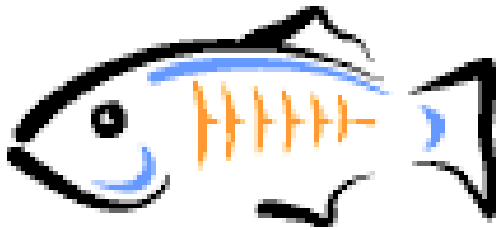
JavaOne℠

Java™ Platform, Enterprise Edition 6 with Extensible GlassFish™ Application Server v3



Jerome Dochez
Mahesh Kannan
Sun Microsystems, Inc.

Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

GlassFish V3

- Java EE 6
  - Profiles
  - Web-developer productivity
- Modularity
  - Grow from minimal (3mb) to full Java EE support and more...
  - Based on OSGi
- Embedded
  - Start using Java EE 6 platform features in your Java SE application (JPA, local ejb...)
  - API driven

GlassFish V3 product differentiation

- Open for all JVM based technologies
  - JRuby/Grails
  - Native deployment (no war repackaging)
- Extensible
  - Extensive APIs to replace or extend features
  - OSGi also provides extensions capabilities
- Service based architecture
  - services are defined by contracts and can be easily substituted
  - lazy loading based on usage patterns

Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

Modularity

- Based on OSGi
- GlassFish v3 is delivered as 170 bundles
  - Undoubtedly too many...
  - Successfully maintained quick startup
- Modules nomenclature
  - Services they provide
  - Lifecycle
- Strong build tool is necessary, we used maven
- Used to deliver Java EE profiles

Embedded API

- Start and configure GlassFish using APIs

```
// Create the Server
  ServerBuilder builder = ServerBuilder.get(“sample”)
  Server server = builder.setLogFile(“/tmp/foo.log”).create();

// Configure the web container
  server.add((server.getBuilder(WebContainerBuidler)).setHtt
  pPort(server.createPort(8080));

// Start the server
  server.start();
```

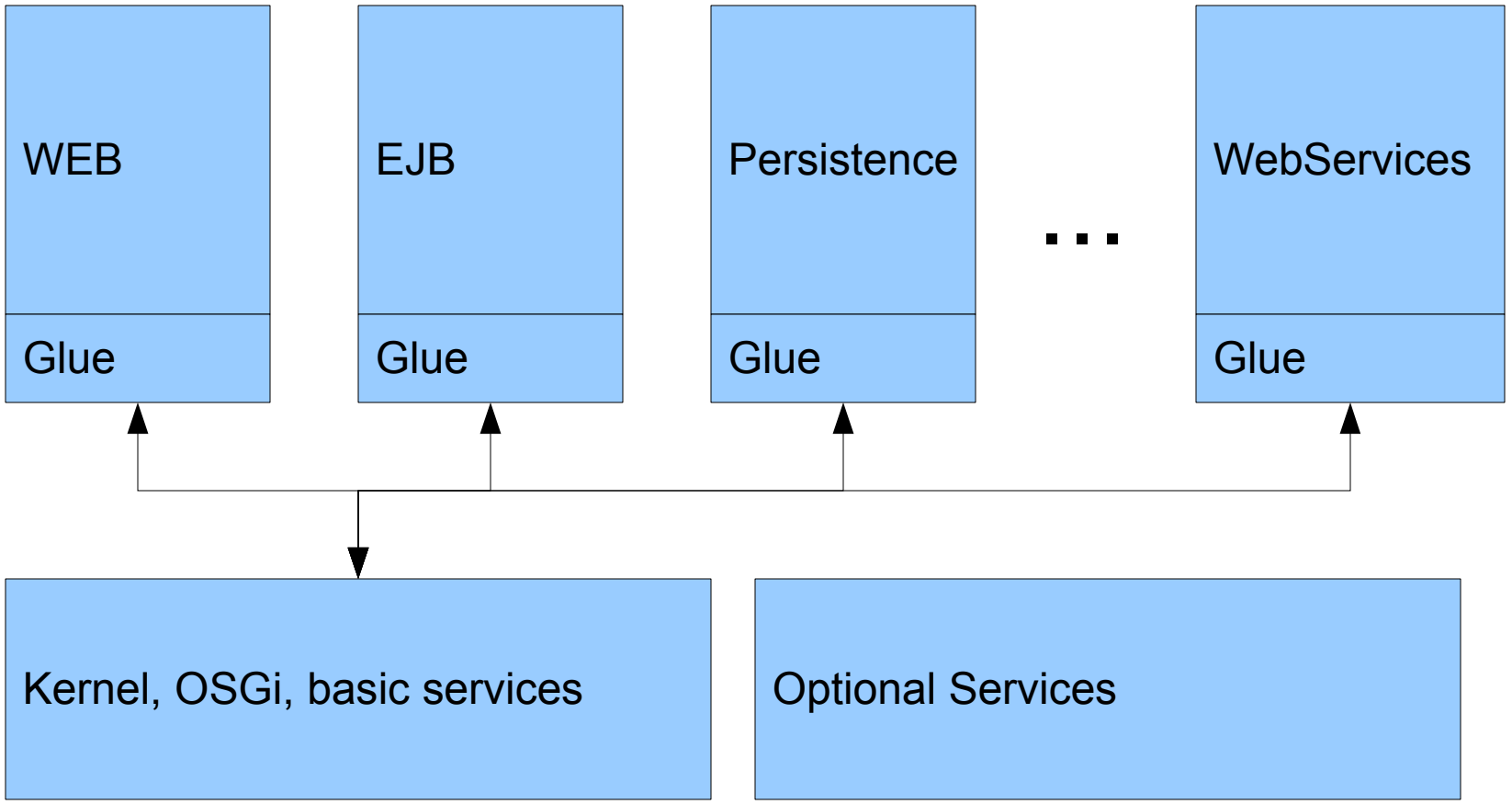
- Shortcuts possible :

```
  new WebContainer(8080).deploy(...);
```

