dm Server: Der Open-Source-OSGi-Application-Server

Eberhard Wolff – SpringSource

Java Forum Stuttgart 2009

About Eberhard Wolff

- eberhard.wolff@springsource.com
- Java Champion
- Regional Director and Principal Consultant
- Author of the first German Spring book

- Blog: http://JandIandMe.blogspot.com (German)

- 10+ year (Enterprise) Java

SpringSource Solution

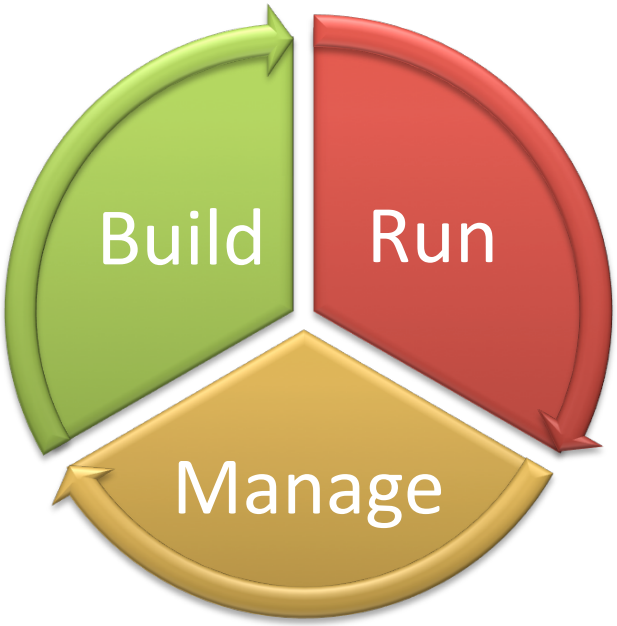
Unifying the Application Lifecycle: from Developer to Datacenter

High Productivity Tools

Spring Enterprise

Groovy and Grails

SpringSource Tool Suite



Lean Powerful Runtimes

SpringSource tc Server

SpringSource dm Server

SpringSource http Server

Application Infrastructure Management

SpringSource Hyperic HQ

SpringSource Hyperic IQ

SpringSource Offerings

- Subscriptions
  - Development and Production Support
  - Certified Software
  - Legal Indemnification
- Training (SpringSource University)
  - Public, on-site, certification program
- Professional Services
  - Client-driven engagements; packaged & custom

Delivered by Spring, Apache, Groovy/Grails and Hyperic Experts

Why another Application Server?

SpringSource dm Server™

- Modularization is key to maintainable software
- Application Server itself is modularized
  - No more "one size fits all"
  - Java EE 6 introduces profiles
- On the client and in the embedded world OSGi has succeeded as a standard for modularization
- OSGi enters the server market…

Building on OSGi

- Almost all app server vendors base their systems on OSGi (JBoss, Sun, Oracle, IBM)
- But *none* offers OSGi as a programming model for the customer

- Why shouldn't the customer be as empowered as the app server vendor?
- Enter SpringSource dm Server…

- SpringSource dm Server is Open Source (GPL)
- Professional support available

OSGi

It's a module system

- Partition a system into a number of modules – "bundles"

- Dynamic: Bundles can be installed, started, stopped, uninstalled and updated
- ...at runtime
- better operations

- Strict visibility rules
- Resolution process satisfies dependencies of a module
- Understands versioning

It's even service-oriented

- Bundles can **publish** services… *dynamically*!
- **Service Registry** allows other bundles to **consume** services
- Services come and go at runtime
  - … *transparently* when using Spring-DM

OSGi Bundle

- The fundamental unit of deployment and modularity in OSGi
- Just a JAR file
  - with additional entries in **`META-INF/MANIFEST.MF`**
- Common manifest headers:
  - **`Bundle-SymbolicName`**
  - **`Bundle-Version`**
  - **`Bundle-Name`**
  - **`Bundle-ManifestVersion`**
  - **`Bundle-Vendor`**

Import / Export -Package

Declares package-level dependencies of your bundle.

```
Import-Package: com.xyz.foo;
Import-Package:
  com.xyz.foo;version="1.0.3"
Import-Package:
  com.xyz.foo;version="[1.0.3,1.0.3]"
Import-Package:
  com.xyz.foo;version="[1.0.3,1.1.0)",
  com.xyz.bar;version="[1.0.3,2.0.0)"
Export-Package: com.xyz.foo
Export-Package: com.xyz.foo;version="1.0.5"
```

>= 1.0.3; e.g., 1.0.3.GA, 1.0.4, etc.

