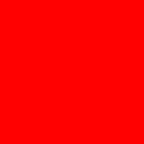




**ORACLE®**

**Java EE 6 & GlassFish 3:  
Light-weight, Extensible, Powerful**

Arun Gupta, Java EE & GlassFish Guy  
[blogs.sun.com/arungupta](http://blogs.sun.com/arungupta), @arungupta



The following/preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

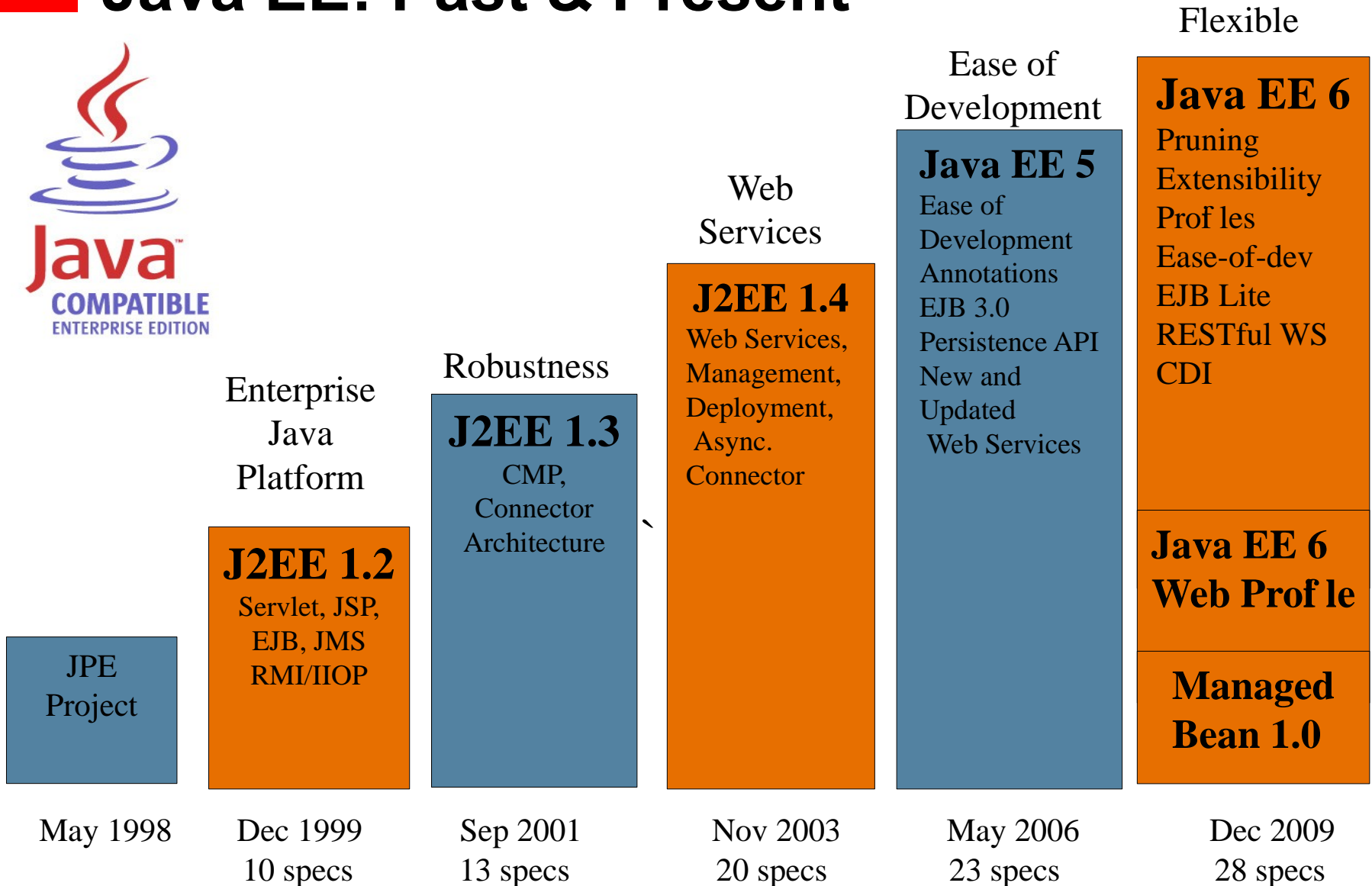


**Are you tweeting ?**

**#glassfish**

**#javaee6**

# Java EE: Past & Present



Flexible

Ease of Development

Web Services

Robustness

Enterprise Java Platform

Managed Bean 1.0

Java EE 6 Web Profile

**Java EE 5**  
Ease of Development  
Annotations  
EJB 3.0  
Persistence API  
New and Updated  
Web Services

**J2EE 1.4**  
Web Services, Management, Deployment, Async. Connector

**J2EE 1.3**  
CMP, Connector Architecture

**J2EE 1.2**  
Servlet, JSP, EJB, JMS, RMI/IIOP

# Compatible Java EE 5 Impl



<http://java.sun.com/javaee/overview/compatibility-javaee5.jsp>

# Compatible Java EE 6 Impls

Today:  **Tmax Soft**



Announced:

Oracle WebLogic Suite 11g



# Goals for the Java EE 6 Platform

- Flexible & Light-weight
  - JAX-RPC, EJB 2.x Entity Beans, JAXR, JSR 88
- Extensible
  - Embrace Open Source Frameworks
- Easier to use, develop on
  - Continue on path set by Java EE 5

# Java EE 6 Web Profile 1.0

- Fully functional mid-sized profile
  - Actively discussed in the Java EE 6 Expert Group and outside it
  - Technologies
    - Servlets 3.0, JSP 2.2, EL 2.2, Debugging Support for Other Languages 1.0, JSTL 1.2, JSF 2.0, Common Annotations 1.1, EJB 3.1 Lite, JTA 1.1, JPA 2.0, Bean Validation 1.0, Managed Beans 1.0, Interceptors 1.1, Context & Dependency Injection 1.0, Dependency Injection for Java 1.0

# Java EE 6 - Done

- Specifications approved by the JCP
- Reference Implementation is GlassFish v3
- TCK