Runtime

- Kernel
  - startup/shutdown sequences
  - basic services (deployment)
  - configuration reading
- Services
  - Cross containers functionalities
    - Java EE 6 : Security, Naming Manager...
    - Products : Admin Console
- Containers
  - handle user's applications
  - independent of each others

Container Isolation (Java EE 6 Profiles)



Application Container

- Deployment
  - Ability to add/remove/start/stop applications
- Configuration
  - Ability to have configuration stored in central configuration
  - Automatic clustering support
- Commands
  - CLI, REST invocations
- Monitoring (MBeans, Gmbal)

Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

Service Based Architecture

- Modules are just packaging artifact
- Capabilities and relationships are formed using services
  - OSGi services
  - HK2 services
    - Abstraction to OSGi services
    - Per thread/Per request scopes
    - Integrated lightweight dependency injection

HK2 Service

- 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
{
... }
```

Service Lookup

- Dependency injection

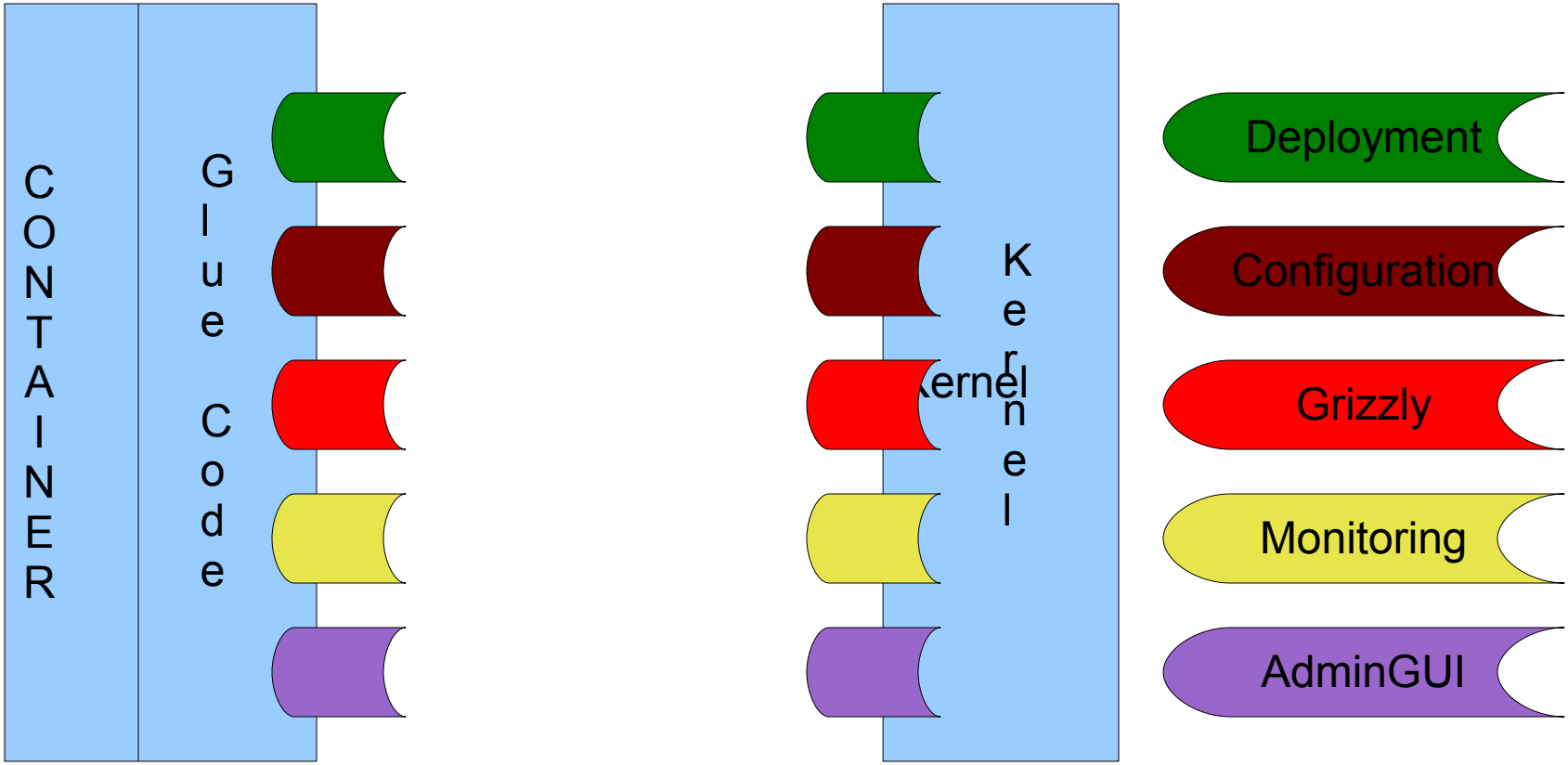
```
@Service
Public class RandomService implements SomeCtr {
        @Inject
        Startup someStartupService
}
```

- Automatic cascading
- Name resolution : @Inject(name=”foo”)
- Can be API driven (no DI)

Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

Application Container Runtime



Deployment

- Sniffer : recognize applications
  - Stigma : META-INF/ejb-jar.xml
  - Any class annotated with @Stateless
- Container (1 per container)
  - Maintain container state
  - Points to the deployer implementation
- Deployer (1 per container)
  - Deploy, undeploy
- ApplicationContainer (1 per application)
  - Start, stop, suspend, resume

Configuration

- All configuration in v3 uses annotated interface

```
public interface CtrConfig extends Container {
    @attribute
    String s1;
    @Element
    SubCtrConfig
}
```

- Will result in our central configuration

