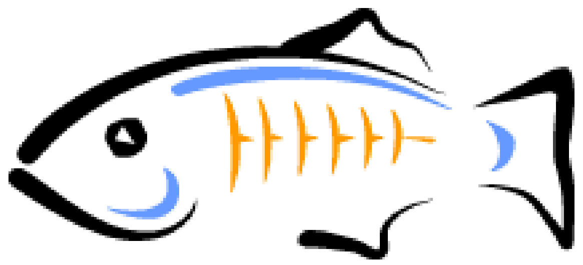
GlassFish v3
The future of Java EE is here

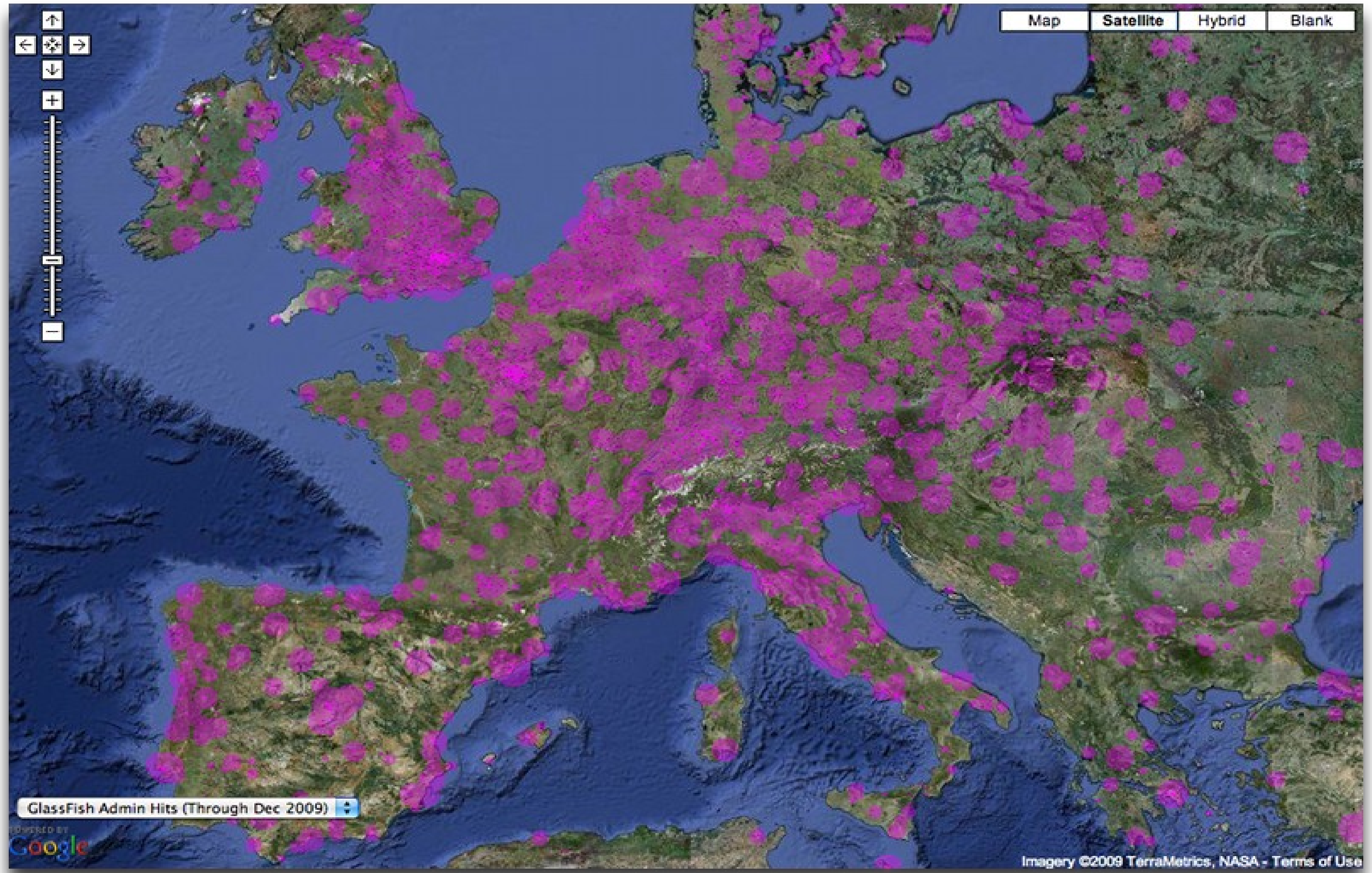


Alexis Moussine-Pouchkine
GlassFish Team

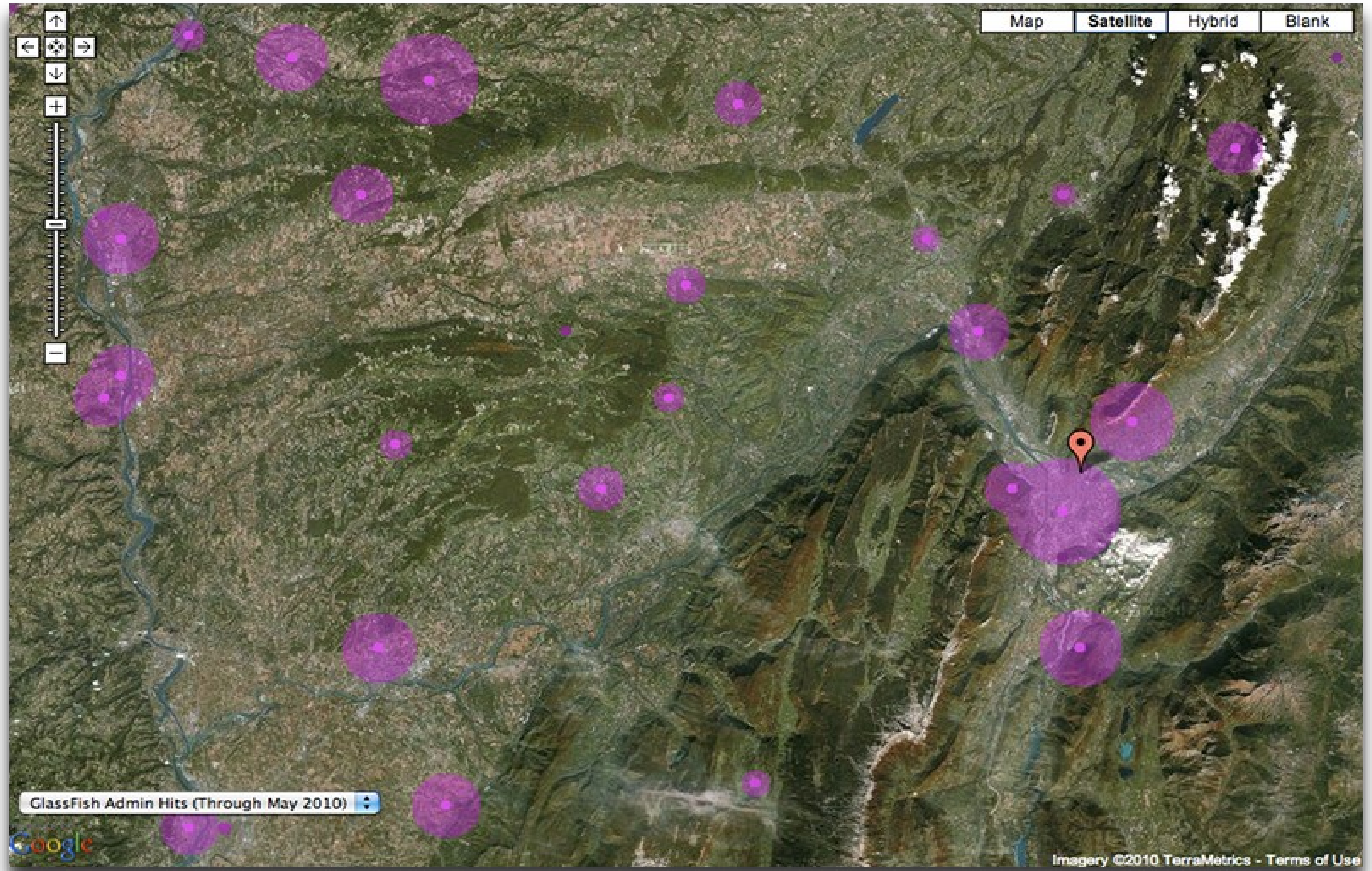
This is no science fiction