Register a Servce

```
ServiceRegistration reg =
   bundleContext.registerService(
    Bar.class.getName(),
  myBarService);
...
reg.unregister();
```

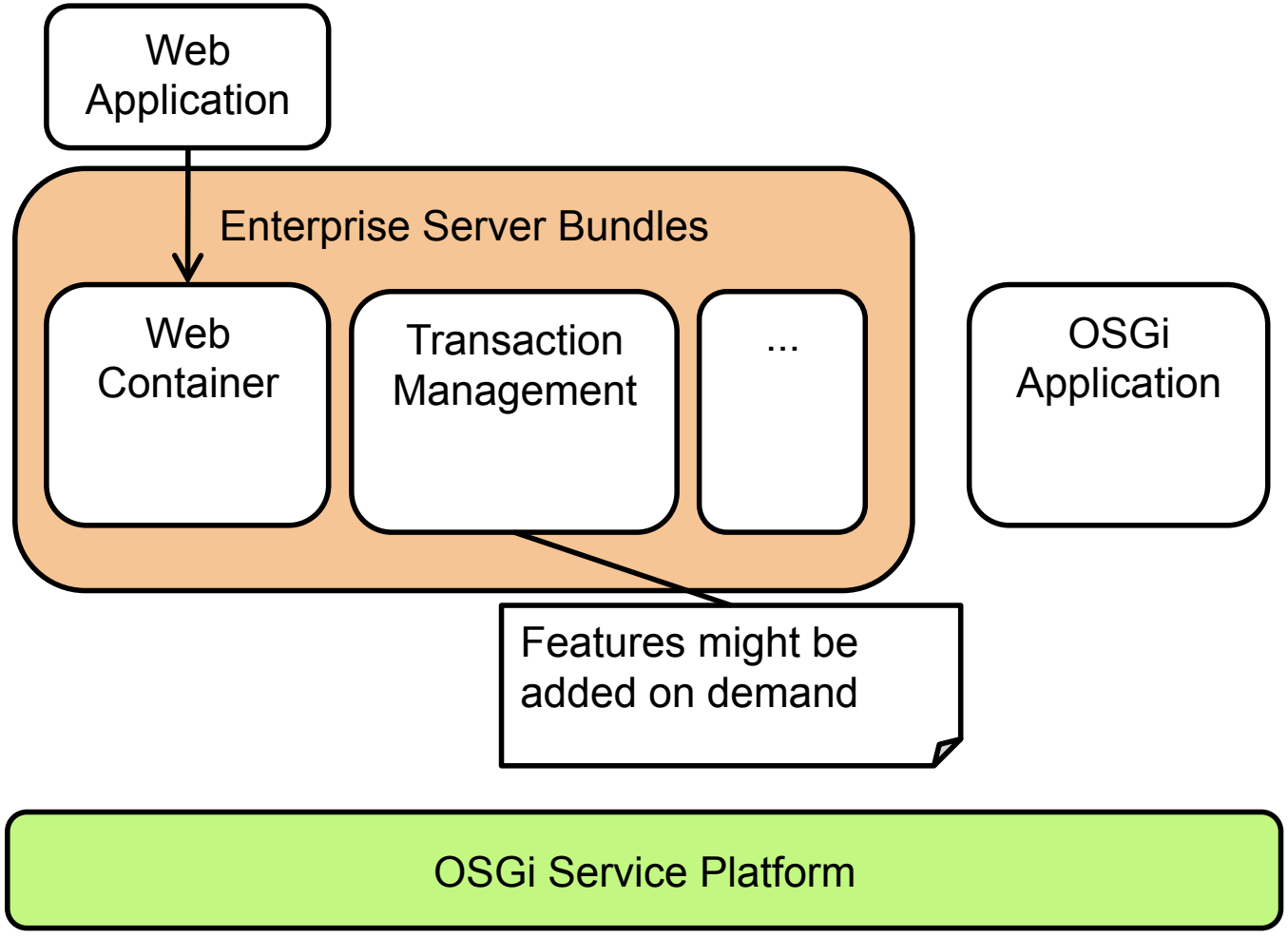
Use a Service

```
ServiceReference ref =
  bundleContext.getServiceReference(
   Bar.class.getName());
  Bar bar = (Bar)
    bundleContext.getService(ref);
  ...
  bundleContext.ungetService(ref);
  // bar should no longer be used here
```

Complex…
Potential resource leaks!

OSGi in the Enterprise

OSGi as a server platform



OSGi for your enterprise application...

