GlassFish V3 as an extensible server platform

Jerome Dochez, Principal Engineer
Kohsuke Kawaguchi, Senior Staff Engineer

TS-5921

Java

Sun microsystems

Agenda

- Modularity
- Services
- Runtime
- Distributions
- Developer's features
- Demo

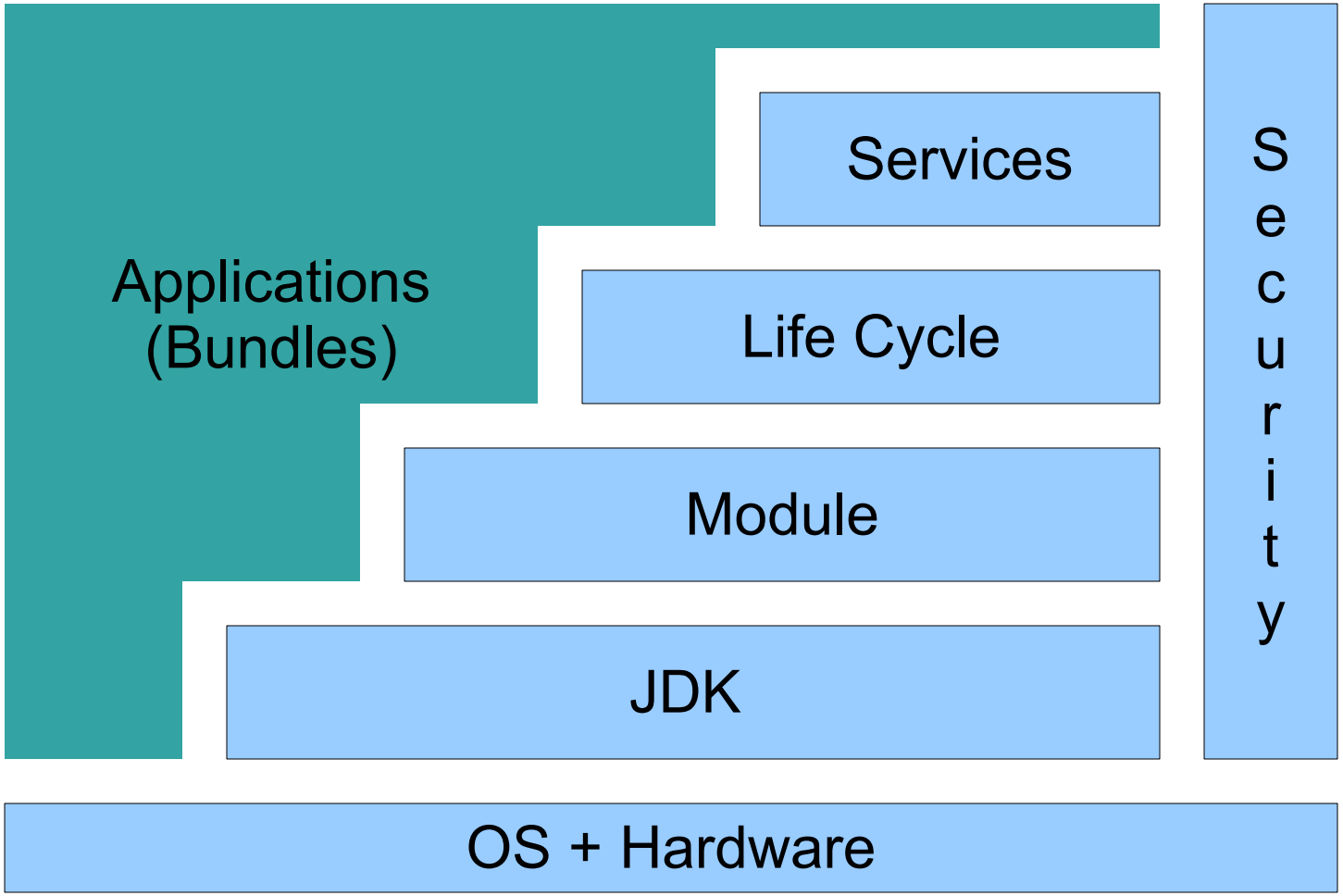
Introduction

- GlassFish, an Open Source application server built on and with Open Source components :
  - Maven 2
    - build system
    - module description
  - Felix
    - OSGi module management
  - Eclipse Link
    - Java Persistence Architecture implementation
- V3 is the third open source release of GlassFish