Java EE 6 and GlassFish v3 shipped **final releases** on December 10th 2009

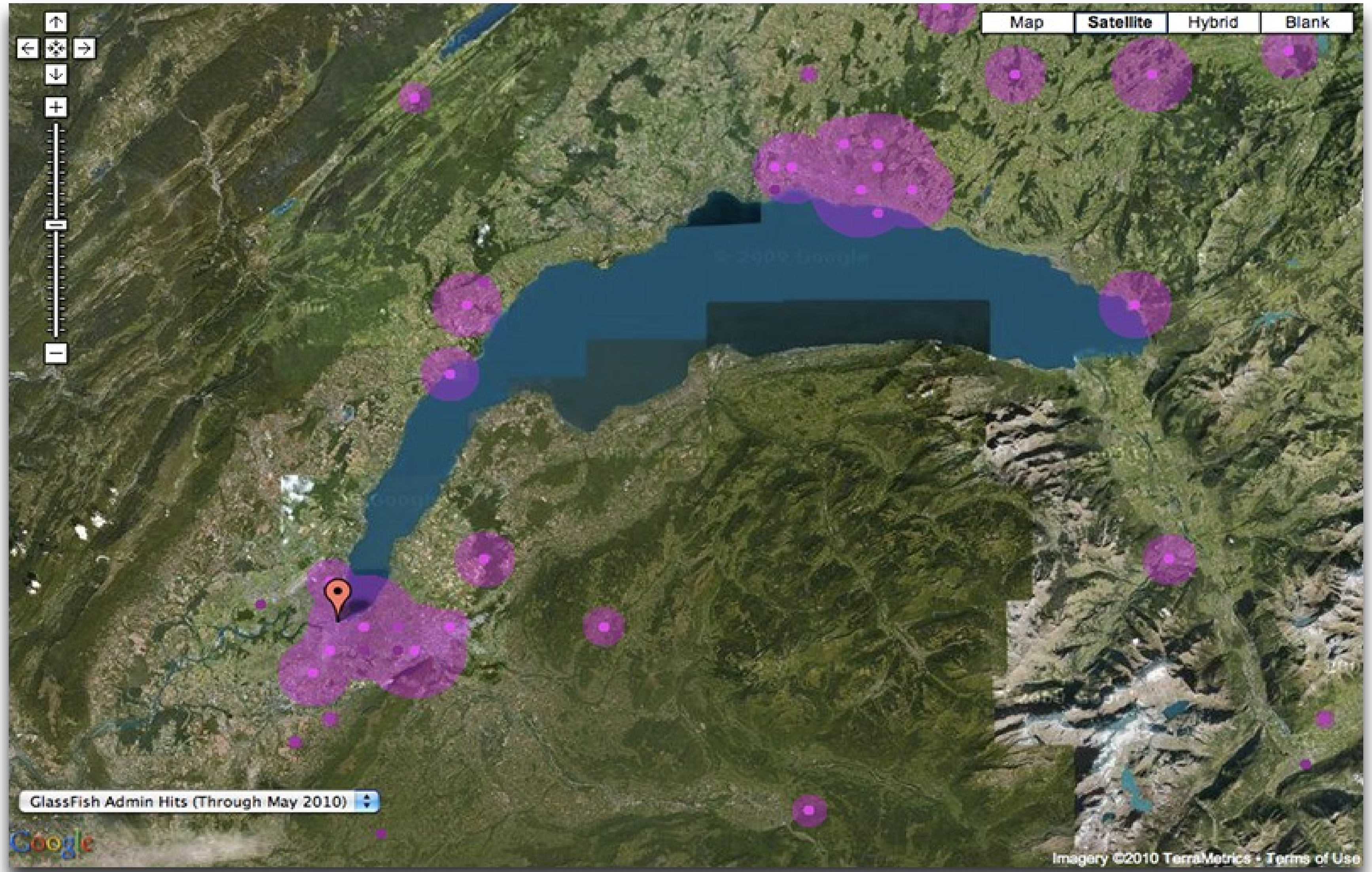
GlassFish Around You


Map
Satellite
Hybrid
Blank
GlassFish Admin Hits (Through Dec 2009)
POWERED BY Google
Imagery ©2009 TerraMetrics, NASA - Terms of Use

