Equinox OSGi: Pervasive Componentization

Thomas Watson
Equinox Development Lead
IBM Lotus

Jeff McAffer,
Eclipse RCP and Equinox Lead
IBM Rational Software

Why is Eclipse interesting?

- Extensible
- Platform independence
- Native look and feel
- GUI frameworks
- Rich ecosystem of offerings
  - Eclipse, other open source communities and products

Platform for building Platforms

What gives Eclipse its power?

- Equinox – the Eclipse runtime
- Modular
  - Building platforms requires componentization
  - Function is captured in self-describing ***bundles***
- Dynamic
  - Bundles can be installed, started, stopped, uninstalled at any time
- Standard
  - Based on the Service Platform Specification R4 from the OSGi Alliance

OSGi, Eclipse and Equinox

- OSGi Alliance
  - Produces open specifications for runtime environments
  - Traditional focus on embedded (home gateway, telematics, …)
  - Broadening scope
    - Mobile devices, desktops, enterprise and servers
  - Several open source implementations including Equinox

- Eclipse
  - Eclipse 3.0 saw the rise of Eclipse the Rich Client Platform (RCP)
  - Needed a standard, open, flexible, dynamic, modular runtime to replace the proprietary Eclipse runtime
  - Eclipse has been OSGi-based since 3.0 (3 years, 3 releases)

- Equinox
  - Eclipse OSGi implementation
  - OSGi R4 reference implementation
  - Provides consistent component story across computing environment and domains

Pervasive Componentization

Technical Details

What is the OSGi Service Platform?

- Component model for Java
- Defines ***bundles*** (typically JAR files) that contain
  - Java classes, Resources, Files, Metadata
- Bundle metadata declaratively defines
  - Java packages exported
  - Dependencies on other bundles and Java packages
  - Bundle classpath
  - Bundle lifecycle
- Framework manages dependencies and lifecycle notification
  - Explicitly supports dynamic scenarios
- Interaction through service interfaces

Bundle Metadata

Identification

```
Bundle-SymbolicName: org.eclipse.equinox.registry
Bundle-Version: 3.2.100.v20060918
Bundle-Name: Eclipse Extension Registry
Bundle-Vendor: Eclipse.org
```

Classpath

```
Bundle-ClassPath: .
```

Lifecycle

```
Bundle-Activator: org.eclipse.core.internal.registry.osgi.Activator
```

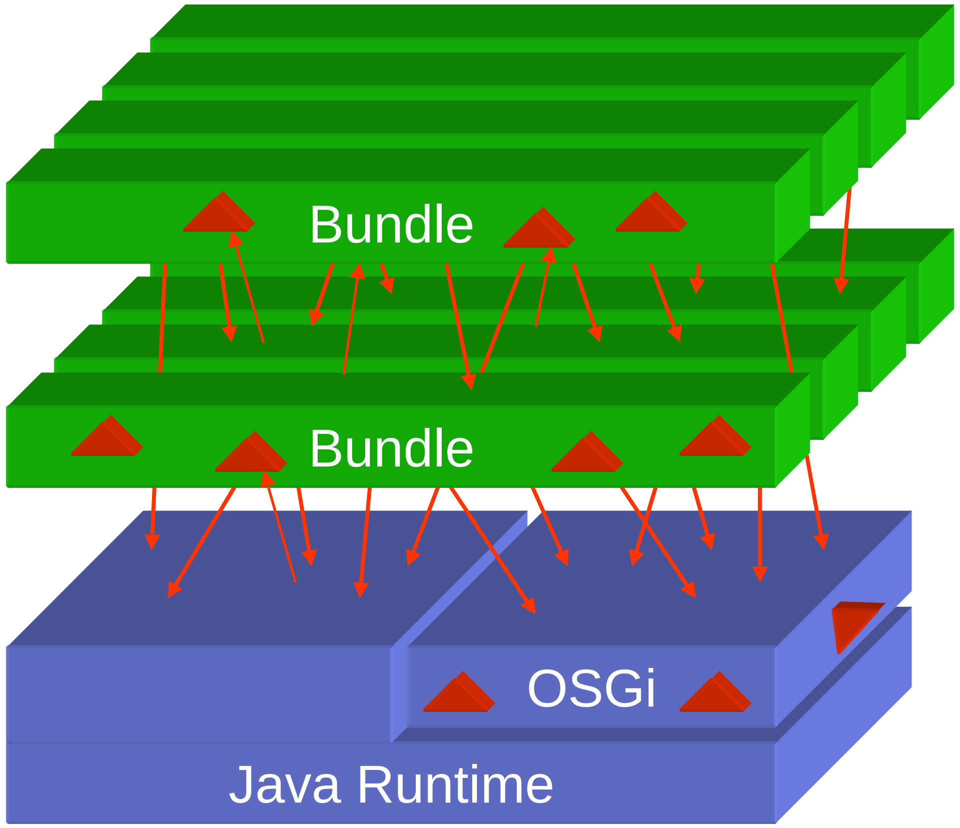
Dependencies

```
Import-Package: javax.xml.parsers,
 org.xml.sax,
 org.osgi.framework;version=1.3
Require-Bundle: org.eclipse.equinox.common;bundle-version="[3.2.0,4.0.0)"
Bundle-RequiredExecutionEnvironment: CDC-1.0/Foundation-1.0,J2SE-1.3
```