- Dynamic services are hard to develop (boiler plate code)
- Basic infrastructure for OSGi + Web has to be done by yourself
- What do we do about the WARs?
- How do you keep a service / type from leaking out of an application? There is no concept of an *application* in OSGi…
- Many enterprise libraries are not suited for OSGi

Enterprise Libraries under OSGi

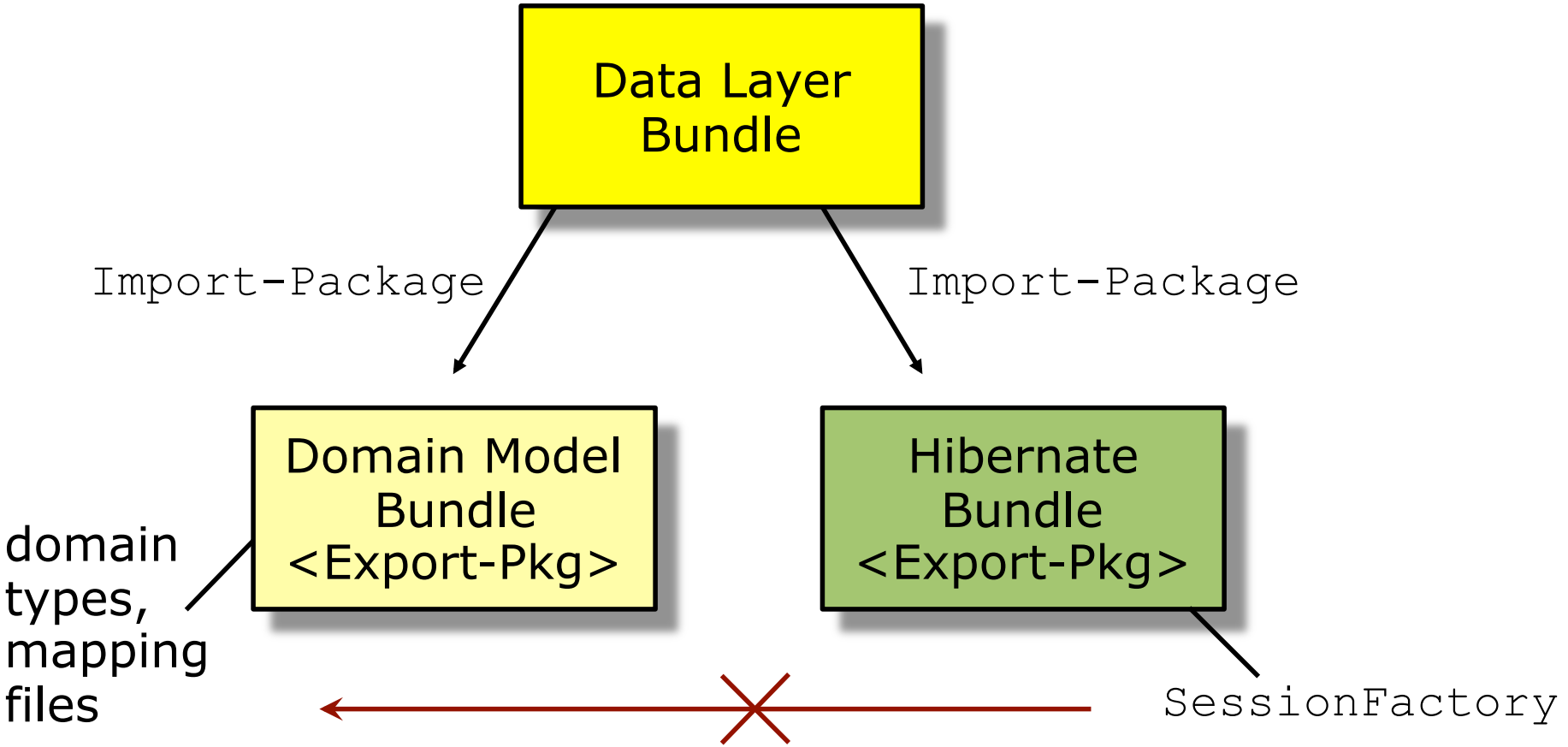
- Class and resource loading problems
  - class visibility
  - context class loader is undefined in OSGi
  - resources in `META-INF`
  - …
- Not directly supported: libraries with multiple bundles

Enterprise Libraries under OSGi

- Good news: Spring 2.5 and many other Spring projects are OSGi-ready
  - modules shipped as bundles
  - all class loading behaves correctly under OSGi