Dec 2009

# Java EE 6 Specifications

- The Platform
- Java EE 6 Web Profile 1.0
- Managed Beans 1.0

# Java EE 6 Specifications

## New

- Contexts and Dependency Injection for Java EE (JSR 299)
- Bean Validation 1.0 (JSR 303)
- Java API for RESTful Web Services (JSR 311)
- Dependency Injection for Java (JSR 330)

# Java EE 6 Specifications

## Extreme Makeover

- Java Server Faces 2.0 (JSR 314)
- Java Servlets 3.0 (JSR 315)
- Java Persistence 2.0 (JSR 317)
- Enterprise Java Beans 3.1 & Interceptors 1.1 (JSR 318)
- Java EE Connector Architecture 1.6 (JSR 322)

# Java EE 6 Specifications

## Updates

- Java API for XML-based Web Services 2.2 (JSR 224)
- Java API for XML Binding 2.2 (JSR 222)
- Web Services Metadata MR3 (JSR 181)
- JSP 2.2/EL 2.2 (JSR 245)
- Web Services for Java EE 1.3 (JSR 109)
- Common Annotations 1.1 (JSR 250)
- Java Authorization Contract for Containers 1.3 (JSR 115)
- Java Authentication Service Provider Interface for Containers 1.0 (JSR 196)

# Java EE 6 Specifications

## As is

- JDBC 4.0 API
- Java Naming and Directory Interface 1.2
- Java Message Service 1.1
- Java Transaction API 1.1
- Java Transaction Service 1.0
- JavaMail API Specification 1.4
- JavaBeans Activation Framework 1.1
- Java API for XML Processing 1.3
- Java API for XML-based RPC 1.1
- SOAP with Attachments API for Java 1.3
- Java API for XML Registries 1.0
- Java EE Management Specification 1.1 (JSR 77)
- Java EE Deployment Specification 1.2 (JSR 88)
- Java Management Extensions 1.2
- Java Authentication and Authorization Service 1.0
- Debugging Support for Other Languages (JSR 45)
- Standard Tag Library for JSP 1.2 (JSR 52)
- Streaming API for XML 1.0 (JSR 173)

# Java EE 6 & Ease-of-development

- Continue advancements of Java EE 5
- Primary focus: Web Tier
- General principles
  - Annotation-based programming model
  - Reduce or eliminate need for DD
  - Traditional API for advanced users

# Servlets in Java EE 5

## At least 2 files

```
<!--Deployment descriptor
web.xml -->
<web-app>
  <servlet>
    <servlet-name>MyServlet
      </servlet-name>
    <servlet-class>
      com.sun.MyServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet
      </servlet-name>
    <url-pattern>/myApp/*
      </url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

```
/* Code in Java Class */

package com.sun;
public class MyServlet extends
HttpServlet {
  public void
doGet(HttpServletRequest
req,HttpServletResponse res)
{
  ...
}
...
}
```

# Servlets 3.0 (JSR 315)

## Annotations-based @WebServlet

```
package com.sun;
@WebServlet(name="MyServlet", urlPatterns={"/myApp/*"})
public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest req,
        HttpServletResponse res)
    {
        ...
    }
}
```

Deployment descriptor web.xml:

```
<!--Deployment descriptor web.xml -->
<web-app>
  <servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>
      com.sun.MyServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/myApp/*</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

[http://blogs.sun.com/arungupta/entry/totd\\_81\\_getting\\_started\\_with](http://blogs.sun.com/arungupta/entry/totd_81_getting_started_with)

# Servlets 3.0

## Annotations-based @WebServlet

```
@WebServlet(name="mytest",  
            urlPatterns={"/myurl"},  
            initParams={  
                @InitParam(name="n1", value="v1"),  
                @InitParam(name="n2", value="v2")  
            })  
public class TestServlet extends  
    javax.servlet.http.HttpServlet {  
    ....  
}
```

# Servlets 3.0

## Annotations-based @WebListeners

```
<listener>  
  <listener-class>  
    ▶ server.LoginServletListener ◀  
  </listener-class>  
</listener>
```

```
package ▶ server;
```

```
...
```

```
@WebListener()
```

```
public class LoginServletListener implements  
ServletContextListener {
```

# Servlets 3.0

## Asynchronous Servlets

- Useful for Comet, long waits
- Must declare `@WebServlet(asyncSupported=true)`

```
AsyncContext context = request.startAsync();
context.addListener(new AsyncListener() { ... });
context.dispatch("/request.jsp");
//context.start(Runnable action);
. . .
context.complete();
```

[http://blogs.sun.com/arungupta/entry/totd\\_139\\_asynchronous\\_request\\_processing](http://blogs.sun.com/arungupta/entry/totd_139_asynchronous_request_processing)

# Servlets 3.0

## Extensibility



- Plugin libraries using *web fragments*
  - Modular web.xml
- Bundled in framework JAR file in META-INF directory
- Zero-configuration, drag-and-drop for web frameworks
  - Servlets, servlet filters, context listeners for a framework get discovered and registered by the container
- Only JAR files in WEB-INF/lib are used

# Servlets 3.0

## Extensibility



```
<web-fragment>
  <filter>
    <filter-name>wicket.helloworld</filter-name>
    <filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>
    <init-param>
      <param-name>applicationClassName</param-name>
      <param-value>...</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>wicket.helloworld</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-fragment>
```

[http://blogs.sun.com/arungupta/entry/totd\\_91\\_applying\\_java\\_ee](http://blogs.sun.com/arungupta/entry/totd_91_applying_java_ee)

# Servlets 3.0

Much more ...

- Programmatic authentication, login, logout

- > `HttpServletRequest.authenticate`
- > `HttpServletRequest.login`
- > `HttpServletRequest.logout`

- File upload support

- Servlet Security

- > `@ServletSecurity`

```
<error-page>  
  <error-code>...</error-code>  
  <exception-type>...</exception-type>  
  <location>/404.html</location>  
</error-page>
```

- Default Error Page

- > Any HTTP status code
- > [http://blogs.sun.com/arungupta/entry/totd\\_136\\_default\\_error\\_page](http://blogs.sun.com/arungupta/entry/totd_136_default_error_page)

# EJB 3.1 (JSR 318)

## Package & Deploy in a WAR

Java EE 5

foo.ear

foo\_web.war

WEB-INF/web.xml  
WEB-INF/classes  
com.sun.FooServlet  
com.sun.TickTock

foo\_ejb.jar

com.sun.FooBean  
com.sun.FooHelper

Java EE 6

foo.war

WEB-INF/classes  
com.sun.FooServlet  
com.sun.TickTock  
com.sun.FooBean  
com.sun.FooHelper

~~web.xml~~ ?