Exports

```
Export-Package: org.eclipse.equinox.registry
```

What does it look like?



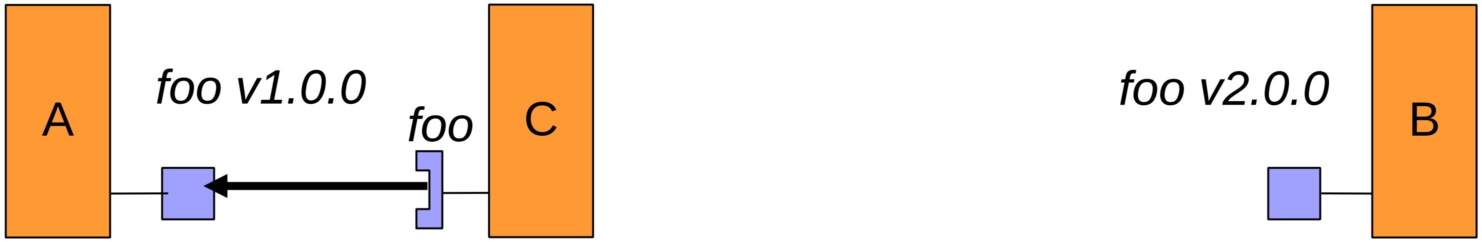
Multi-version support

- Possible to have more than one version of a shared package in memory at the same time
- General change of philosophy to the prior OSGi specifications
- Has deep impact on collaboration as well as modularity
  - For a given bundle, the service registry is implicitly partitioned according to the package versions visible to it
  - Impact on services not explored further in this presentation

```
Export-Package: foo;
 version="1.0.0"
```

```
Import-Package: foo;
 version="[1.0,2.0)"
```

```
Export-Package: foo;
 version="2.0.0"
```