- Good news: Hundreds of other enterprise libraries now also packaged for use under OSGi at SpringSource Enterprise Bundle Repository

Example: Class visibility



Spring Dynamic Modules & SpringSource dm Server

Spring-DM: ApplicationContext

- Configuration files in `/META-INF/spring`
- Automatically merged
- ..and `ApplicationContext` is created

Service export and import

```
<beans ...>
  <osgi:service ref="customerDAO"
  interface="dao.ICustomerDAO" />
  <osgi:reference id="dataSource"
  interface="javax.sql.DataSource" />
</beans>
```

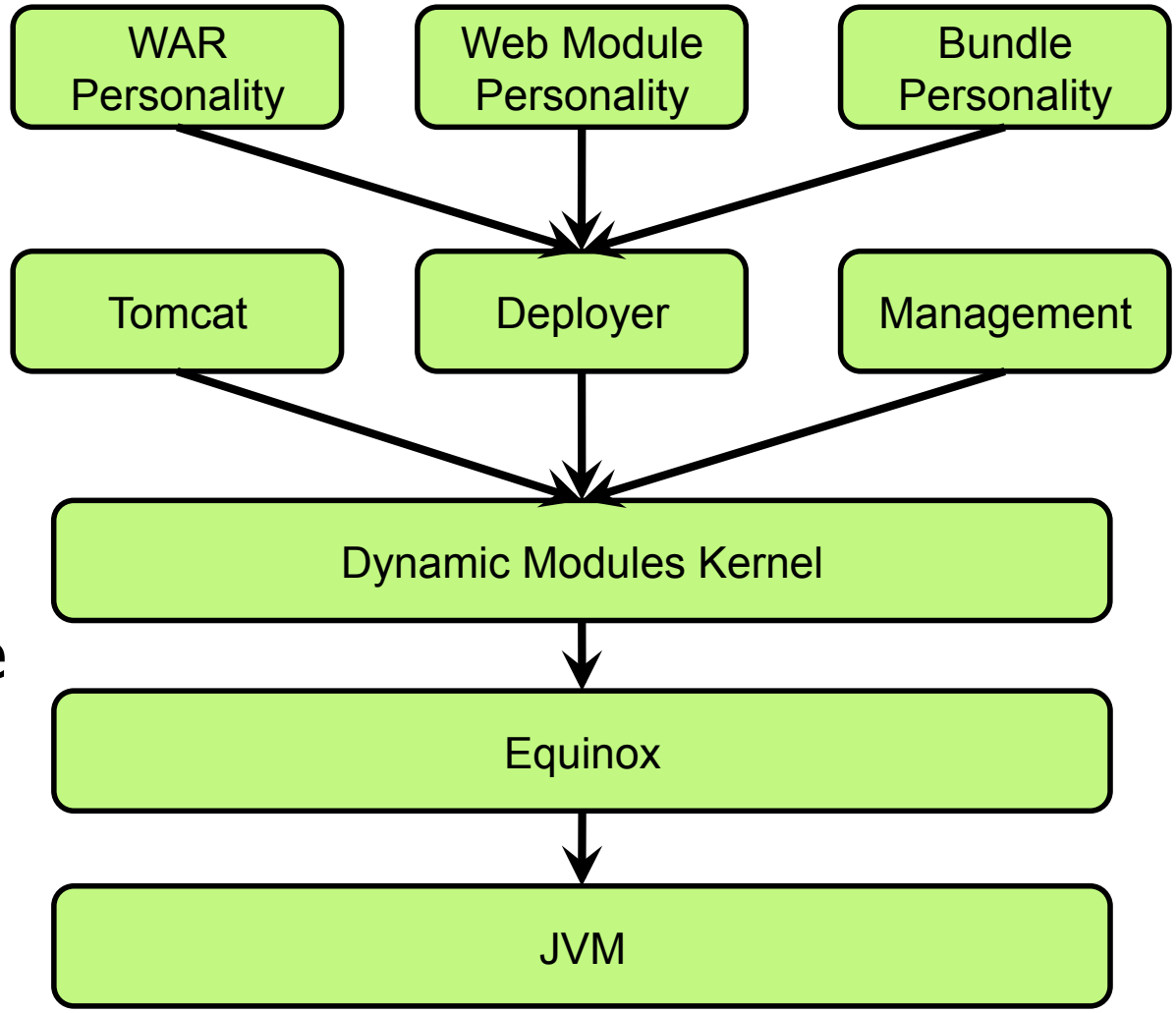
Service export and import

- Dynamic services automatically dealt with
- i.e. method calls are buffered
- Purely declarative
- No dependencies on OSGi in the code
- No resource leaks