```
<ctr-config s1=”foo”>
    <sub-ctr-config .../>
</ctr-config>
```

Configuration (2)

- Lands in our central config (domain.xml)
  - Automatic clustering support
  - Automatic MBeans generation
  - Automatic REST interface
  - Automatic transaction access to mutating data
- Well integrated product, yet made of disparate bits and pieces.
- All configuration interfaces are implemented by a single class we own, very lightweight.

REST

- Use JAX-RS to provide REST interfaces to
  - Configuration data
  - Commands invocation (deploy, undeploy, etc..)
  - Monitoring
- Binding use annotations on @Configured configuration model, @Service command implementations to generate WADL.
  http://localhost:4848/rest-resources/domain
- You can even use REST clients as an AdminGUI substitute

CLI

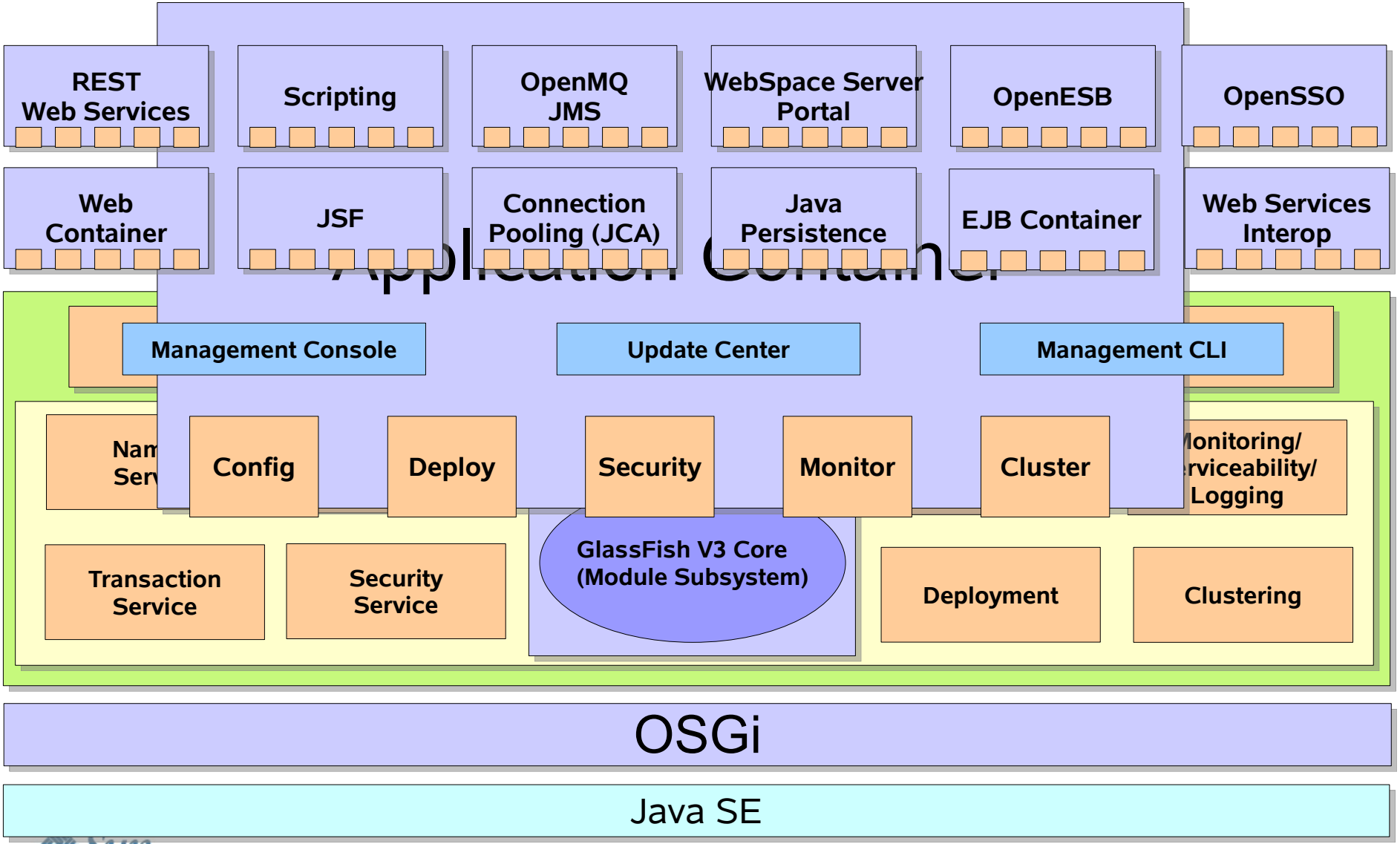
- Administrative commands can be added with each container :
  - Manipulates the container configuration

```
@Service(name=”change-random-ctr”)
  public class ChangeRandomCtr implements
  AdminCommand {
        @Param
        String s1;
        @Param
        String s2;
  … }
```

  - Available as :