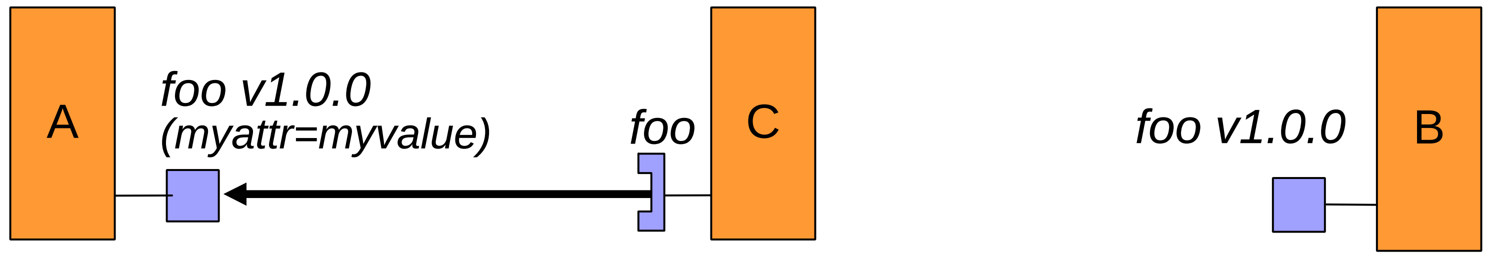
Import/Export attributes

- Exporters attach arbitrary attributes to their exports
- Importers match against these arbitrary attributes
  - Exported attributes can be mandatory
  - Mandatory attributes provide simple means to limit package visibility
  - Importers influence package selection using arbitrary attribute matching

```
Export-Package: foo;
 version="1.0.0";
 myattr="myvalue"
```

```
Import-Package: foo;
 version="1.0.0";
 myattr="myvalue"
```

```
Export-Package: foo;
 version="1.0.0"
```



Sophisticated package consistency model

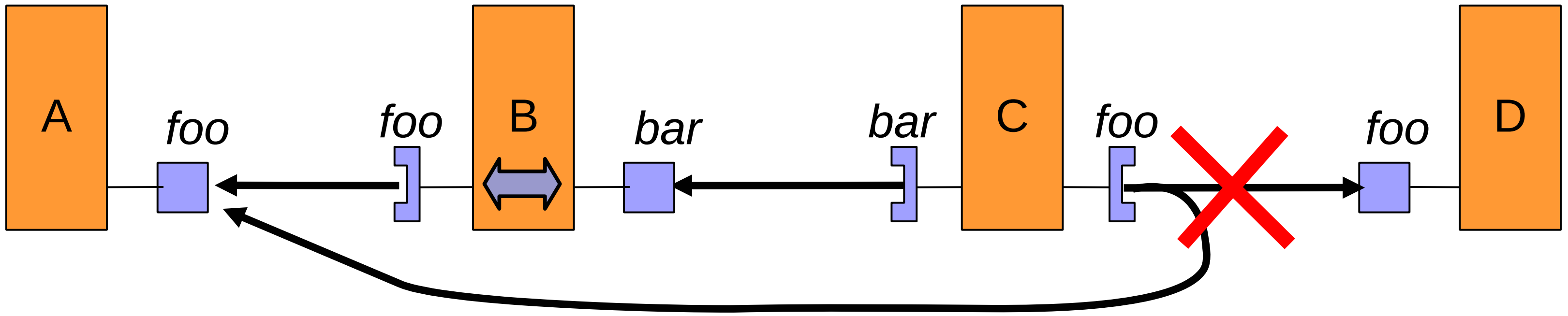
- Exporters can declare package “uses” dependencies
  - Exported packages express dependencies on imported or other exported packages, which constrain the resolve process
- The framework must ensure that importers do not violate constraints implied by “uses” dependencies

Export-Package: **foo**

Import-Package: **foo**
Export-Package: **bar**;
 uses:=“**foo**”