[http://blogs.sun.com/arungupta/entry/totd\\_95\\_ejb\\_3\\_1](http://blogs.sun.com/arungupta/entry/totd_95_ejb_3_1)

# EJB 3.1

- No interface view – **one** source file per bean
  - Only for Local and within WAR
  - Required for Remote
  - No location transparency
- Component initialization in `@PostConstruct`
  - No assumptions on no-arg ctor
- Portable Global JNDI Name
  - Until now only `java:comp`

# EJB 3.1

## Embeddable API – Deploy the Bean

```
public void testEJB() throws NamingException {
    EJBContainer ejbC =
        EJBContainer.createEJBContainer();
    Context ctx = ejbC.getContext();
    App app = (App)
        ctx.lookup("java:global/classes/App");
    assertNotNull(app);
    String NAME = "Duke";
    String greeting = app.sayHello(NAME);
    assertNotNull(greeting);
    assertTrue(greeting.equals("Hello " + NAME));
    ejbC.close();
}
```

[http://blogs.sun.com/arungupta/entry/totd\\_128\\_ejbcontainer\\_createejbcontainer\\_embedded](http://blogs.sun.com/arungupta/entry/totd_128_ejbcontainer_createejbcontainer_embedded)

# EJB 3.1

## Singleton Beans

- Singleton Bean
  - One instance per app/VM, not pooled
  - `@Singleton`
- Asynchronous Session Bean
  - Light-weight JMS
- Calendar-based timers – cron-like semantics
  - Every Mon & Wed midnight  
`@Schedule(dayOfWeek="Mon,Wed")`
  - Every 14<sup>th</sup> minute within the hour, for the hours 1 and 2 am  
`(minute="*/14", hour="1,2")`

[http://blogs.sun.com/arungupta/entry/totd\\_137\\_asynchronous\\_ejb\\_a](http://blogs.sun.com/arungupta/entry/totd_137_asynchronous_ejb_a)

# EJB 3.1

## EJB 3.1 Lite – Feature Comparison

TABLE 2. EJB LITE AND EJB 3.1 FEATURES COMPARISON

FEATURE	EJB LITE	EJB 3.1
Local session beans	X	X
Declarative security	X	X
Transactions (CMT/BMT)	X	X
Interceptors	X	X
Security	X	X
Message-driven beans		X
RMI/IIOP interoperability and remote view		X
EJB 2.x backward compatibility		X
Time service		X
Asynchronous method invocations		X

# Contexts & Dependency Injection (JSR 299)

- Type-safe Dependency Injection
  - Builds on @Inject API
- Context/Scope management
- Works with multiple bean types
- Includes ELResolver

# CDI

## Injection Points

- Field, Method, Constructor
- 0 or more qualifiers
- Type

**@Inject @LoggedIn User user**

Which one ?  
(Qualifier)

Request  
Injection

What ?  
(Type)

# CDI

## Basics

- Separate from `@Resource` but can co-exist
  - `@Resource` for container managed DI
  - `@Inject` for application managed DI
- Strong typing, loose coupling
  - Clients only declare dependencies via injection points
  - Bean selection is done by CDI

# CDI – Sample Client Code

## Field and Method Injection

```
public class CheckoutHandler {  
  
    @Inject @LoggedIn User user;  
  
    @Inject PaymentProcessor processor;  
  
    @Inject void setShoppingCart(@Default Cart cart) {  
        ...  
    }  
  
}
```

# CDI – Sample Client Code

## Constructor Injection

```
public class CheckoutHandler {  
  
    @Inject  
    CheckoutHandler(@LoggedIn User user,  
                   PaymentProcessor processor,  
                   @Default Cart cart) {  
  
        ...  
    }  
  
}
```

- Only one constructor can have @Inject

# CDI - Sample Client Code

## Multiple Qualifiers and Qualifiers with Arguments

```
public class CheckoutHandler {  
  
    @Inject  
    CheckoutHandler(@LoggedIn User user,  
                   @Reliable  
                   @PayBy(CREDIT_CARD)  
                   PaymentProcessor processor,  
                   @Default Cart cart) {  
  
        ...  
    }  
  
}
```

# CDI - How to configure ?

There is none!

- Discovers bean in all modules in which CDI is enabled
- Beans are automatically selected for injection
- Possible to enable groups of bean selectively via a descriptor

# CDI - Scopes

- Beans can be declared in a scope
  - Everywhere: `@ApplicationScoped`, `@RequestScoped`
  - Web app: `@SessionScoped`
  - JSF app: `@ConversationScoped`
  - Pseudo-scope (default): `@Dependent`
- CDI runtime will make sure the right bean is created at the right time
- Client do NOT have to be scope-aware

# CDI

Much more ...

- Events
- Producer methods and fields
- Bridging Java EE resources
- Alternatives
- Interceptors
- Decorators
- Stereotypes

# Java Server Faces 2.0 (JSR 314)

- Facelets as “templating language” for the page
- Custom components much easier to develop

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Enter Name & Password</title>
  </h:head>
  <h:body>
    <h1>Enter Name & Password</h1>
    <h:form>
      <h:panelGrid columns="2">
        <h:outputText value="Name:" />
        <h:inputText value="#{simplebean.name}" title="name"
                     id="name" required="true" />
        <h:outputText value="Password:" />
        <h:inputText value="#{simplebean.password}" title="password"
                     id="password" required="true" />
      </h:panelGrid>
      <h:commandButton action="show" value="submit" />
    </h:form>
  </h:body>
</html>
```

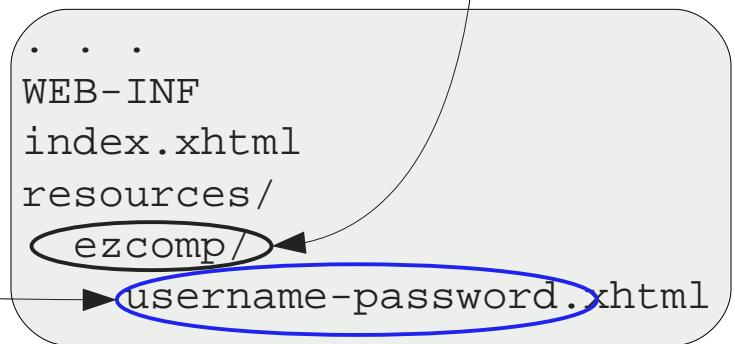
# JSF 2 Composite Components