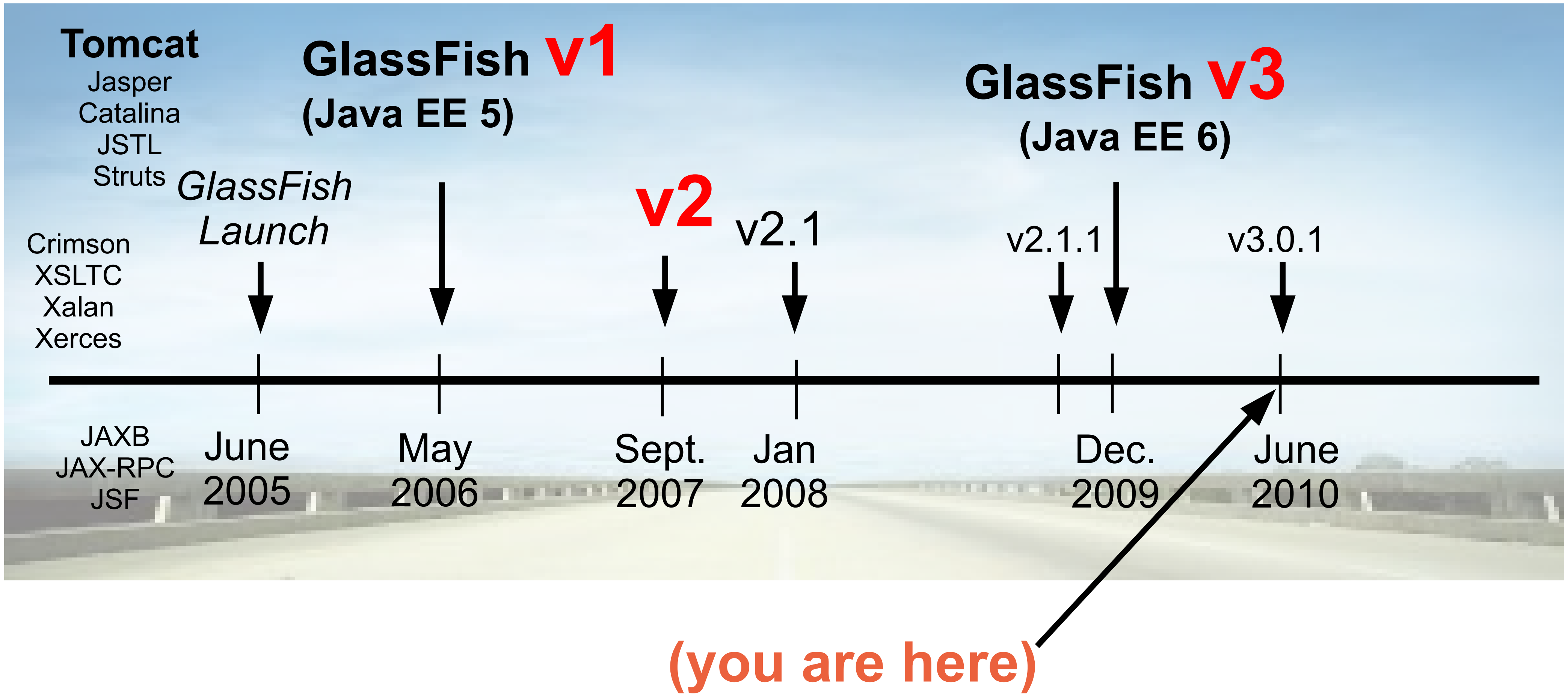
GlassFish Around You


Map
Satellite
Hybrid
Blank
GlassFish Admin Hits (Through May 2010)
Google
Imagery ©2010 TerraMetrics - Terms of Use

GlassFish Around You


Map
Satellite
Hybrid
Blank
GlassFish Admin Hits (Through May 2010)
Google
Imagery ©2010 TerraMetrics · Terms of Use

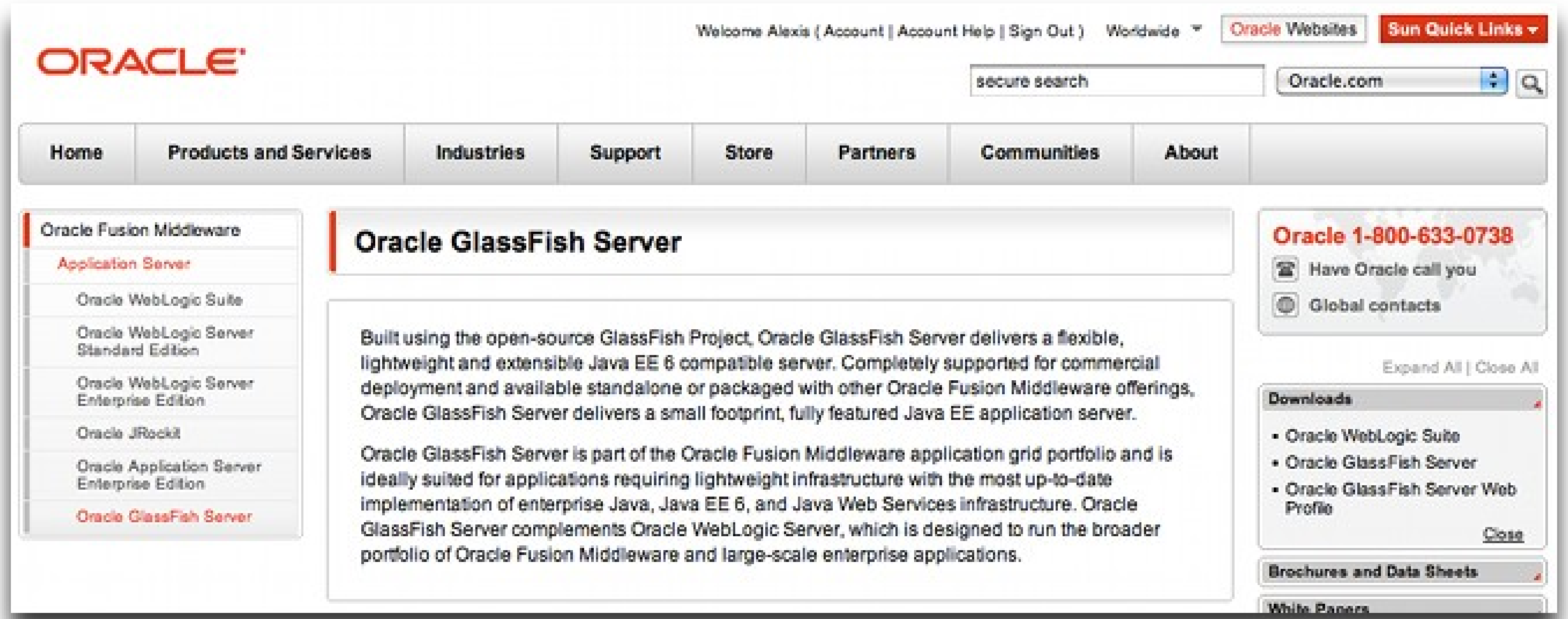
Some History and Context



GlassFish

- A Community
  - Users, Partners, Testers, Developers
  - Started in 2005 on java.net
  - Sub-projects
    - Jersey (JAX-RS), Metro (JAX-WS), Grizzly (nio), Atmosphere (Comet), OpenMQ (JMS), and scripting
- Application Server
  - Enterprise Quality and Open Source
  - Java EE 5 / 6 Reference Implementation
  - Full Commercial Support from Oracle

Oracle GlassFish Server



Oracle GlassFish Server

Built using the open-source GlassFish Project, Oracle GlassFish Server delivers a flexible, lightweight and extensible Java EE 6 compatible server. Completely supported for commercial deployment and available standalone or packaged with other Oracle Fusion Middleware offerings, Oracle GlassFish Server delivers a small footprint, fully featured Java EE application server.

Oracle GlassFish Server is part of the Oracle Fusion Middleware application grid portfolio and is ideally suited for applications requiring lightweight infrastructure with the most up-to-date implementation of enterprise Java, Java EE 6, and Java Web Services infrastructure. Oracle GlassFish Server complements Oracle WebLogic Server, which is designed to run the broader portfolio of Oracle Fusion Middleware and large-scale enterprise applications.