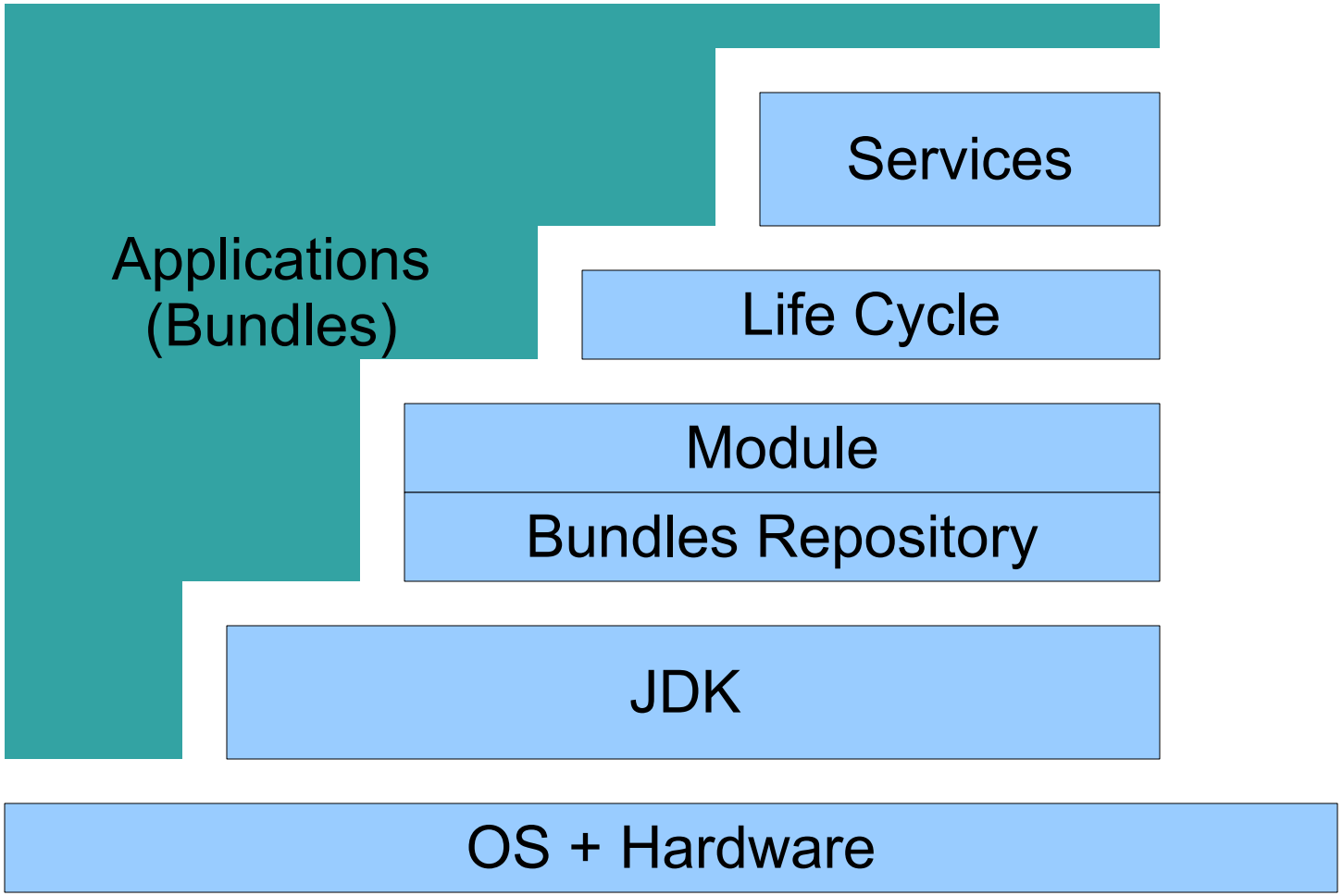
GlassFish V3 modularity

- Based on a module management runtime
  - last year : HK2 homegrown module management
  - today we add : OSGi Felix as the underlying module management runtime.
- Modules are great but let's face it, they do nothing just being modules !
- V3 innovation :
  - allow any type of container to be plugged in.
  - start all containers and services on demand yielding an unmatched startup time for a Java EE server.
  - focus on developer (keeping the already numerous deployer's features).