- Not solved in Spring Dynamic Modules:
  - Easy import of libraries
  - Using JPA or Hibernate in OSGi
  - Seamless Web Support
  - Notion of an application
- Enter dm Server

dm Server Platform

- Modular profiles
- Bundle repository
- Library provisioning
- Serviceability
  - FFDC
  - Logging / Tracing
- Built on Equinox
- Modular architecture
  - Subsystems
  - Bundles
- Small footprint



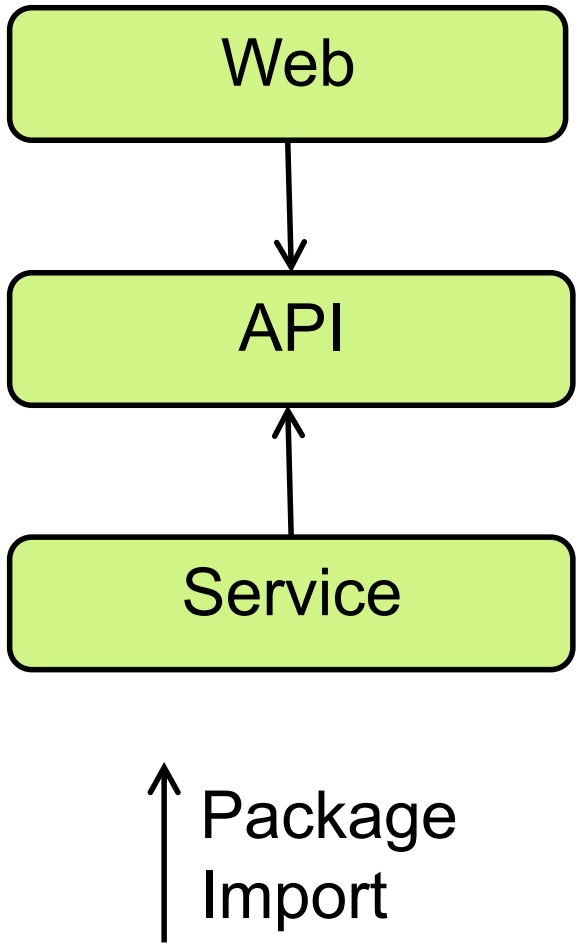
Type export and import

- Plain OSGi
- + libraries (i.e. multiple bundles)
- Library imports are converted to bundle imports
- i.e. OSGi runtime not changed

```
Import-Library: org.springframework.spring
```

Bundles for the example

- Web
- Service
- API: only interfaces and domain classes
  - Implementation can be exchanged
- Could add infrastructure: `DataSource` / `PlatformTransactionManager`