GlassFish going forward

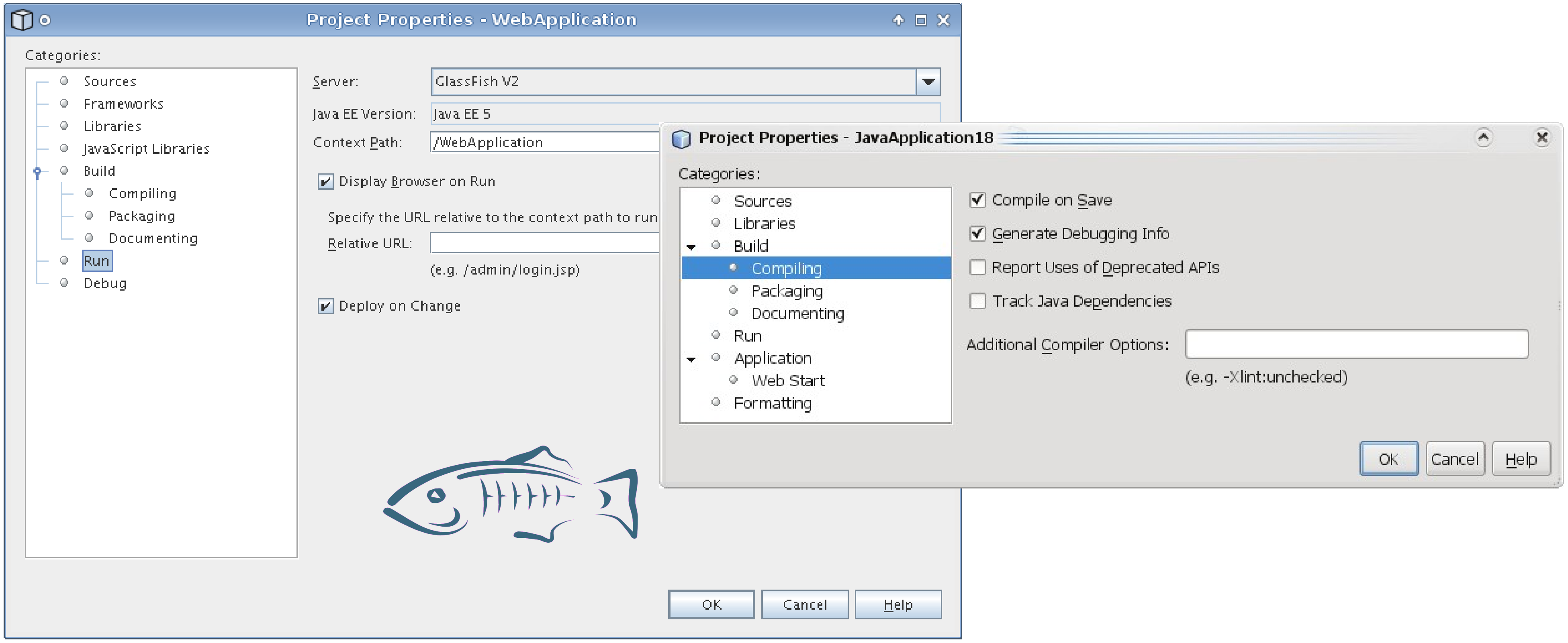
- No change to operation of open source project
  - GlassFish Open Source Edition under existing license
  - Remains transparent and participatory
  - Strengthened by Oracle leadership
  - Customer and community driven product roadmap
- GlassFish 3.0.1 shipped June 2010 as planned
  - Additional platforms, jrockit, RHEL, 64-bit JVM's
  - 100+ bug fixes, ...
- Feature releases
  - GlassFish v3.1 in 2010, v3.2 in 2011, v4 in 2012
  - Clustering, centralized admin, Coherence, virtualization
  - Details at http://glassfish.org/roadmap

Demo

Painless development with GlassFish v3

Painless Java EE development !

- Incremental compile of all Java EE artifacts
- Auto-deploy of all Java EE and static artifacts



Project Properties - WebApplication

Categories:

- Sources
- Frameworks
- Libraries
- JavaScript Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Debug

Server: GlassFish V2

Java EE Version: Java EE 5

Context Path: /WebApplication

[✓] Display Browser on Run

Specify the URL relative to the context path to run

Relative URL:

(e.g. /admin/login.jsp)

[✓] Deploy on Change

Project Properties - JavaApplication18

Categories:

- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Application
  - Web Start
- Formatting

[✓] Compile on Save

[✓] Generate Debugging Info

[ ] Report Uses of Deprecated APIs

[ ] Track Java Dependencies

Additional Compiler Options:

(e.g. -Xlint:unchecked)

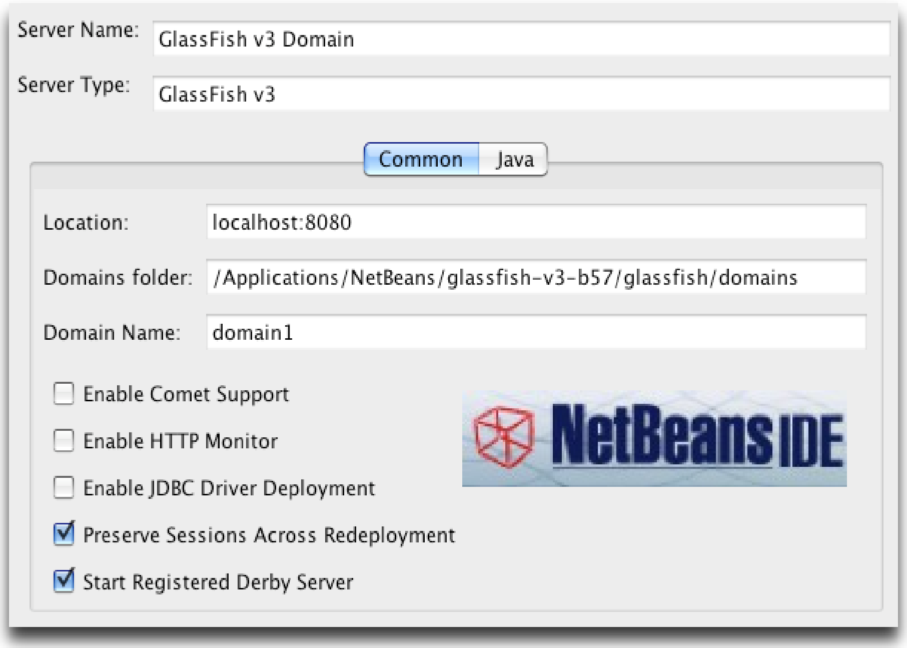
Session Retention

- Deployment option to maintain stateful sessions across re-deployments

```
$ asadmin redeploy --properties
  keepSessions=true myapp.war
```