OSGi R4 Runtime



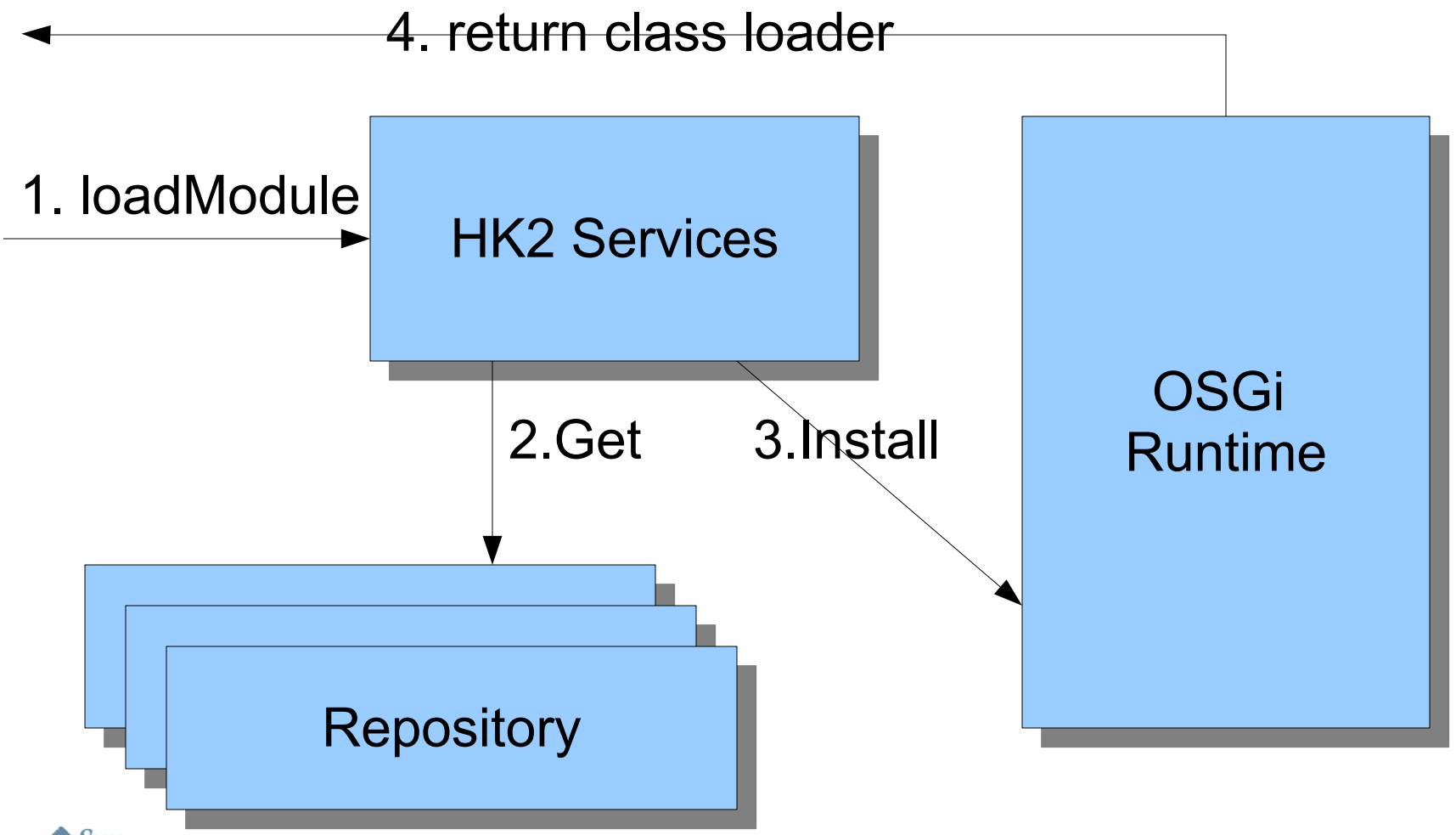
HK2 Module Management Runtime



HK2 + OSGi

- Use OSGi as the underlying module management
  - capable of consuming any OSGi modules
  - all GlassFish V3 modules are OSGi modules
- HK2 Provides runtime services on top of OSGi :
  - Repository management
  - Isolation from OSGi APIs or implementation
  - Services definition and lookup
  - MBeans support
  - Configuration facilities
- HK2 is a separate Open Source project from GlassFish because we think it could be reused by the community to build any type of server side software.