```
<h:body>
  <h1>Enter Name & Password</h1>
  <h:form>
    <h:panelGrid columns="2">
      <h:outputText value="Name:" />
      <h:inputText value="#{simplebean.name}" title="name"
        id="name" required="true" />
      <h:outputText value="Password:" />
      <h:inputText value="#{simplebean.password}" title="password"
        id="password" required="true" />
    </h:panelGrid>
    <h:commandButton value="Submit" action="show" value="submit" />
  </h:form>
</h:body>
```

The image shows a code editor with a context menu open over a JSF 2 composite component. The code defines a form with a grid of two columns. The first column contains a label 'Name:' and an input field for the name. The second column contains a label 'Password:' and an input field for the password. A submit button is located below the grid. The context menu is open, showing options such as View, Find Usages, Refactor, Format, Insert Code, Cut, Copy, Paste, Code Folds, and Select in. The 'Refactor' option is selected, and a sub-menu is visible with options like Rename, Move, Copy, Safely Delete, and Convert To Composite Component...

# JSF 2 Composite Components

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ez="http://java.sun.com/jsf/composite/ezcomp">
  <h:head>
    <title>Enter Name & Password</title>
  </h:head>
  <h:body>
    <h1>Enter Name & Password</h1>
    <h:form>
      <ez:username-password/>
      <h:commandButton action="show" value="submit"/>
    </h:form>
  </h:body>
</html>
```



[http://blogs.sun.com/arungupta/entry/totd\\_135\\_jsf2\\_custom\\_components](http://blogs.sun.com/arungupta/entry/totd_135_jsf2_custom_components)

# Java Server Faces 2.0

## Integrated Ajax Support

- `f:ajax`

```
<h:commandButton  
    actionListener="#{sakilabean.findActors}"  
    value="submit">  
    <f:ajax execute="length"  
        render="actorTable totalActors"/>  
</h:commandButton>
```

[http://blogs.sun.com/arungupta/entry/totd\\_123\\_f\\_ajax\\_bean](http://blogs.sun.com/arungupta/entry/totd_123_f_ajax_bean)

# Java Server Faces 2.0

- “faces-config.xml” optional in common cases
  - <managed-bean> → @ManagedBean or @Named
  - Validator, Renderer, Listener, ...
  - Default navigation rules – match a view on the disk

```
@Named( "simplebean" )
```

```
public class SimpleBean {
```

```
    . . .  
}
```

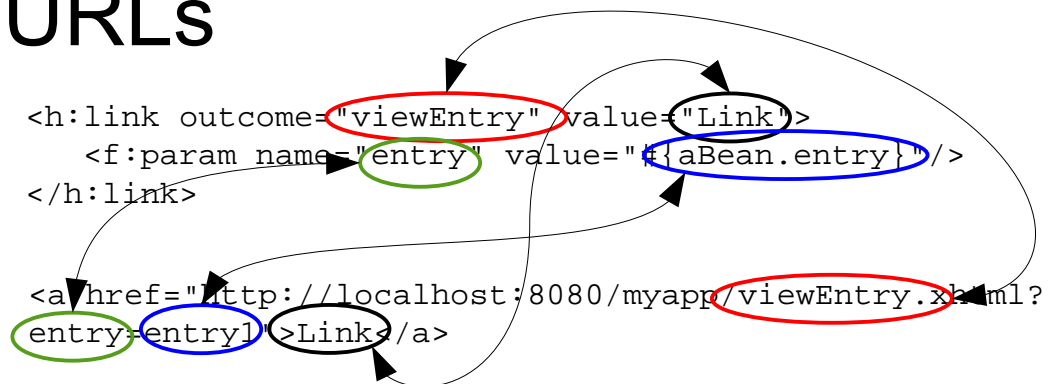
```
<h:commandButton action="show" value="submit" />
```

# Java Server Faces 2.0

Much more ...

- Runs on Servlet 2.5+
- Conditional navigation
- Project Stages
  - Development, UnitTest, SystemTest, Production
- Custom Scopes for Managed Beans
- Bookmarkable URLs
  - h:link, h:button

```
<navigation-case>
  <from-outcome>success</from-outcome>
  <to-view-id>/page2.xhtml</to-view-id>
  <!-- Only accept if the following condition
is true -->
  <if>#{foo.someCondition}</if>
</navigation-case>
```



# Java Persistence API 2 (JSR 317)

## Sophisticated mapping/modeling options

- Collection of basic types

```
@Entity
public class Person {
    @Id protected String ssn;
    protected String name;
    protected Date birthDate;
    . . .
    @ElementCollection
    @CollectionTable(name="ALIAS")
    protected Set<String> nickNames;
}
```

# Java Persistence API 2

## Sophisticated mapping/modeling options

- Collection of embeddables

```
@Embeddable public class Address {  
    String street;  
    String city;  
    String state;  
    . . .  
}
```

```
@Entity public class RichPerson extends Person {  
    . . .  
    @ElementCollection  
    protected Set<Address> vacationHomes;  
    . . .  
}
```

# Java Persistence API 2

## Sophisticated mapping/modeling options

- Multiple levels of embedding

```
@Embeddable public class ContactInfo {  
    @Embedded Address address;  
    . . .  
}
```

```
@Entity public class Employee {  
    @Id int empId;  
    String name;  
    ContactInfo contactInfo;  
    . . .  
}
```