- Greatly simplifies the development paradigm

- Integrated in IDEs

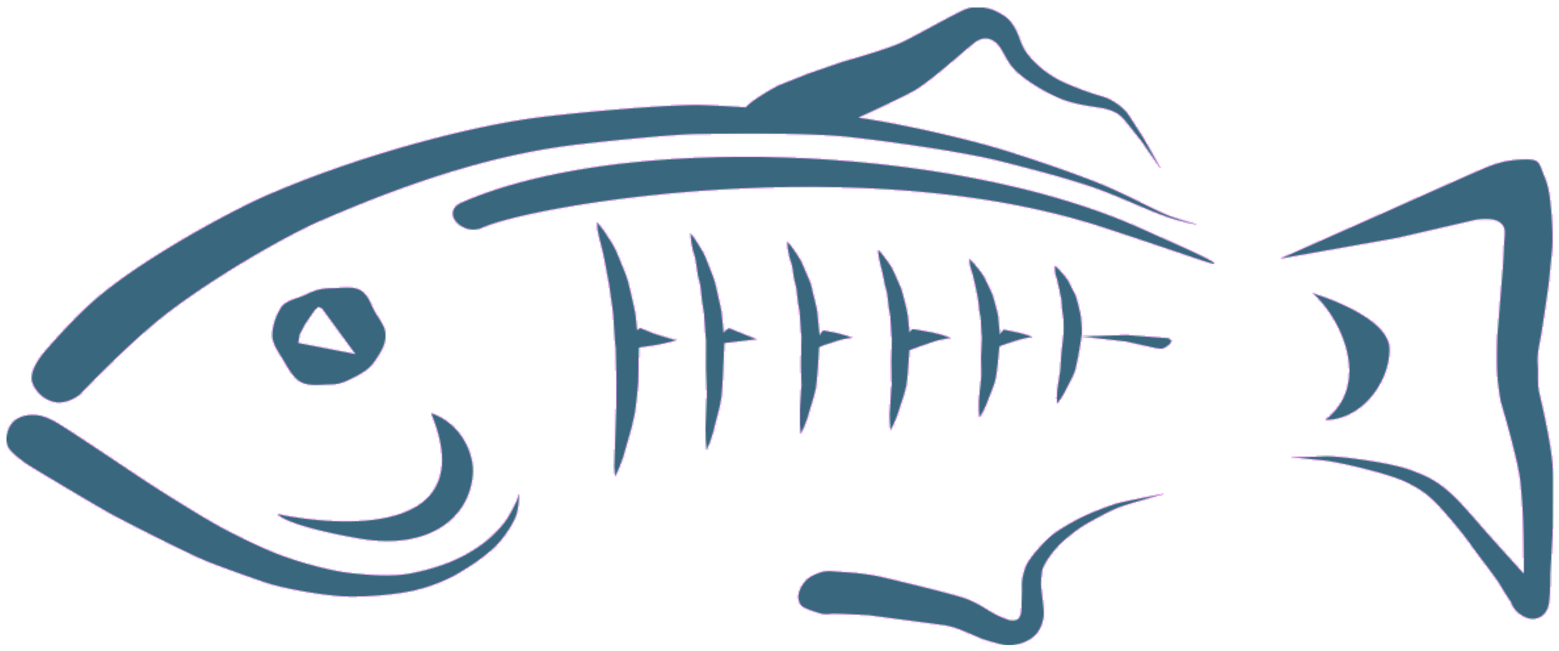


Server Name: GlassFish v3 Domain
Server Type: GlassFish v3

Location: localhost:8080
Domains folder: /Applications/NetBeans/glassfish-v3-b57/glassfish/domains
Domain Name: domain1

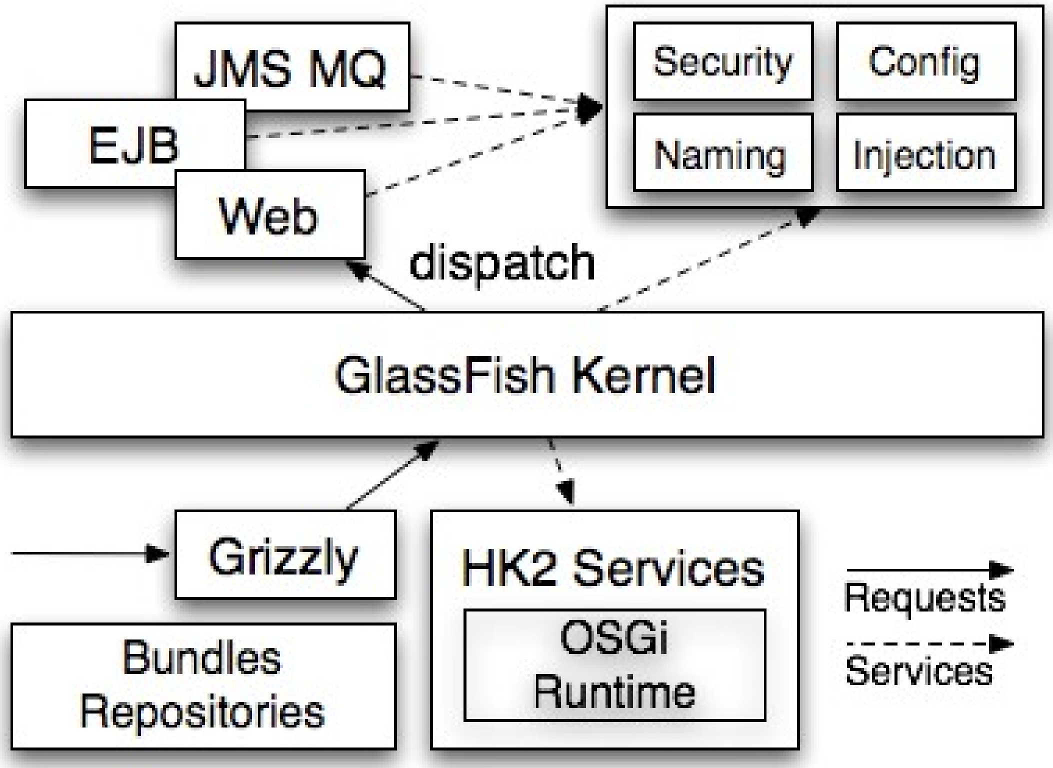
- [ ] Enable Comet Support
- [ ] Enable HTTP Monitor
- [ ] Enable JDBC Driver Deployment
- [✓] Preserve Sessions Across Redeployment
- [✓] Start Registered Derby Server