GlassFish V3 Module management



Maven 2

- Build system (obviously)
- Relatively easy to add plugins to specialize build time activities :
  - generation of metadata
  - APT plugins hookup
- Modules descriptions resemble individual maven project files (pom.xml)
  - easy to create complicated reports based on pom.xml relationships
- One tool to build GlassFish makes automation, reporting and code separation easier.

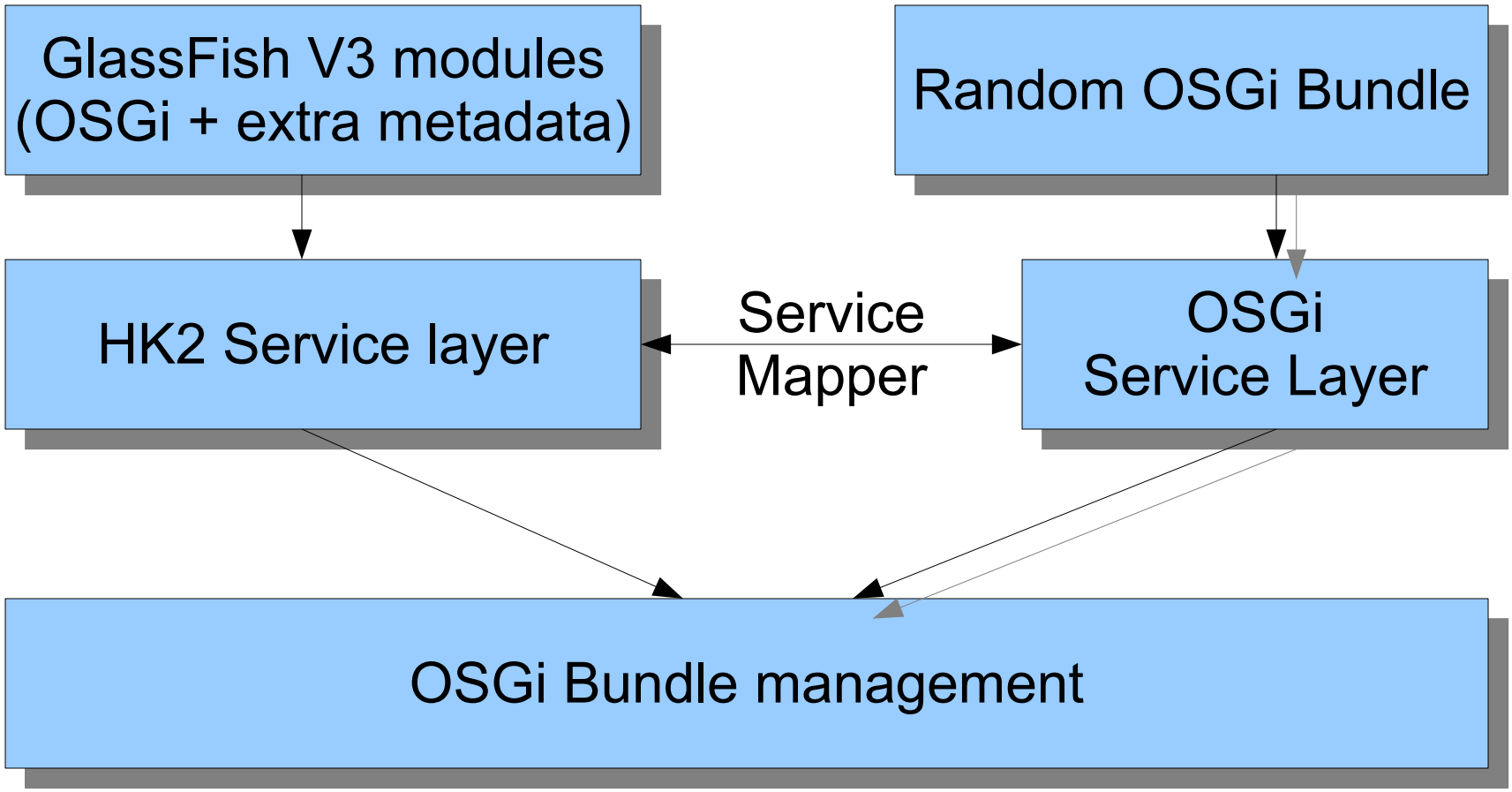
Agenda

- Modularity
- Services
- Runtime
- Distributions
- Developer's features
- Demo

Services

- GlassFish V3 use extensively Services to identify extension points like :
  - Application Containers (like Web-App, Phobos, Jruby...)
  - Administrative Commands
- Services are :
  - implementing an interface
    - Java SE style with a META-INF/services file
    - OSGi style
    - HK2
- Can be stateless or statefull

GlassFish V3 Runtime



HK2 Services

- Interfaces are declared with @Contract
- Implementations are declared with @Service
- Build system will generate Metadata automatically

```
@Contract
public interface Startup {...}

@Service
public class ConfigService implements Startup
{
... }
```

Services Resolution : Injection

- Services are injected at runtime using a dependency injection framework
- @Inject to declare a dependency on a Service
  - On any @Service annotated class
  - Field :

```
@Inject
ConfigService config;
```

  - Setter method :

```
@Inject
public void set(ConfigService svc) {..}
```

Services Resolution : API access

- Use Habitat to retrieve services instances :

```
public <T> T getComponent(
    Class<T> providerClass)

public Iterable<T> getComponents(
    Class<T> contract)
```

- Possible of obtaining lazy references to services
- You can also add services at runtime for newly available/installed extensions.

Instantiation cascading

```
@Contract public interface Startup {...}
```

- somewhere some code does :

```
@Inject Startup[] startups
```

- in DeploymentService:java

```
@Service public class DeploymentService implements
   Startup {
@Inject ConfigService config; }
```

- in ConfigService.java:

```
@Service public Class ConfigService implements ...
```