# Java Persistence API 2

## Sophisticated mapping/modeling options

- Improved Map support

```
@Entity public class VideoStore {  
    @Id Integer storeId;  
    Address location;  
    . . .  
    @ElementCollection  
    Map<Movie, Integer> inventory;  
}
```

```
@Entity public class Movie {  
    @Id String title;  
    @String director;  
    . . .  
}
```

# Java Persistence API 2

## Expanded JPQL – CASE Expression

```
UPDATE Employee e
SET e.salary =
    CASE e.rating
        WHEN 1 THEN e.salary * 1.05
        WHEN 2 THEN e.salary * 1.02
        ELSE e.salary * 0.95
    END
```

# Java Persistence API 2

## Type-safe Criteria API

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<ResultType> cquery =
    cb.createQuery(ResultType.class);
Root<MyEntity> e = cquery.from(MyEntity.class);
Join<MyEntity, RelatedEntity> j = e.join(...);
...
cquery.select(...)
    .where(...)
    .orderBy(...)
    .groupBy(...);

TypedQuery<ResultType> tq = em.createQuery(cquery);
List<ResultType> result = tq.getResultList();
```

# Java Persistence API 2

Much more ...

- 2nd-level Cache
  - @Cacheable on entities
  - contain, evict ...
- New locking modes
  - PESSIMISTIC\_READ – grab shared lock
  - PESSIMISTIC\_WRITE – grab exclusive lock
  - PESSIMISTIC\_FORCE\_INCREMENT – update version
- Standard configuration options
  - `javax.persistence.jdbc.[driver | url | user | password]`

# Bean Validation (JSR 303)

- Tier-independent mechanism to define constraints for data validation
  - Represented by annotations
  - `javax.validation.*` package
- Integrated with JSF and JPA
  - JSF: `f:validateRequired`, `f:validateRegexp`
  - JPA: `pre-persist`, `pre-update`, and `pre-remove`
- `@NotNull(message="...")`, `@Max`, `@Min`, `@Size`
- Fully Extensible
  - `@Email` String recipient;

# JAX-RS 1.1

- Java API for building RESTful Web Services
- POJO based
- Annotation-driven
- Server-side API
- HTTP-centric

# JAX-RS 1.1

## Code Sample - Simple

```
@Path("helloworld")
public class HelloWorldResource {
    @Context UriInfo ui;

    @GET
    @Produces("text/plain")
    public String sayHello() {
        return "Hello World";
    }

    @Path("morning")
    public String morning() {
        return "Good Morning!";
    }
}
```

# JAX-RS 1.1

## Code Sample – Specifying Output MIME type

```
@Path("/helloworld")
@Produces("text/plain")
public class HelloWorldResource {
    @GET
    public String doGetAsPlainText() {
        . . .
    }

    @GET
    @Produces("text/html")
    public String doGetAsHtml() {
        . . .
    }

    @GET
    @Produces({
        "application/xml",
        "application/json"})
    public String doGetAsXmlOrJson() {
        . . .
    }
}
```

# JAX-RS 1.1

## Code Sample

```
import javax.inject.Inject;
import javax.enterprise.context.RequestScoped;

@RequestScoped
public class ActorResource {
    @Inject DatabaseBean db;

    public Actor getActor(int id) {
        return db.findActorById(id);
    }
}
```

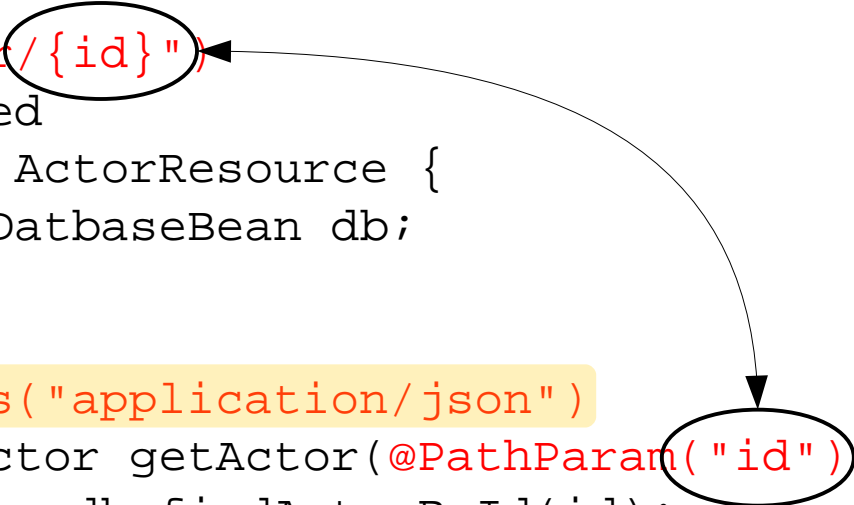
# JAX-RS 1.1

## Code Sample

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.PathParam;
import javax.inject.Inject;
import javax.enterprise.context.RequestScoped;

@Path("/actor/{id}")
@RequestScoped
public class ActorResource {
    @Inject DatabaseBean db;

    @GET
    @Produces("application/json")
    public Actor getActor(@PathParam("id") int id) {
        return db.findActorById(id);
    }
}
```



[http://blogs.sun.com/arungupta/entry/totd\\_124\\_using\\_cdi\\_jpa](http://blogs.sun.com/arungupta/entry/totd_124_using_cdi_jpa)

# JAX-RS 1.1

## Jersey Client-side API

- Consume HTTP-based RESTful Services
- Easy-to-use
  - Better than HttpURLConnection!
- Reuses JAX-RS API
  - Resources and URI are first-class citizens
- Not part of JAX-RS yet
  - `com.sun.jersey.api.client`

# JAX-RS 1.1

## Jersey Client API – Code Sample

```
Client client = Client.create();
```

```
WebResource resource = client.resource("...");
```

```
//curl http://example.com/base
```

```
String s = resource.get(String.class);
```

```
//curl -HAccept:text/plain http://example.com/base
```

```
String s = resource.  
    accept("text/plain").  
    get(String.class);
```

[http://blogs.sun.com/enterprisetechtips/entry/consuming\\_restful\\_web\\_services\\_with](http://blogs.sun.com/enterprisetechtips/entry/consuming_restful_web_services_with)

# JAX-RS 1.1

Much more ...