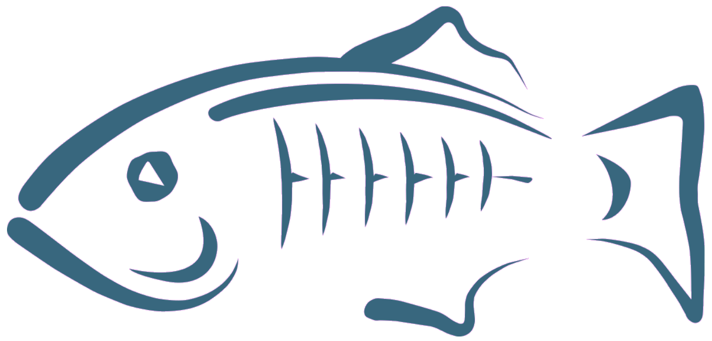
Introducing GlassFish v3

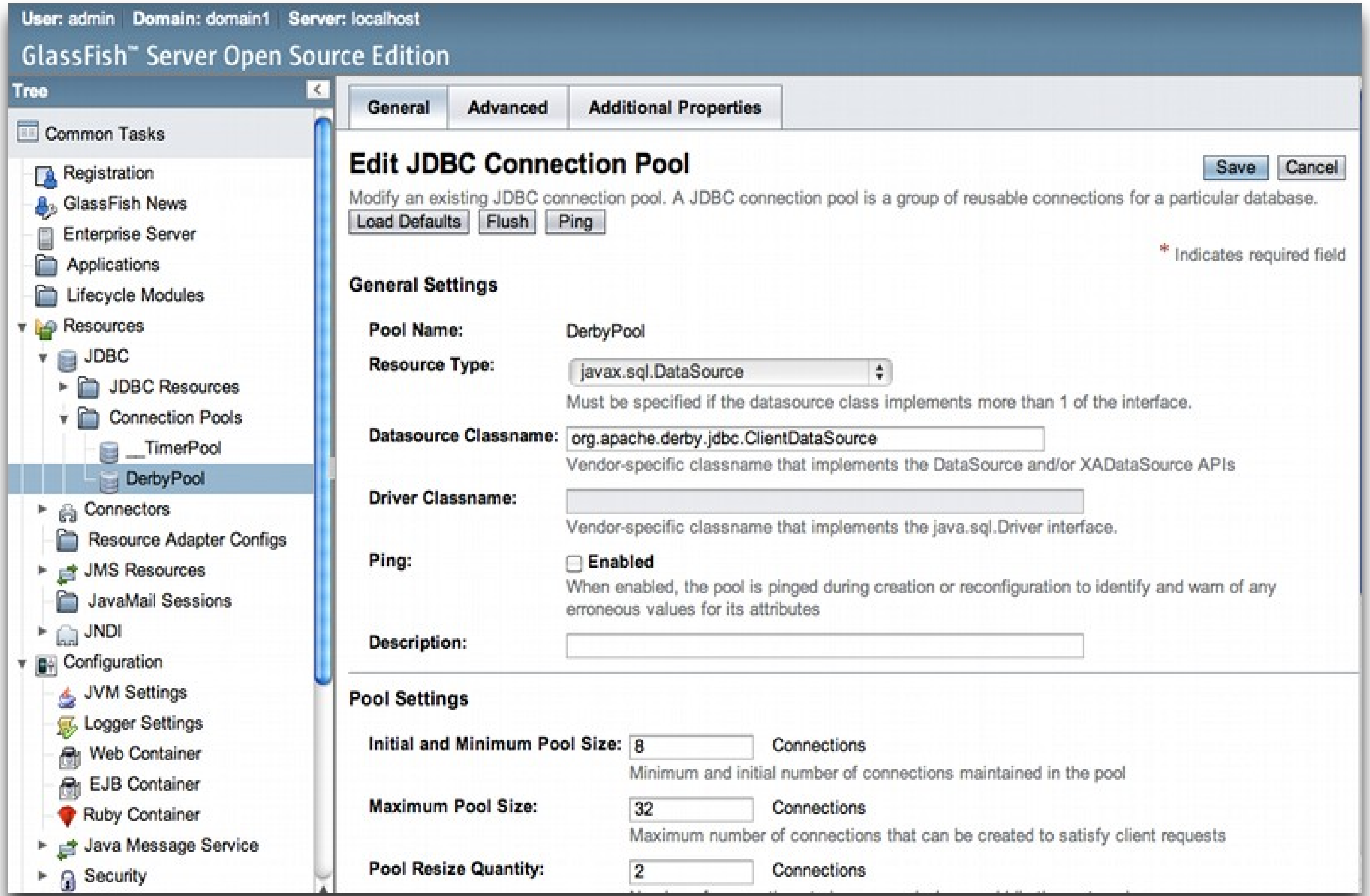


Modular and Dynamic

- Modular : Apache Felix (OSGi)
- Extensible : HK2
- Yet very Fast !







Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

* Indicates required field

General Settings

Pool Name: DerbyPool

Resource Type: javax.sql.DataSource
Must be specified if the datasource class implements more than 1 of the interface.

Datasource Classname: org.apache.derby.jdbc.ClientDataSource
Vendor-specific classname that implements the DataSource and/or XADataSource APIs

Driver Classname:
Vendor-specific classname that implements the java.sql.Driver interface.

Ping: ☐ Enabled
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

Description:

Pool Settings

Initial and Minimum Pool Size: 8 Connections
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: 32 Connections
Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity: 2 Connections

Java EE, a brief History

	Enterprise Application	Robust Scalable	Web Services	Ease of development	Ease of development (web)
Project JPE	**J2EE 1.2**	**J2EE 1.3**	**J2EE 1.4**	**Java EE 5**	**Java EE 6**
	Servlet JSP EJB JMS RMI/IIOP	CMP JCA	WS Management Deployment	Annotations EJB 3 JPA 1.0 WS-* JSF	EJB 3.1 JPA 2.0 Servlet 3.0 JSF 2.0 JAX-RS 1.1 CDI 1.0 @Inject Bean Validat° **Web Profile** **Managed Bean**
May 1998	Dec 1999 **10 specs**	Sept 2001 **13 specs**	Nov 2003 **20 specs**	May 2006 **23 specs**	Q4 2009 **28 specs**

Java EE 6 – What's New?

- Several new APIs
- Web Profile
- Extensibility & Pluggability
- Dependency Injection
- Improvement to many APIs

New and improved specifications

- EJB 3.1
- JPA 2.0
- Servlet 3.0
- JSF 2.0
- JAX-RS 1.1
- Connectors 1.6
- Bean Validation 1.0
- DI 1.0
- CDI 1.0
- Managed Beans 1.0
- Interceptors 1.1
- JAX-WS 2.2
- JSR-109 1.3
- JSP 2.2 / EL 2.2
- JSR-250 1.1

JAX-RS

- RESTful web services API
- Already widely adopted
- Really a general, high-level HTTP API
- Annotation-based programming model
- Programmatic API when needed
- JAX-RS 1.1 integration with EJBs

JAX-RS sample code

```
@Path("widgets/{id}")
@Produces("application/widgets+xml")
public class WidgetResource {
    public WidgetResource(
            @PathParam("id") String id) {
        ...
    }

    @GET
    Widget getWidget() {
        ...
    }
}
```

Bean Validation 1.0

```
public class Address {
   @NotNull @Size(max=30,
       message="longer than {max} characters")
   private String street1;
   ...
   @NotNull @Valid
   private Country country;
}

public class Country {
   @NotNull @Size(max=20)
   private String name;
   ...
}
```

request recursive object graph validation

Build your own!

```
@Size(min=5, max=5)
@ConstraintValidator(ZipcodeValidator.class)
@Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
public @interface ZipCode {
    String message() default "Wrong zipcode";
    String[] groups() default {};
}
```

Integrated in JPA and JSF

Bootstrap APIs

Java EE 6 Web Profile

- Servlet 3.0
- JSP 2.2 / EL 2.2
- JSR-45 1.0
- JSTL 1.2
- JSF 2.0
- Bean Validation 1.0
- EJB 3.1 Lite
- JPA 2.0
- JTA 1.1
- DI 1.0
- CDI 1.0
- Managed Beans 1.0
- Interceptors 1.1
- JSR-250 1.1

Modular Web Applications

- Libraries can contain `web-fragment.xml`
- `web.xml` is optional
- `@WebServlet`, `@WebFilter` annotations
- `ServletContainerInitializer` interface
- Programmatic registration
- Resource jars

JSF 2.0

- Standardized facelets
- Auto-discovery of component libraries
- Composite components
- Ajax support with partial views
- Even a JavaScript API !