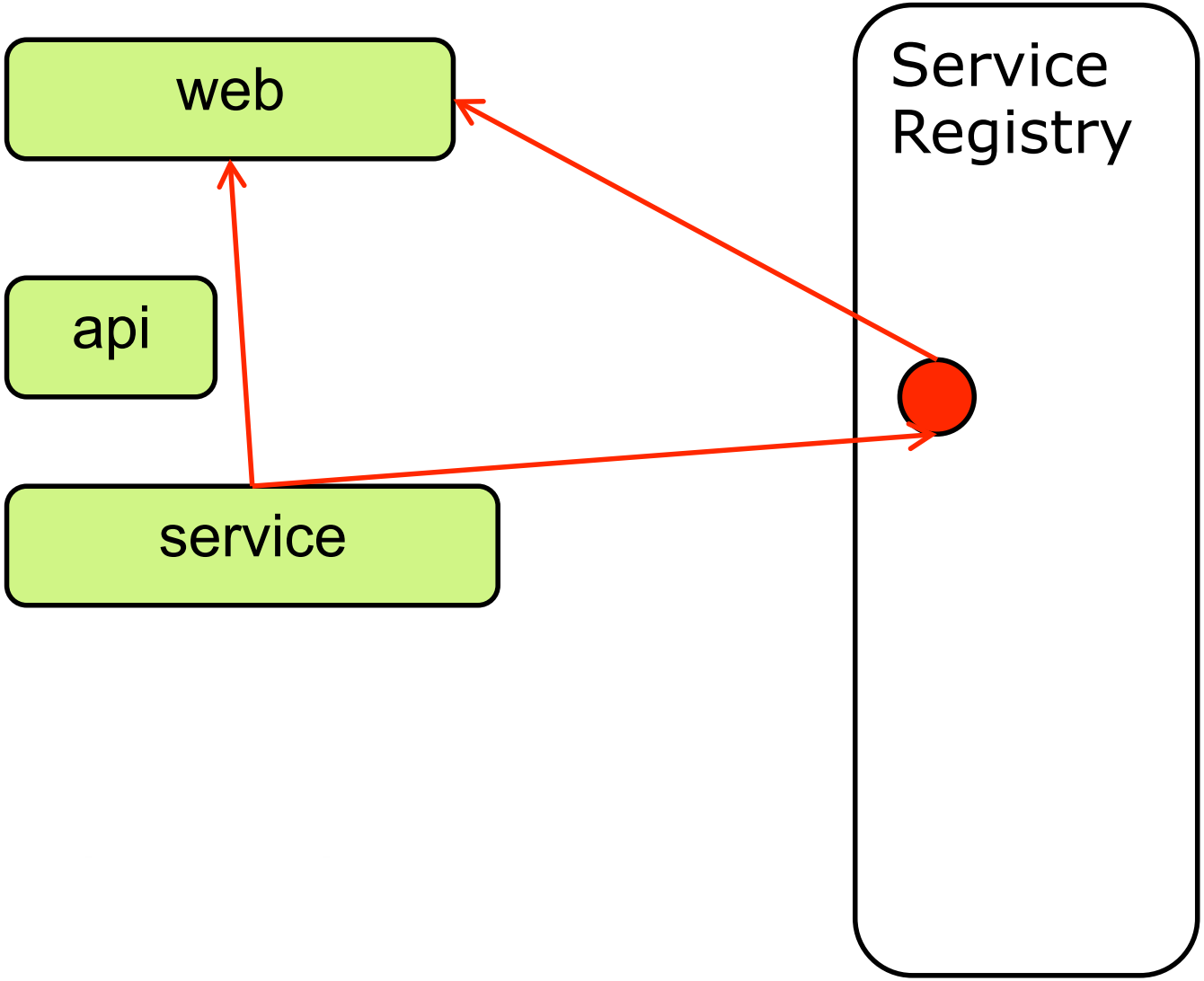


Bundles & Types

- Only dependencies to the API
- Therefore: implementation can be *exchanged even at* runtime
- No direct dependencies to any implementation
- Not shown: dependencies to external bundles
- … can be installed in dm Server
- … modular middleware!