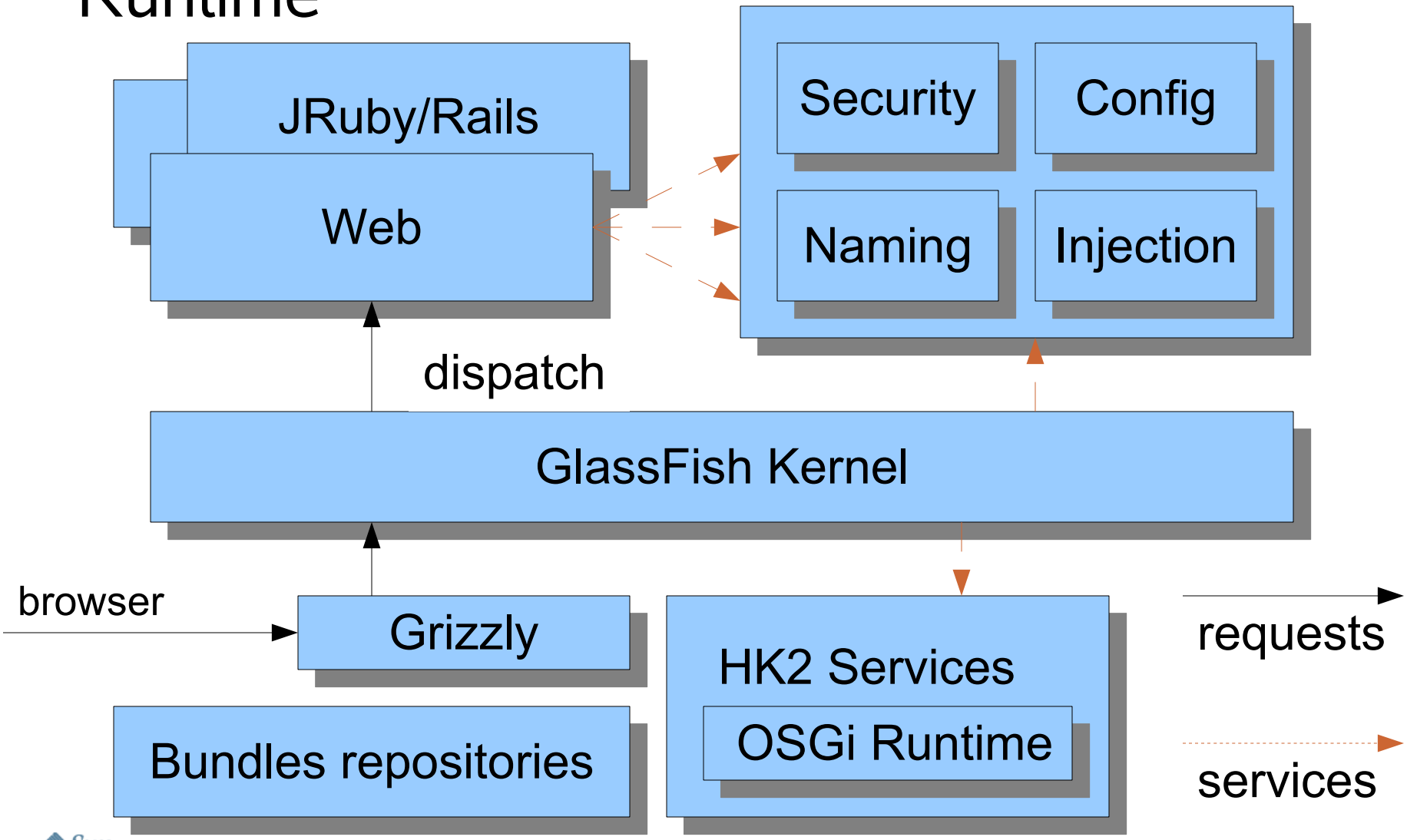
Agenda

- Modularity
- Services
- Runtime
- Distributions
- Developer's features
- Demo

GlassFish V3 Runtime

- Kernel
  - startup/shutdown sequences
  - basic services (deployment)
  - configuration reading
- Services
  - Cross containers functionalities
    - Security, Naming Manager...
    - Admin Console
- Containers
  - handle user's applications
  - independent of each others