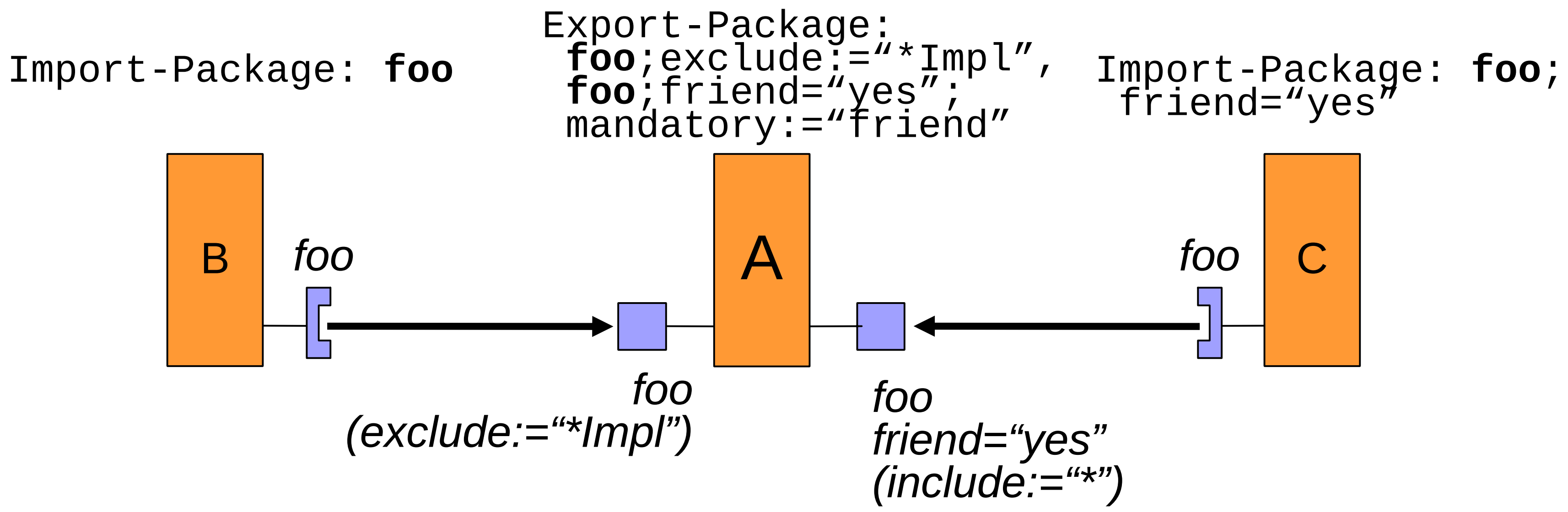
Import-Package: **foo**, **bar**

Export-Package: **foo**



Package filtering

- Exporters can declare that certain classes are included/excluded from the exported package

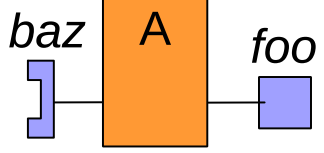

Import-Package: **foo**
Export-Package: **foo**;exclude:="*Impl", **foo**;friend="yes"; mandatory:="friend"
Import-Package: **foo**; friend="yes"
B
foo
A
foo
C
foo (exclude:="*Impl")
foo friend="yes" (include:="*")

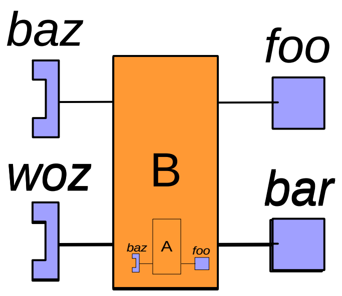
Bundle Fragments

- A special bundle that attaches to a host bundle and uses the same class loader
  - Conceptually becomes part of the host bundle

```
Bundle-SymbolicName: A
Fragment-Host: B
Export-Package: foo
Import-Package: baz
```

```
Bundle-SymbolicName: B
Export-Package: bar, foo
Import-Package: woz, baz
```



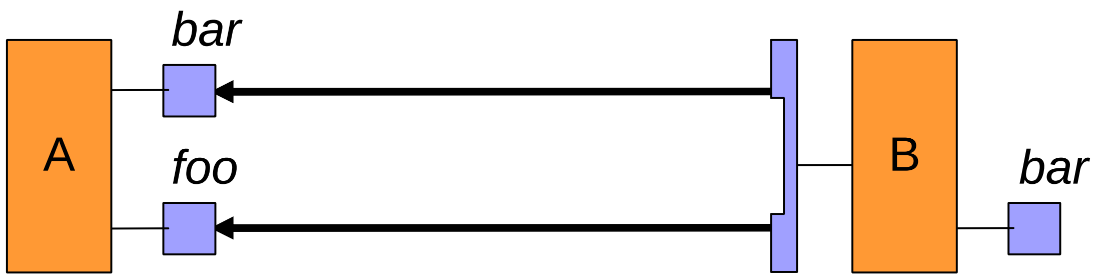


Bundle to Bundle dependencies

- Import everything that another, specific bundle exports
- Allows re-exporting and package splitting

```
Bundle-SymbolicName: A
Export-Package: bar, foo
```

```
Require-Bundle: A
Export-Package: bar
```