Bundles & Services

Services: Publish / Consume



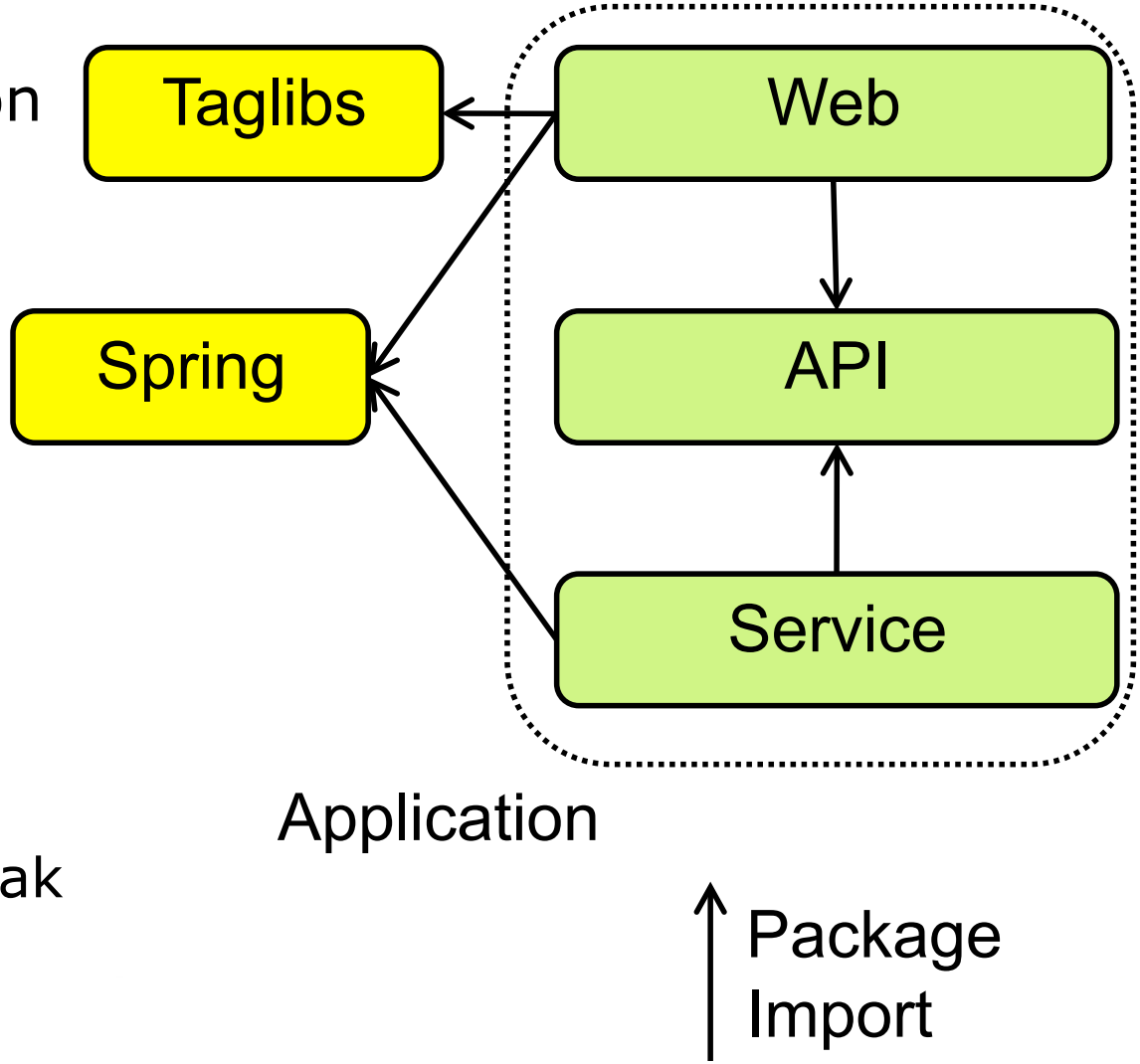
Bundles & Services

- Infrastructure can use the same principle as application services
- i.e. `DataSource` and `PlatformTransactionManager` are just another service

- Can I still run on plain Java EE?
- Yes: instead of OSGi Service directly inject Spring Beans
- no more more dynamic services / modularization
- No code change needed
- Application can run on Java EE or OSGi

PAR

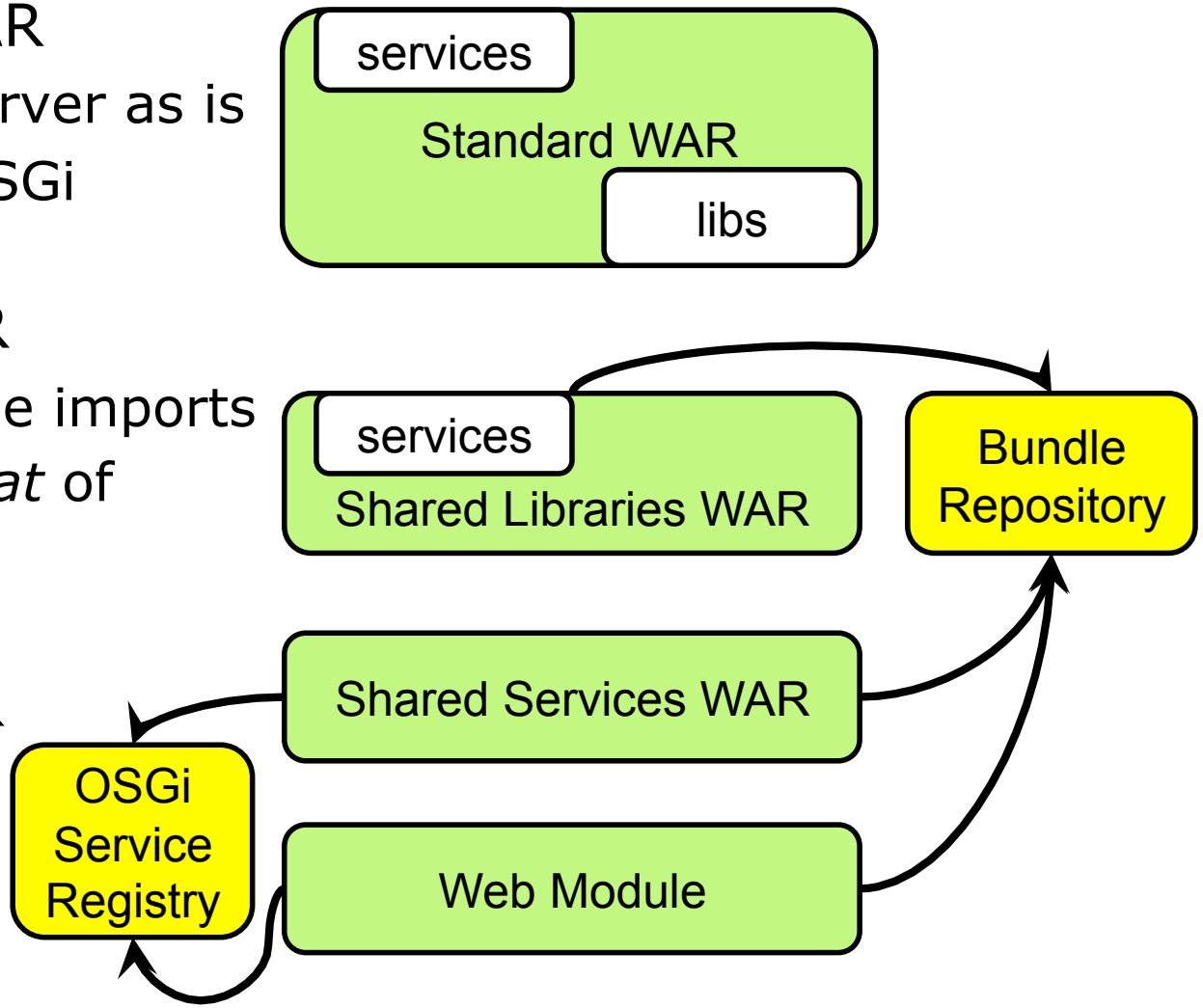
- Packaging format for all modules in an application
- JAR with Application-* manifest headers
- Single unit: deploy, refresh, undeploy
- Application boundaries
  - Scoping of types and services
  - DataSource does not leak out of the application
  - Hibernate can change domain objects