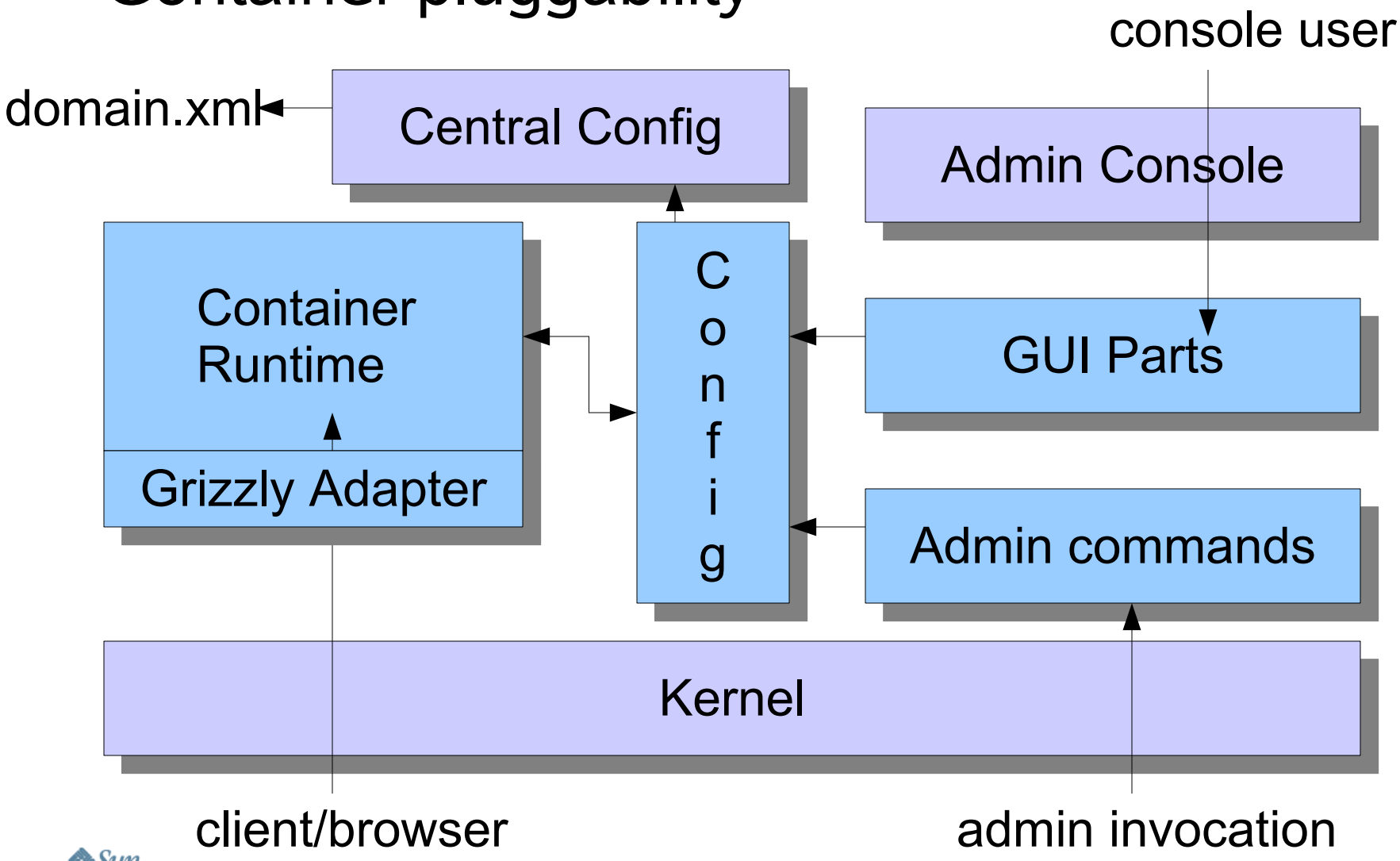
Runtime


JRuby/Rails
Web
Security
Config
Naming
Injection
dispatch
GlassFish Kernel
browser
Grizzly
HK2 Services
OSGi Runtime
Bundles repositories
requests
services

GlassFish V3 containers

- Basic Functionalities
  - handles certain application types
  - started and stopped on demand
- Optional features
  - Configuration services (so all containers configuration can be centrally managed)
  - administration commands
  - administration console plugins
  -

Container pluggability



Agenda

- Modularity
- Services
- Runtime
- Distributions
- Developer's features
- Demo

GlassFish Distributions

- Distributions are GlassFish V3 final deliverables and contains :
  - basic set of features
  - containers
  - libraries
- Built using maven 2 and ant
- Adding a module to a distribution is as simple as adding it to the distribution's pom.xml
  - features will be discovered with the Services mechanisms
  - application imports will use the OSGi package wiring

Basic Distribution : Nucleus

- Contains minimum set of modules and functionalities but no containers
- Size : 2Mb
- Features :
  - configuration
  - basic application lifecyle management
    - deployment/undeploy/redeployment
  - module management
- Used to build GlassFish V3 distributions
- Use it to build your tailored application server.

Distributions

- Distributions are all starting with the nucleus
  - then, add more features to it
- Distributions can inherit from each other
  - Example : Web Distribution with basic web container functionalities
  - Possible children
    - Web Services Distribution with metro stack
    - Web+Spring