Proof is in the Demos

Eclipse on the Server

Eclipse as a Server Platform

- Modular, Dynamic and Flexible
  - Ideal for server side use

Eclipse

- Equinox
- Rich Ajax Platform
- Rich Server Platform – UI
- Communications Framework
- Corona
- Enterprise Component Project

Apache

- Felix
- Directory
- Cocoon
- James
- Geronimo

- Spring community investigating OSGi integration
- IBM WAS 6.1 based on Equinox
- Apache Harmony using OSGi modularity

Vol. 7

eclipse MAGAZIN

www.eclipse-magazin.de

Auf Magazin-CD:

ApacheCon Europe

Server-Side Eclipse

- In Zukunft mehr als Rich Clients
- Equinox: modulare Basis für Eclipse
- Rich Ajax Platform (RAP) – ein Interview

Großprojekt Callisto
Simultan-Release von zehn Eclipse-Projekten

Eclipse vs. Visual Studio
Konzepte und Philosophie im Vergleich

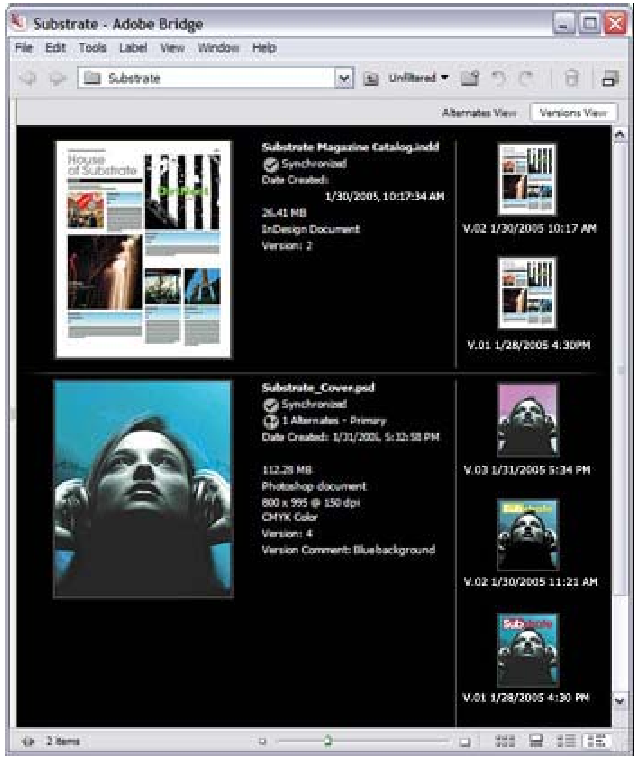
Build to last
Evolutionäre API-Entwicklung

Eclipse Communication Framework
Medium für die kollaborative Entwicklung

Datenzugriff in Eclipse
Integration des Java Persistence API

Adobe Version Cue

- Embedded client/server document management system
- Project management functionality for small workgroups
  - version control, file collaboration, streamlined reviews
- Eclipse offers
  - Multi-platform support
  - Strong, dynamic, standard component model (Equinox/OSGi)
  - Configuration management
  - Reuse components on clients and servers



Substrate Magazine Catalog.indd
Synchronized
Date Created: 1/30/2005, 10:17:34 AM
26.41 MB
InDesign Document
Version: 2