EJB 3.1

- `@Singleton` beans
- `@Startup` beans
- `@Asynchronous` invocations
- `@Schedule` tasks
- `EJBContainer` API works on Java SE
- Define EJBs directly inside a web app

Packaging in a war

foo.ear

```
lib/foo_common.jar
  com/acme/Foo.class

foo_web.war
  WEB-INF/web.xml
  WEB-INF/classes
      com/acme/FooServlet.class

foo_ejb.jar
  com/acme/FooEJB.class
  com/acme/FooEJBLocal.class
```

foo.war

```
WEB-INF/classes
   com/acme/Foo.class
   com/acme/FooServlet.class
   com/acme/FooEJB.class
```

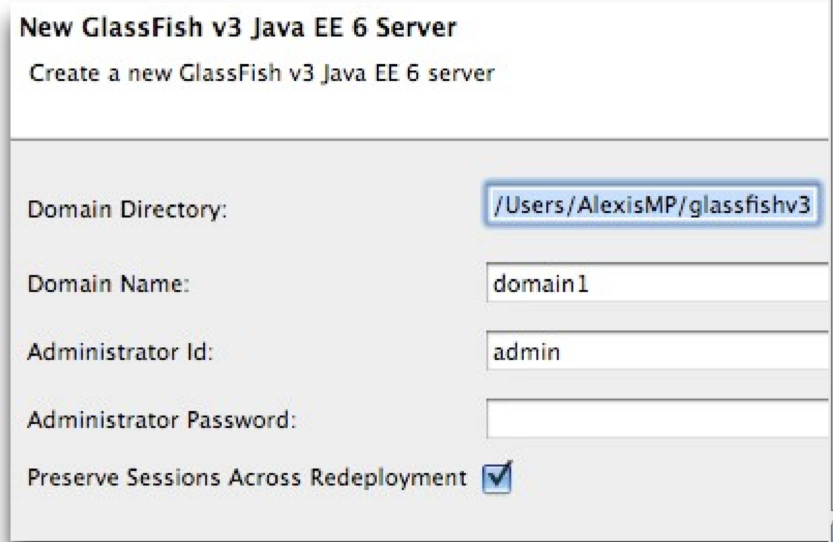
Demo

Painless (Java EE 6) development

Yes, Eclipse too !

GlassFish Tools Bundle For Eclipse 1.2
release candidate build 1

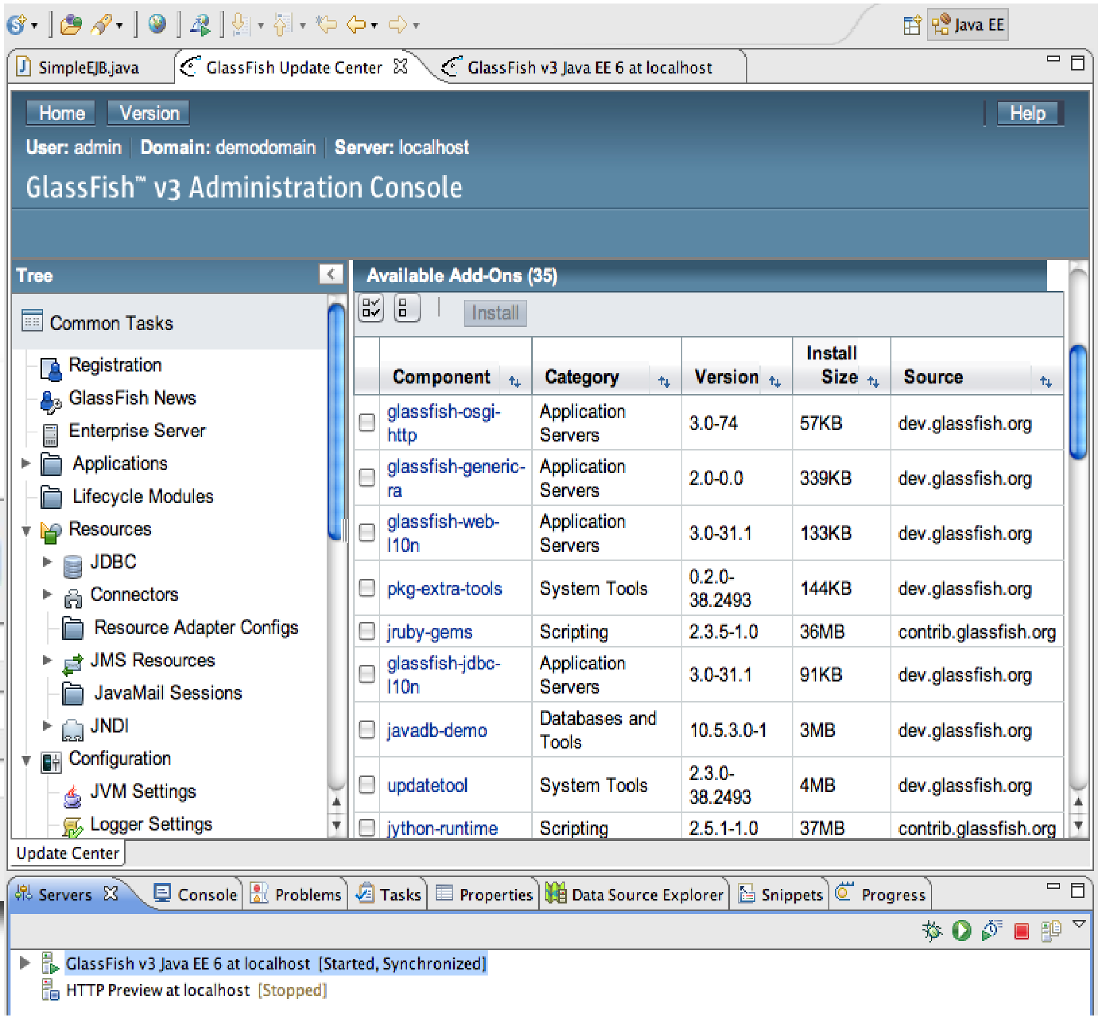
(c) Copyright Eclipse contributors and others, 2000,2009. All rights reserved. Java and all Java related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S, other countries, or both. Eclipse is a trademark of the Eclipse Foundation, Inc.



New GlassFish v3 Java EE 6 Server

Create a new GlassFish v3 Java EE 6 server

Domain Directory:	/Users/AlexisMP/glassfishv3
Domain Name:	domain1
Administrator Id:	admin
Administrator Password:	
Preserve Sessions Across Redeployment	✓



User: admin | Domain: demodomain | Server: localhost

GlassFish™ v3 Administration Console

Available Add-Ons (35)

Component	Category	Version	Install Size	Source
glassfish-osgi-http	Application Servers	3.0-74	57KB	dev.glassfish.org
glassfish-generic-ra	Application Servers	2.0-0.0	339KB	dev.glassfish.org
glassfish-web-l10n	Application Servers	3.0-31.1	133KB	dev.glassfish.org
pkg-extra-tools	System Tools	0.2.0-38.2493	144KB	dev.glassfish.org
jruby-gems	Scripting	2.3.5-1.0	36MB	contrib.glassfish.org
glassfish-jdbc-l10n	Application Servers	3.0-31.1	91KB	dev.glassfish.org
javadb-demo	Databases and Tools	10.5.3.0-1	3MB	dev.glassfish.org
updatetool	System Tools	2.3.0-38.2493	4MB	dev.glassfish.org
jython-runtime	Scripting	2.5.1-1.0	37MB	contrib.glassfish.org

GlassFish v3 Java EE 6 at localhost [Started, Synchronized]
HTTP Preview at localhost [Stopped]