- Jersey is the Reference Implementation, included in GlassFish
  - RESTEasy, Restlet, CXF, Wink
- Hypermedia support (only in Jersey)
- Integration with Spring, Guice, Atom, ...
- WADL representation
  - Complete, Per resource
- Jersey 1.2 modules are OSGi compliant

# IDE Support for Java EE 6



# What is GlassFish ?



- A community
  - Users, Partners, Testers, Developers, ...
  - Started in 2005 on java.net
- Application Server
  - Open Source (CDDL & GPL v2)
  - Java EE Reference Implementation

# GlassFish Distributions

Distribution	License	Features
GlassFish Open Source Edition 3.0.1	CDDL & GPLv2	<ul style="list-style-type: none"> <li>• Java EE 6 Compatibility</li> <li>• No Clustering</li> <li>• Clustering planned in 3.1</li> <li>• mod_jk for load balancing</li> </ul>
GlassFish Open Source Edition 2.1.1	CDDL & GPLv2	<ul style="list-style-type: none"> <li>• Java EE 5 Compatibility</li> <li>• In memory replication</li> <li>• mod_loadbalancer</li> </ul>
Oracle GlassFish Server 3.0.1	Commercial	<ul style="list-style-type: none"> <li>• GlassFish Open Source Edition 3.0.1</li> <li>• Enterprise Manager (Server Control??)</li> <li>• Clustering planned in 3.1</li> </ul>
Oracle GlassFish Server 2.1.1	Commercial	<ul style="list-style-type: none"> <li>• GlassFish Open Source Edition 2.1.1</li> <li>• Enterprise Manager</li> <li>• HADB</li> </ul>



# GlassFish 3

- **Modular**
  - Maven 2 – Build & Module description
  - Felix – OSGi runtime (200+ bundles)
  - Allow any type of Container to be plugged
    - Start Container and Services on demand
- **Embeddable**: runs in-VM
- **Extensible**
  - Rails, Grails, Django, ...
  - Administration, Monitoring, Logging, Deployment, ...

[http://blogs.sun.com/dochez/entry/glassfish\\_v3\\_extensions\\_part\\_5](http://blogs.sun.com/dochez/entry/glassfish_v3_extensions_part_5)

# GlassFish™ v3 Administration Console

## Tree

- Common Tasks
- Registration
- GlassFish News
- Enterprise Server
- Applications
- Lifecycle Modules
- Resources
  - JDBC
  - Connectors
  - Resource Adapter Configs
  - JMS Resources
  - JavaMail Sessions
  - JNDI
- Configuration
  - JVM Settings
  - Logger Settings
  - Web Container
  - EJB Container
  - Ruby Container
  - Java Message Service
  - Security

- Registration
- GlassFish News
- Subscriptions

## Deployment

- List Deployed Applications
- Deploy an Application

- Quick Start Guide
- Administration Guide
- Developer's Guide
- Application Deployment Guide

## Update Center

- Installed Components
- Available Updates
- Available Add-Ons

## Other Tasks

- Create New JDBC Connection Pool



**Run GlassFish Faster**  
Download the white paper and learn how to optimize GlassFish performance in a production environment. » [Get It Now](#)

- [Get production support.](#)
- Stay current with GlassFish news at [The Aquarium](#) or learn about GlassFish deployments.
- Get community support, [Developer Expert Assistance](#) or [Expert-to-engineer web training for Java EE 5.](#)
- Join [Project GlassFish](#)
- Learn about upcoming Java EE 6 Release in this [webinar.](#)

# Boost your productivity

## Retain session across deployment

asadmin redeploy -properties keepSessions=true helloworld.war

The image shows two overlapping screenshots from the GlassFish v3 administration console. The left screenshot shows the 'Application Server' configuration page with a red circle around the 'Preserve Sessions Across Redeployment' checkbox. The right screenshot shows the 'Common' tab of the configuration page, also with a red circle around the 'Preserve Sessions Across Redeployment' checkbox.

**Application Server Configuration:**

- Domain Name: domain1
- Domain Directory: /Users/arun
- Admin Name: admin
- Admin Password: [Empty]
- Server Port Number: 8080
- Admin Server Port Number: 4848
- Preserve Sessions Across Redeployment

**Common Tab Configuration:**

- Server Type: GlassFish v3
- Location: localhost:8080
- Domains folder: /Users/arun/tools/glassfish/v3/74b/glassfish
- Domain Name: domain1
- Enable Comet Support
- Enable HTTP Monitor
- Enable JDBC Driver Deployment
- Preserve Sessions Across Redeployment
- Start Registered Derby Server

# Boost your productivity

## Deploy-on-Save

### Publishing

Modify settings for publishing.

- Never publish automatically
- Automatically publish when resources change

Publishing interval (in seconds):

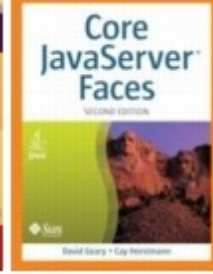
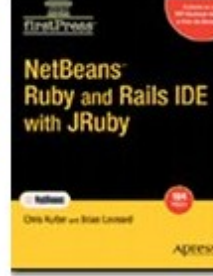
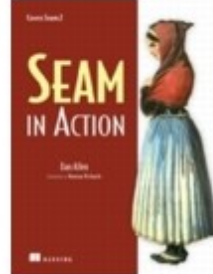
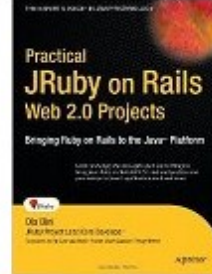
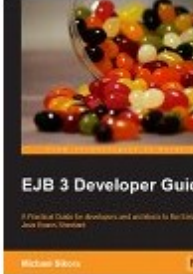
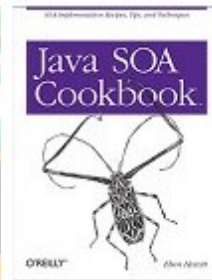
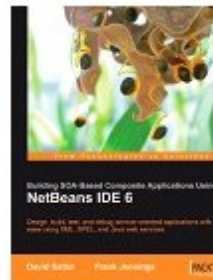
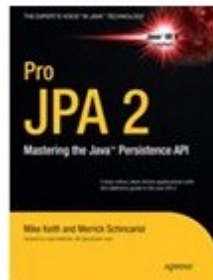
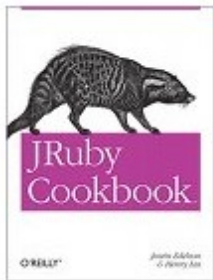
  

### Deploy on Save

If selected, files are compiled and deployed when you save. This option saves you time when you run or debug your code.