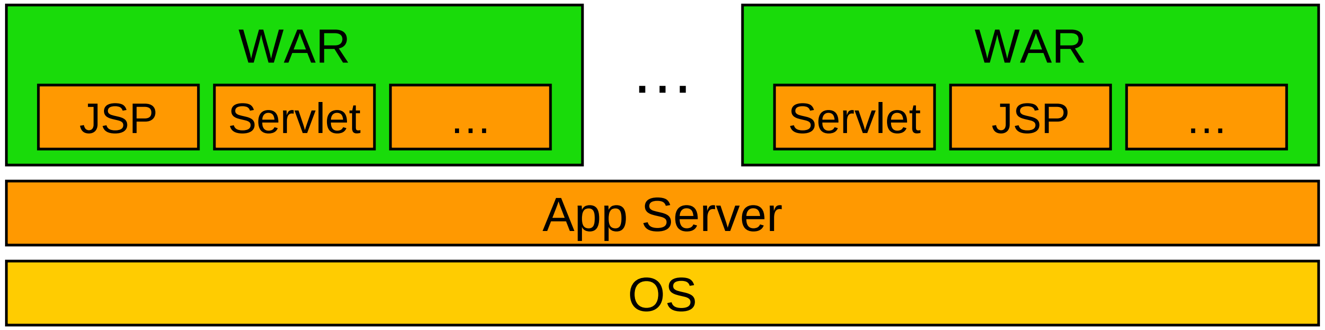
Substrate_Cover.psd
Synchronized
1 Alternates - Primary
Date Created: 1/31/2005, 5:32:58 PM
112.28 MB
Photoshop document
800 x 995 @ 150 dpi
CMYK Color
Version: 4
Version Comment: Bluebackground

Server-side Variations

- Traditional App Server
- Equinox nested in an App Server
- Raw Equinox
- Equinox nested in another Equinox
- App Server on Equinox

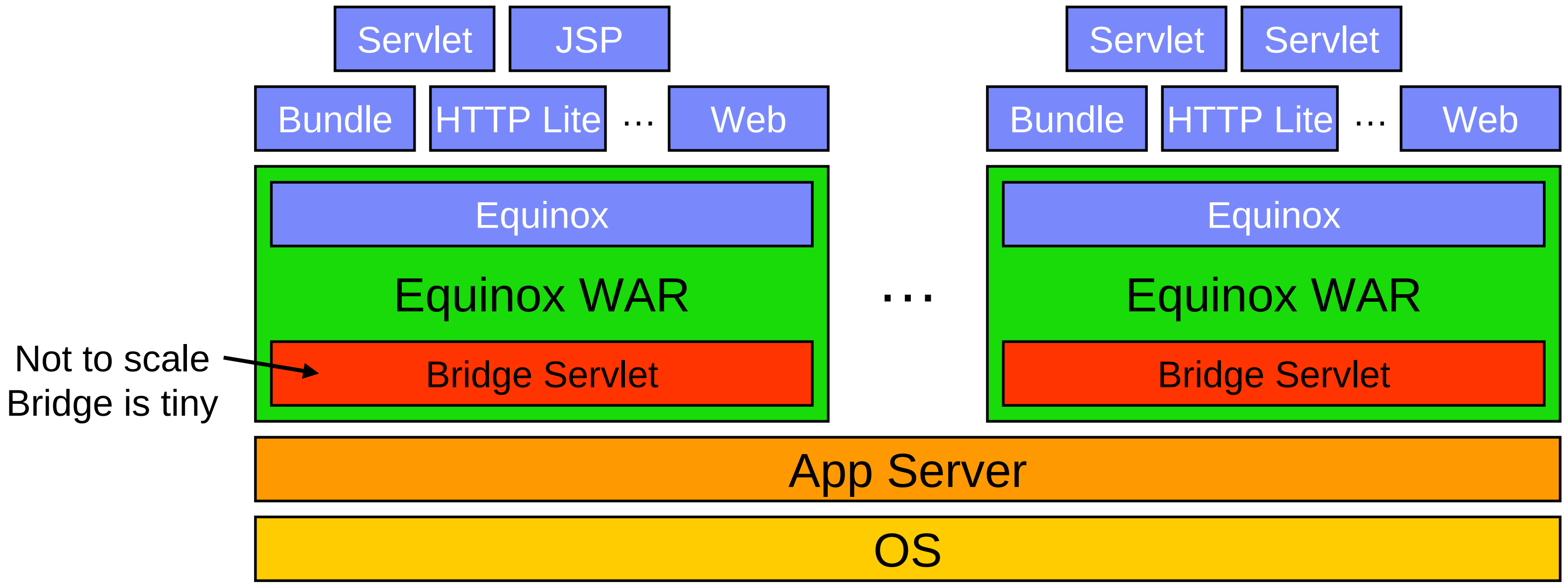
Traditional Server Example

- Server function (e.g., servlets) packaged in a WAR
- Application Install/Update/Manage whole WARs
- Application isolation
- No OSGi