    - Java EE 6 web distribution with local EJB (transactional) container.
- Anyone can extend a distribution to tailor their needs
- Others can use technology like Update Center to augment their distribution with new(er) modules.

Agenda

- Modularity
- Services
- Runtime
- Distributions
- Developer's features
- Demo

Embeddable GlassFish

- Make GlassFish usable as a library
  - Not as the master of JVM that you have to run in
- This is not about running GlassFish in embedded devices
- Requirements
  - No OSGi, no modules, no installation directory
  - No asadmin

What we came up with

- single jar that includes the whole thing
  - Can also use individual jars
- Drop it in your classpath and you are done
- Programmatic API to replace “asadmin”
  - Configure HTTP listeners
  - Deploy/undeploy applications
- No need to have a war file
  - Classes from here, jars from there, web.xml from another place...

Why does it matter?

- Same container from development to production
  - Without loss of productivity
- Tap into the power of JavaEE
  - EJB, JMS, web services, ...
- Improved unit testing opportunities
- App server built to order
  - Pick up just the modules you need

Agenda

- Modularity
- Services
- Runtime
- Distributions
- Developer's features
- Demo

Summary

- GlassFish V3 application server
  - modular application server implementation
  - flexible distribution mechanism
  - unmatched startup time
  - use OSGi compliant runtime for industry support
- Run multiple server side containers not limited to Java EE™
- Container for all types of server side containers

THANK YOU

GlassFish V3 as an extensible server platform

Jerome Dochez
Kohsuke Kawaguchi

TS-5921