Web Migration: From WAR to PAR

Web Application Deployment Options

- Standard Java EE WAR
  - supported on dm Server as is
  - converted into an OSGi bundle
- Shared Libraries WAR
  - WAR + OSGi package imports
  - Eradicate *library bloat* of monolithic Java EE
  - WARs
- Shared Services WAR
  - Uses OSGi services with Spring's <osgi:reference>
- Web Module



Web Module

- Deployment & packaging option for OSGi-compliant web applications: stand-alone or within a PAR
- OSGi bundle: structure similar to Shared Services WAR + `MODULE-INF` for resources
- Reduced configuration for Spring MVC applications via web manifest headers
  - Auto-configuration of Spring MVC's **`DispatcherServlet`**
  - Single **`WebApplicationContext`** and no Root WAC
- Additional configuration via `web.xml` *fragments*

Web Manifest Headers


```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
...
Module-Type: Web
Web-ContextPath: /my-web-app
Web-DispatcherServletUrlPatterns: *.do
```

Roadmap

dm Server 2.0 Roadmap

- SpringSource dm Server 2.0: 3rd quarter 2009
- Cloning bundles
  - solves problems around static variables and more
- Shared Repository
  - make a repository available to other servers
- Plan Files
  - Define an application as a collection of bundles
  - Does not contain the bundles, more flexible
- Distributed and improved Management
  - operation on a group of servers
  - like tc Server for Tomcat
- Modular Web Applications

Support for Enterprise OSGi Standards

- **RFC 66**: Web Container for OSGi (RI based on dm Server)
- **RFC 119**: Distributed OSGi
- **RFC 124**: Blueprint Service (RI based on Spring-DM)
- **RFC 139**: JMX interface for OSGi
- **RFC 142**: JNDI and OSGi integration

Summary

SpringSource dm Server

- Based on proven, established modularization technology (OSGi)
- Based on proven, established web technology (Tomcat)
- Solves OSGi problems (e.g., context class loading)
- Dynamic updates to running modules
- Easy to use – special support for Spring
- Lightweight

Summary: Deployment Options

- OSGi bundles
- PAR = logical & physical application boundary
- dm Server supports multiple web deployment formats and therefore migration
  - Java EE WAR → Shared Libraries WAR → Shared Services WAR → Web Module

Resources

- **German Getting Started**:
  http://www.dpunkt.de/buecher/3231.html
- **Blog**:
  http://jandiandme.blogspot.com

- **OSGi**:
  http://osgi.org
- **Spring-DM**:
  http://springframework.org/osgi
- **SpringSource dm Server**:
  http://springframework.org/dmserver
- **SpringSource Team Blog**:
  http://blog.springsource.com

Questions?

Eberhard Wolff
eberhard.wolff@springsource.com

http://SpringSource.com