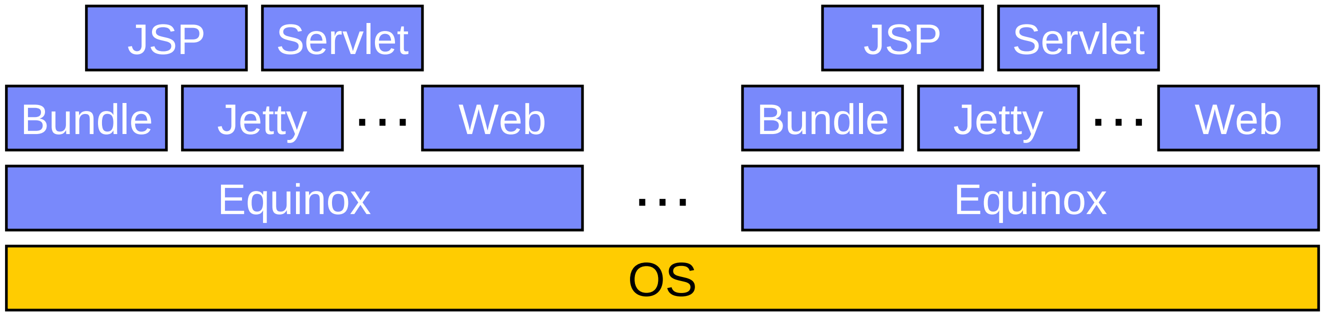
Equinox in an App Server

- Bridge servlet hosts Equinox in traditional App Server
- Application isolation
- Integration with existing infrastructure
- Forwarding (Lite) HTTP Service
  - Expose underlying App Server capabilities
- Add application function as bundles or servlets or JSPs, …
- Install/Update/Manage “WAR” by managing bundles



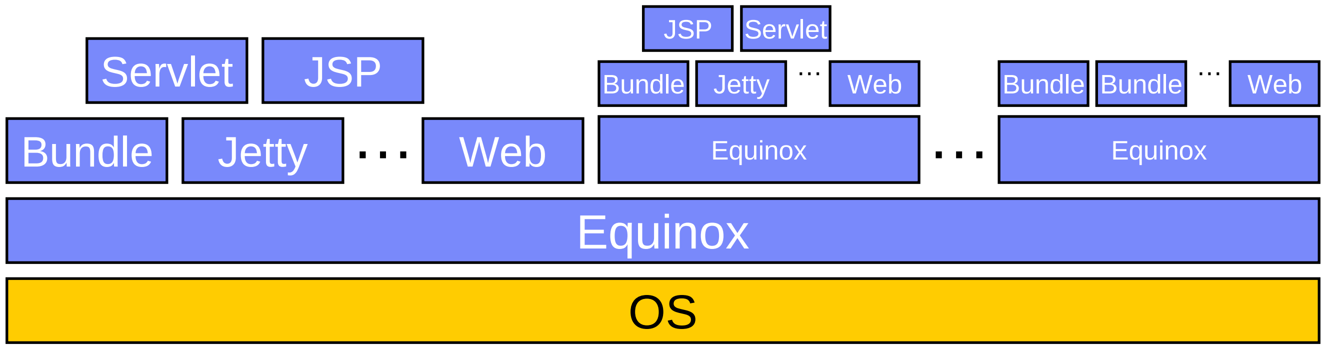
Raw Equinox

- Run Equinox directly
- **Process** isolation
- HTTP Service (e.g., embedded Jetty bundle)
- Add application function as bundles or servlets or JSPs, …
- Install/Update/Manage server by managing bundles
- Web Services



