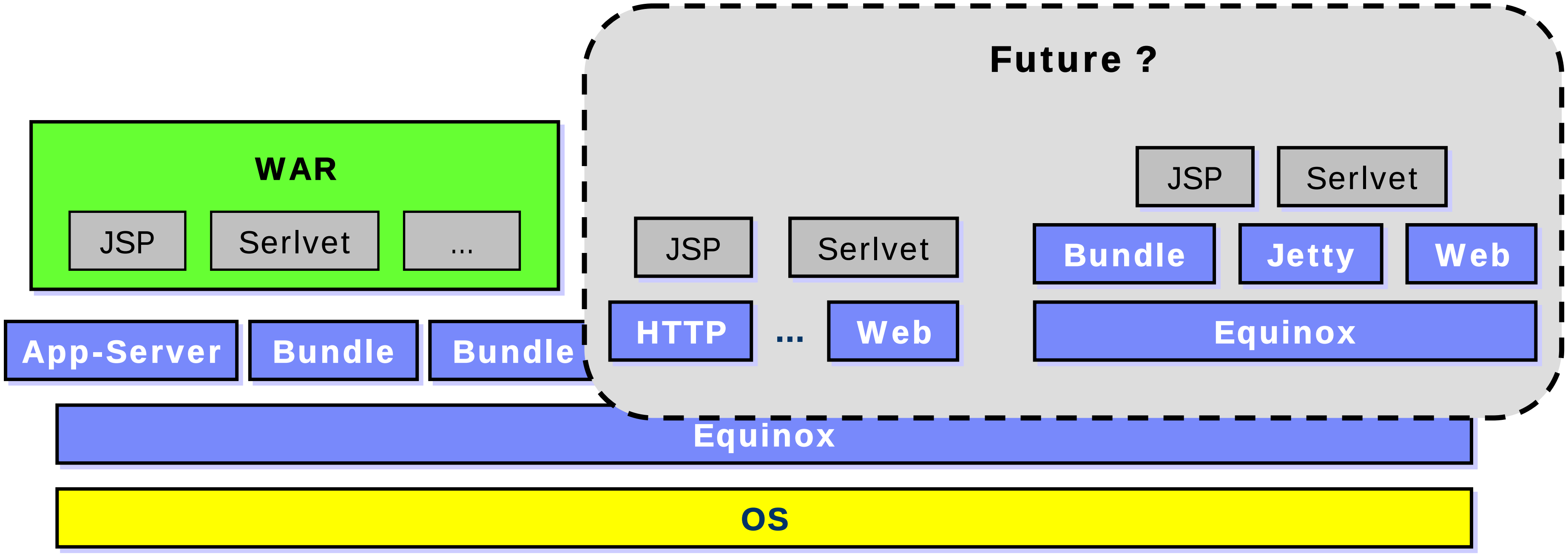
Equinox inside Equinox

- To allow application isolation Equinox can be embedded inside Equinox



The Future?

- App Server build on Equinox
- Additional App Server functionality deployed as OSGi bundles
- Allows you to combine other approaches



Out of the Box: Extension-Registry

- The famous Extension-Point-Mechanism of Eclipse can be used for server-side applications quite well
  - Extension model for server-side applications

- And think about having the same (non-ui) bundles and extensions on both sides
  - Same bundles on RCP client and middle-tier server

- Side note: Can be used even without an OSGi runtime

Out of the Box: Spring and OSGi

- Still in development
- The Spring framework is started as an OSGi Bundle
- Each bundle can define its Spring context in the META-INF directory
- OSGi-Services and Spring-Beans concept integrated
  - E.g. for inter-bundle dependency injection

Web-Server, OSGi and Spring

- As still in development not everything is working perfectly together
  - Classloading issues
- We will run the Eclipse-Platform inside Jetty using the incubator-code
- We defined a servlet which accesses a spring-service
  - → REST-Based
- **Demo**

Technical Challenges

- Classloading
  - many of the libs don‘t like the dynamic attitude of OSGi
    - Hibernate
- Limitations by the JRE
  - e.g. URLStreamHandlerFactory can only be set once
  - SecurityManager
  - …

Thank you for your attention!

- Questions are welcome!!!

- Further help and assistence:
  - Bernd Kolb: b.kolb@kolbware.de
    - Consulting, Coaching, Training
  - Martin Lippert: martin.lippert@akquinet.de
    - Consulting, Coaching, Training