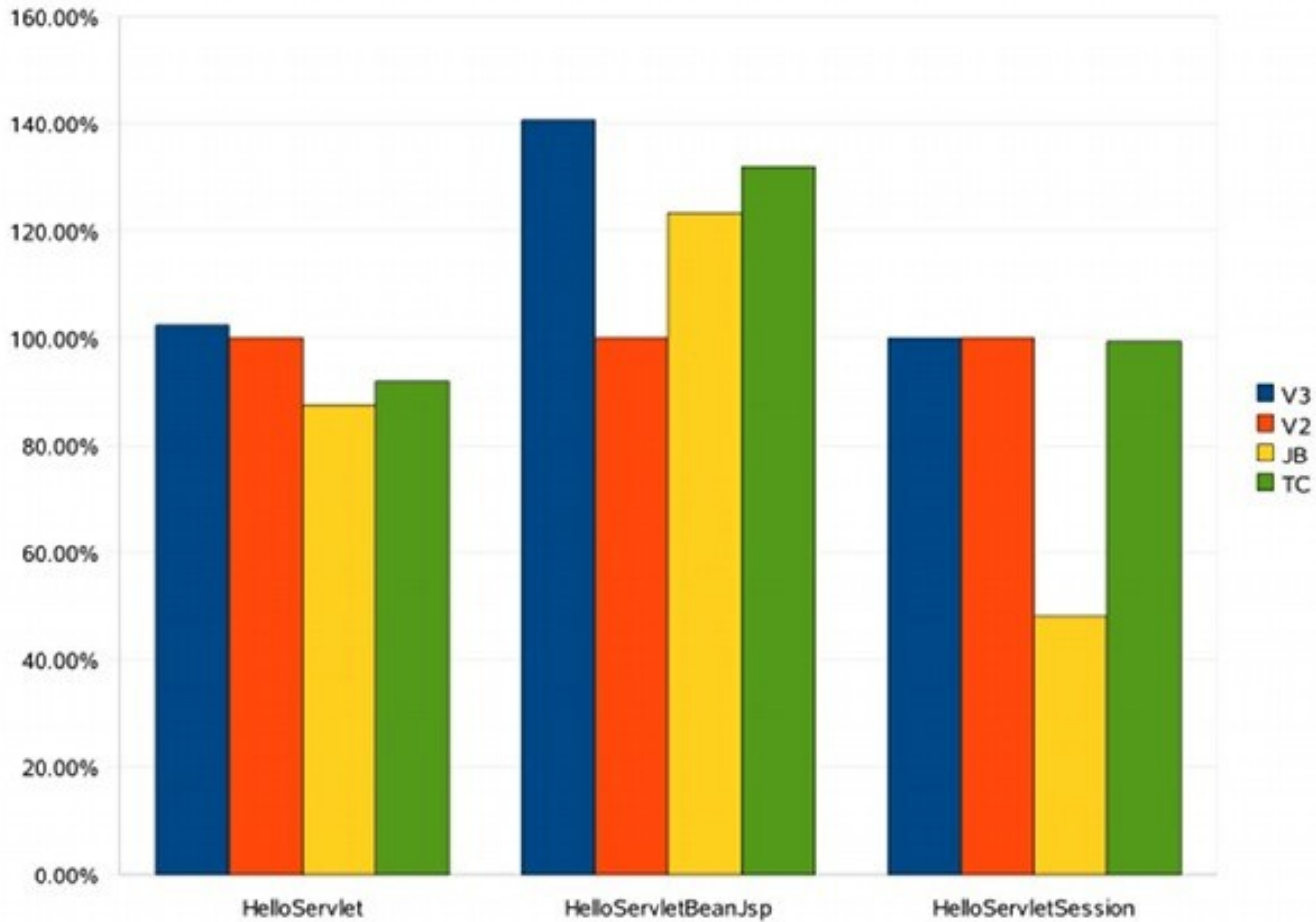
VM Options:

(used for running main classes or unit tests)