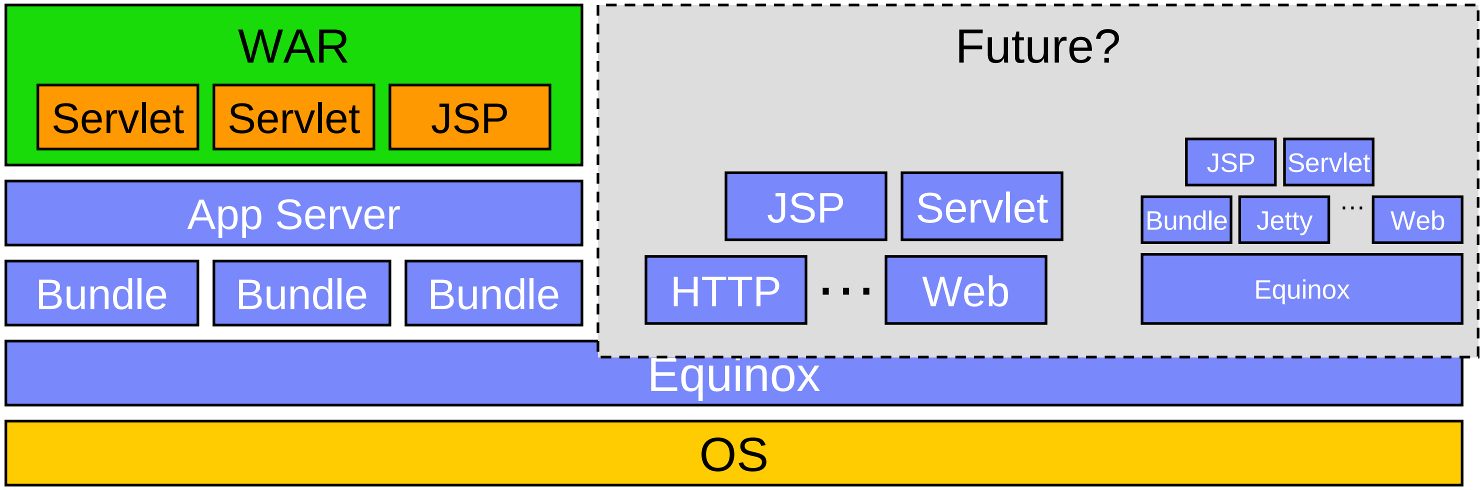
Equinox nested in Equinox

- Run Equinox directly, nest other Equinox instances
- **Nested framework** isolation
- HTTP Service (e.g., embedded Jetty bundle)
- Add server function as bundles, servlets, JSPs, …
- Install/Update/Manage server by managing bundles
- Web Services, …



App Server on Equinox

- Add ***App Server*** function as bundles
  - For example, Tomcat, Jetty, IBM WebShere …
- Tailor server configuration to match application needs
  - Dynamically
- Potential to combine all other approaches!



Advantages

- Incremental update of server function
- Run multiple versions simultaneously
- Individual configuration and management
- Accommodate disparate application prerequisites
- Class loading performance

- Share components across client and server
  - E.g., support disconnected mode

Technical Challenges

- Classloaders
  - Classloader parenting
  - Isolate nested entities from outside world
  - Context Classloader use
- System property isolation
- Statics and factories in the JRE
  - URLStreamHandlerFactory can only be set once

OSGi Looking Forward

- R5 work starting now
- JSR232 (OSGi for Java ME) released
- JSR291 (OSGi for JavaSE) Early Draft
  - R4.1 version of the spec
- Enterprise Expert Group (EEG) starting
  - Distributed Computing
  - Configuration Management
  - Provisioning

Summary

- Equinox is the basis for all Eclipse systems
  - Based on the OSGi R4 specification
- OSGi is gaining momentum across many domains
  - Serverside
  - Embedded
  - Desktop
- Everybody needs modularity

More Information?

- http://eclipse.org/equinox
- http://osgi.org