```
asadmin change-random-ctr –s1 foo –s2 bar
```

GlassFish v3 – Architecture



Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

OSGi integration

- No OSGi APIs are used in GlassFish v3
  - Hk2 provides isolation layer
  - Mockup implementation of hk2 makes V3 capable of running without OSGi
    - Embedded mode
    - Static mode (single class loader)
- All GlassFish modules are OSGi bundles
- Completely portable to any OSGi R4 compliant runtime
  - Felix is the default

OSGi services implemented in v3

- HTTP Service
  - Simple dynamic servlet web server
- OSGi Alliance RFC 98
  - Transactions in OSGi
- OSGi Alliance RFC 66
  - OSGi-based web container

OSGi integration

- Module management
  - Add, remove, update installed modules
- OSGi as a container !
  - Treat OSGi just like any container, bundles are deployed to it.
- Converged Applications
  - Started investigating Java EE 6 + OSGi converged applications :
    - Dependencies in OSGi
    - Lifecycle still governed by Java EE.

OSGi Integration (2)

- OSGi services
  - Available to any Java EE application

    ```
    @Resource(mappedName="osgiName")
    SomeOSGiService injectedService;
    ```
  - JNDI lookup
  - Portable, no OSGi dependencies in you Java EE application code
- No bundle management access
- All installed bundles exported APIs visible to Java EE Applications

OSGi bundle management

- OSGi bundles can be deployed to GlassFish
  - As a library
  - As an extension to glassfish
  - Managed like an application
- Coupled with well-known OSGi extended pattern, can be used to extend Glassfish runtime
- It's possible to extend GlassFish without using a single GlassFish APIs.

Agenda

- Java EE 6 and GlassFish V3
- Modularity, Runtime
- Service Based Architecture
- Containers inside out
- Demo
- OSGi integration
- Monitoring

Monitoring

- Monitoring possible on all platforms
- Probes are enabled dynamically
  - No probe listener result in a no-op
  - When listeners register, probe implementations are dynamically rewritten to start firing the probe events.
- Lightweight
- BTrace under the covers

Probe listeners

```
public class WebRequestMonitor {
   AtomicInteger counter = ...;
   @OnProbe("glassfish:web:request:started")
   public void webRequestStarted(String appName) {
       System.out.println("....");
   }
   @OnProbe("glassfish:web:request:stopped")
   public void webRequestStopped(String appName) {
       counter.incrementAndGet();
   }
}
```

What about DTrace and Solaris

- GlassFish is DTrace enabled
  - Modules define probe points
  - Open APIs, any module can define probe points
- Features
  - End to End Tracing and monitoring
    syscall=>WebRequest=>EjbRequest=>JPARequest==>MySQL
  - Use scripting languages like JavaScript

DTrace example

```
glassfish$1:web:request:started {
   trace(probename);
}
glassfish$1:web:request:stopped {
   trace(probename);
}
glassfish$1:jpa:query:beforeExecute {
   trace(copyintr(arg0));  //trace query string!!
}
```

Demo

- Demo: We will show GlassFish DTrace integration only
- The demo application is a simple Web, JPA application that queries all rows in DEPARTMENT table
- DEPARTMENT table contains only 20 rows
- But the query will take 15 to 20 seconds!!
- We will enable some of the GlassFish DTrace probes to examine whats going on...

GlassFish Community

Open Source and Enterprise Ready

- **GlassFish v3 Preview Available now!**
  - Java EE 6 reference implementation
  - Modular OSGi architecture – easy to develop & deploy
  - Runs in-process and easy to extend
  - Support for Ruby-on-Rails, Groovy and Grails, Python and Django
- **GlassFish v2 – Production Ready**
  - Best price/performance open source App server with Clustering, High Availability, Load Balancing
  - Secure, Reliable, Transactional, .NET-interop Web svcs
  - Support for Ajax and Comet
- **GlassFish ESB**
  - SOA and Business Integration platform
- **GlassFish Communications App Server**
  - SIP servlet technology for converged services

glassfish.org

- **24x7 Enterprise and Mission Critical Support**
  - sun.com/appserver
- **Tools Integration**
  - NetBeans and Eclipse
- **Pavilion booth numbers: 550, 566, 567**
- **Meet Java EE spec leads and experts at Ancillary Event & Booth**