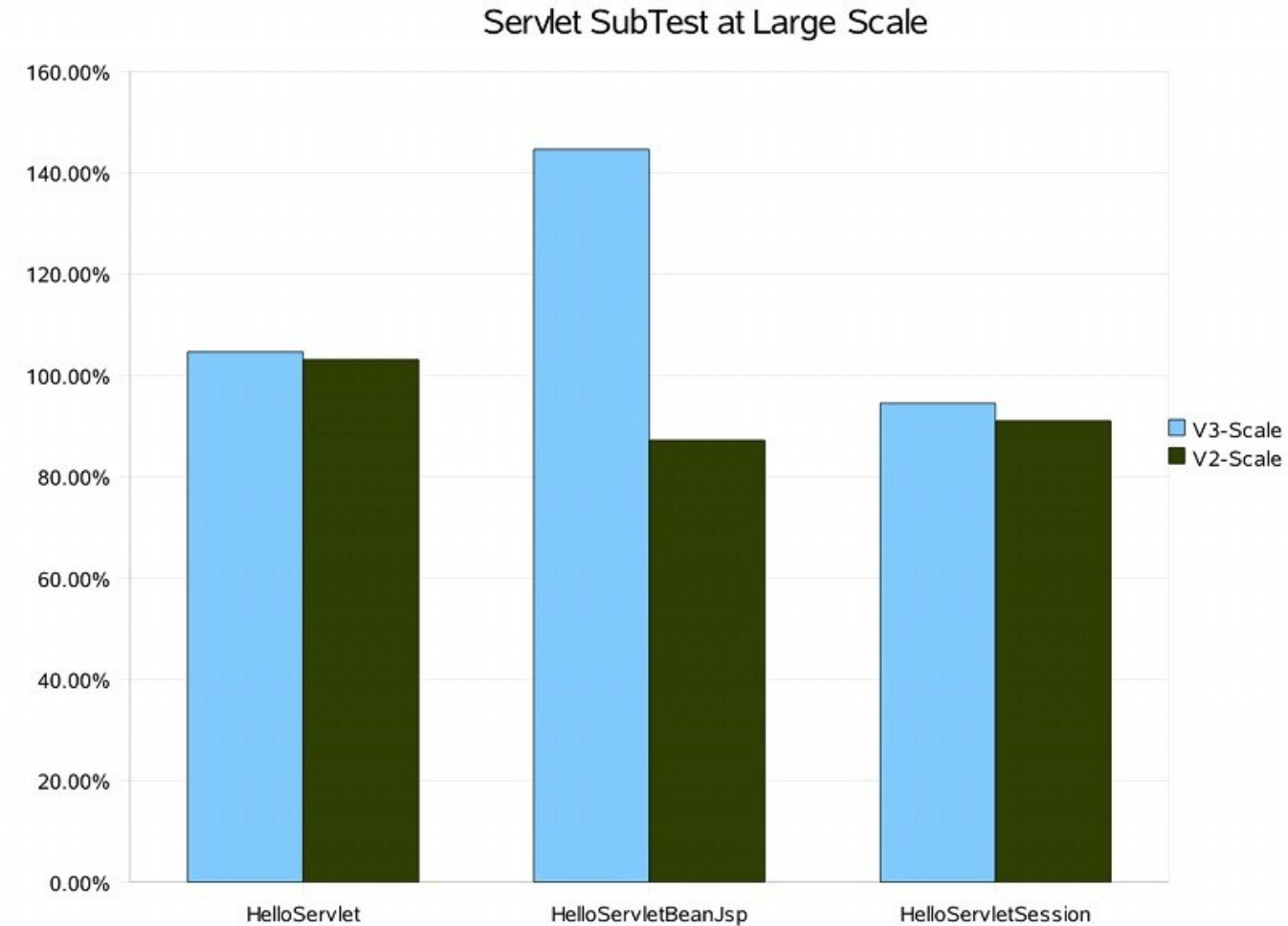
# GlassFish 3 Performance

Servlet Subtest Small Scale



<http://weblogs.java.net/blog/sdo/archive/2009/12/08/first-look-v3-performance>

# GlassFish 3 Performance



<http://weblogs.java.net/blog/sdo/archive/2009/12/08/first-look-v3-performance>

# GlassFish 3.0.1

- First Oracle-branded release of GlassFish
- Additional platform support
  - Oracle Enterprise Linux 4 & 5 (32 & 64-bit)
  - Red Hat Enterprise Linux 64-bit
  - Window 2008 R2 (32 & 64-bit)
  - HP-UX 11i, (32 & 64-bit)
  - JRockit 6 Update 17
- 100+ bugfixes

<http://www.oracle.com/technetwork/middleware/glassfish/overview/index.html>

# GlassFish 3.1

- Main Features
  - Clustering and Centralized Administration
  - High Availability
- Other ...
  - Application Versioning
  - Embedded (extensive)
  - Admin Console based on RESTful API

<http://wikis.sun.com/display/glassfish/GlassFishv3.1>

# GlassFish Roadmap

- Oracle committed to GlassFish community
- No changes to the operation of GlassFish OSS
- Remains transparent and participatory
- GlassFish strengthened by Oracle stewardship
  - Oracle wants to grow the community
- Customer and community driven roadmap

<http://glassfish.org/roadmap>

# GlassFish Roadmap Detail

## CY2010

- **GlassFish 3.0.1**
  - Branding, Patches
  - Multi-Lingual Release
  - Adds Value-Added Features to Oracle GlassFish Server
  - Base Interop w/ Oracle Middleware Products
- **GlassFish 3.1**
  - Centralized Administration / Clusters
  - High Availability / State Replication
  - Value Added Features, like Coherence Support

## CY2011

- **GlassFish 3.2**
  - Improved Cluster/HA administration
  - Better Integration w/ Oracle Identity Management
  - Virtualization Support
  - Some Java EE 6 Specification Updates, some Java EE 7 EE

## CY2012

- **GlassFish 4**
  - Common Server Platform – more shared best of breed with WebLogic Server
  - Java EE 7

# GlassFish Distributions

- GlassFish.org
  - Community Site
  - Mailing lists, Forums, Wikis, Know-How
  - OSS Sources
  - OSS Binary Distribution

GlassFish Server Open Source Edition

# GlassFish Distributions

- Oracle.com
  - Commercial Site
  - Formal documentation and Support
  - Includes Value-adds
  - Oracle distribution with standard Oracle Licenses
    - Evaluation – OTN Evaluation License
    - Production – Deployment License

Oracle GlassFish Server

# Key Changes Under Oracle

- *Not Changed*

- Open Source (most GPL/CDDL)
- Non-Oracle committers
- Transparent development
- OSS binaries at [glassfish.org](http://glassfish.org)

- *Not Changed*

- Add-ons remain closed source

- *Changed*

- New licenses at [oracle.com](http://oracle.com) for Trial and Deployment
- Add-ons easier to try and bundled in Oracle distribution

# Fusion Middleware Integration Strategy

- Commercial version will have integrations with Fusion Middleware
  - Certification on JRockit
  - Integration with Coherence and TopLink
- Fusion Middleware and Fusion Applications currently not planned to be certified on GlassFish
- Initial integrations will be interoperability
  - Web services, Web services policy, Identity Management (OAM)

# References

- [glassfish.org](http://glassfish.org)
- [blogs.sun.com/theaquarium](http://blogs.sun.com/theaquarium)
- [oracle.com/goto/glassfish](http://oracle.com/goto/glassfish)
- [glassfish.org/roadmap](http://glassfish.org/roadmap)
- Follow [@glassfish](https://twitter.com/glassfish)



**ORACLE®**

**Java EE 6 & GlassFish 3:  
Light-weight, Extensible, Powerful**

Arun Gupta, Java EE & GlassFish Guy  
[blogs.sun.com/arungupta](http://blogs.sun.com/arungupta), @arungupta