GlassFish Tools Bundle for Eclipse : http://download.java.net/glassfish/eclipse/

More Painless Development

- Fast auto-deploy of all Java EE and static artifacts
- Application runner
  - `java -jar glassfish.jar toto.war`
- Integration with maven 2
  - `mvn gf:run`, `gf:start`, `gf:deploy`, ...
- Containers can be added/removed dynamically
- Excellent Tools integration

JPA 2.0

- Support for collections of basic types and embeddable objects
- JPQL enhancements
  - e.g. CASE WHEN, NULLIF
- Pessimistic locking
- Criteria API for dynamic query construction

Criteria API

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Book> query =
                     cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);

query.select(book)
     .where(cb.equal(book.get("description"), ""));
```

```
SELECT b
FROM Book b
WHERE b.description IS EMPTY
```

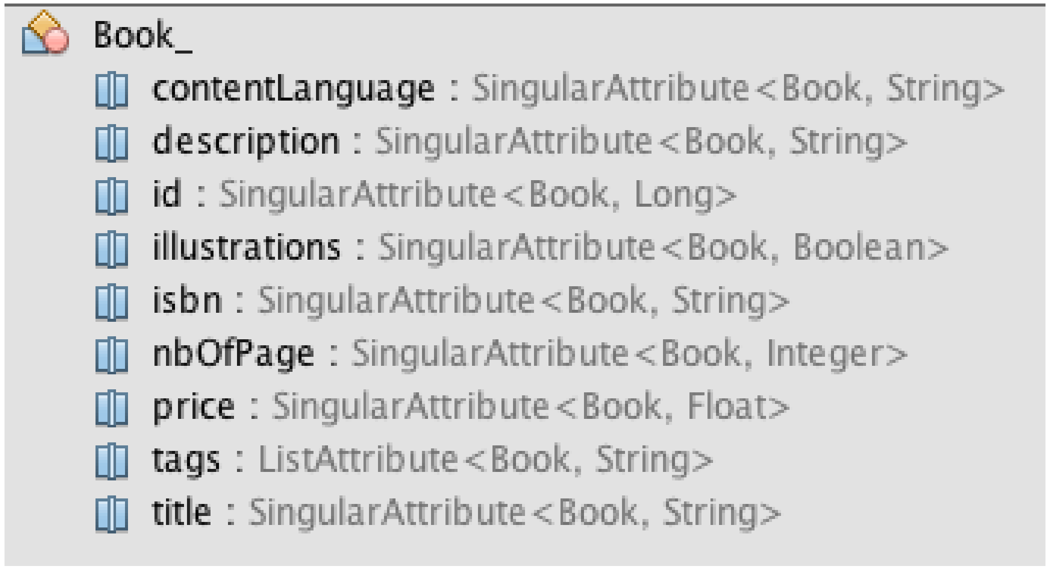
Criteria API

Type-safe

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Book> query =
                     cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);

query.select(book)
     .where(cb.isEmpty(book.get(Book_.description)));
```



```
Book_
  contentLanguage : SingularAttribute<Book, String>
  description : SingularAttribute<Book, String>
  id : SingularAttribute<Book, Long>
  illustrations : SingularAttribute<Book, Boolean>
  isbn : SingularAttribute<Book, String>
  nbOfPage : SingularAttribute<Book, Integer>
  price : SingularAttribute<Book, Float>
  tags : ListAttribute<Book, String>
  title : SingularAttribute<Book, String>
```

Statically generated
JPA 2.0 MetaModel

Criteria API

Builder pattern

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Book> query =
                     cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);

query.select(book)
     .where(cb.isEmpty(book.get(Book_.description)))
     .orderBy(...)
     .distinct(true)
     .having(...)
     .groupBy(...);

                           TypedQuery<Book>
List<Book> books =
             em.createQuery(query).getResultList();
```

Dependency Injection

- Context & Dependency Injection (CDI)
  - JSR 299 with JSR-330 (`@Inject`)
  - Context management (conversation), events, alternatives, stereotypes, decorators & more
- Beans discovered at startup
- Injection metamodel (`BeanManager` API)
- `@Resource` still around

Qualified injection

qualifier (user-defined label)
i.e. « which one? »

```
@Inject @Premium Customer cust;
```

injection point

type

Qualifier Annotation

```
@Target({TYPE,METHOD,PARAMETER,FIELD})
@Retention(RUNTIME)
@Documented
@Qualifier
public @interface Premium {…}


@Premium  // my own qualifier
public class SpecialCustomer
                implements Customer {
    public void buy() {…}
}
```

Qualified injection

qualifier (user-defined label)
i.e. « which one? »

```
@Inject @Premium Customer cust;
```

injection point

type

Dependency Injection Sample

```
public class CheckoutHandler {

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

}
```

How hard should it be to test EJBs?

```
EJBContainer c = EJBContainer.createEJBContainer();

Context ic = c.getContext();

SimpleEjb ejb = (SimpleEjb)
        ic.lookup("java:global/sample/SimpleEjb");

ejb.sayHello();
```

How hard should it be to test EJBs?

New in EJB 3.1

```
EJBContainer c = EJBContainer.createEJBContainer();

Context ic = c.getContext();

SimpleEjb ejb = (SimpleEjb)
        ic.lookup("java:global/sample/SimpleEjb");

ejb.sayHello();
```

Portable JNDI name

How hard should it be to test EJBs?

```
@Test public void test() {
   EJBContainer c = EJBContainer.createEJBContainer();
   Context ic = c.getContext();
   SimpleEjb ejb = (SimpleEjb)
           ic.lookup("java:global/sample/SimpleEjb");
   ejb.sayHello();
}
```

Demo

GlassFish Embedded

```
org.glassfish.api.embedded.Server  server;
Server.Builder builder = new Server.Builder();
server = builder.build();
```

GlassFish Embedded

```
org.glassfish.api.embedded.Server  server;
Server.Builder builder = new Server.Builder();
server = builder.build();

ContainerBuilder b =
     server.createConfig(ContainerBuilder.Type.web);
server.addContainer(b);
```

GlassFish Embedded

```
org.glassfish.api.embedded.Server  server;
Server.Builder builder = new Server.Builder();
server = builder.build();

ContainerBuilder b =
     server.createConfig(ContainerBuilder.Type.web);
server.addContainer(b);

File archive = new File("hello.war");
server.getDeployer().deploy(archive);
```

Same bits, different entry point
All in one JAR available

GlassFish Embedded

```
@BeforeClass public static void initContainer() {

   org.glassfish.api.embedded.Server  server;
   Server.Builder builder = new Server.Builder();
   server = builder.build();

   ContainerBuilder b =
        server.createConfig(ContainerBuilder.Type.web);
   server.addContainer(b);

   File archive = new File("hello.war");
   server.getDeployer().deploy(archive);

}

@Test public static void pingApplication() {

   ...

}
```

GlassFish Embedded

```
public static void main(String[] args) {

   org.glassfish.api.embedded.Server server;
   Server.Builder builder = new Server.Builder();
   server = builder.build();

   ContainerBuilder b =
        server.createConfig(ContainerBuilder.Type.web);
   server.addContainer(b);

   File archive = new File("realApplication.war");
   server.getDeployer().deploy(archive);

}
```

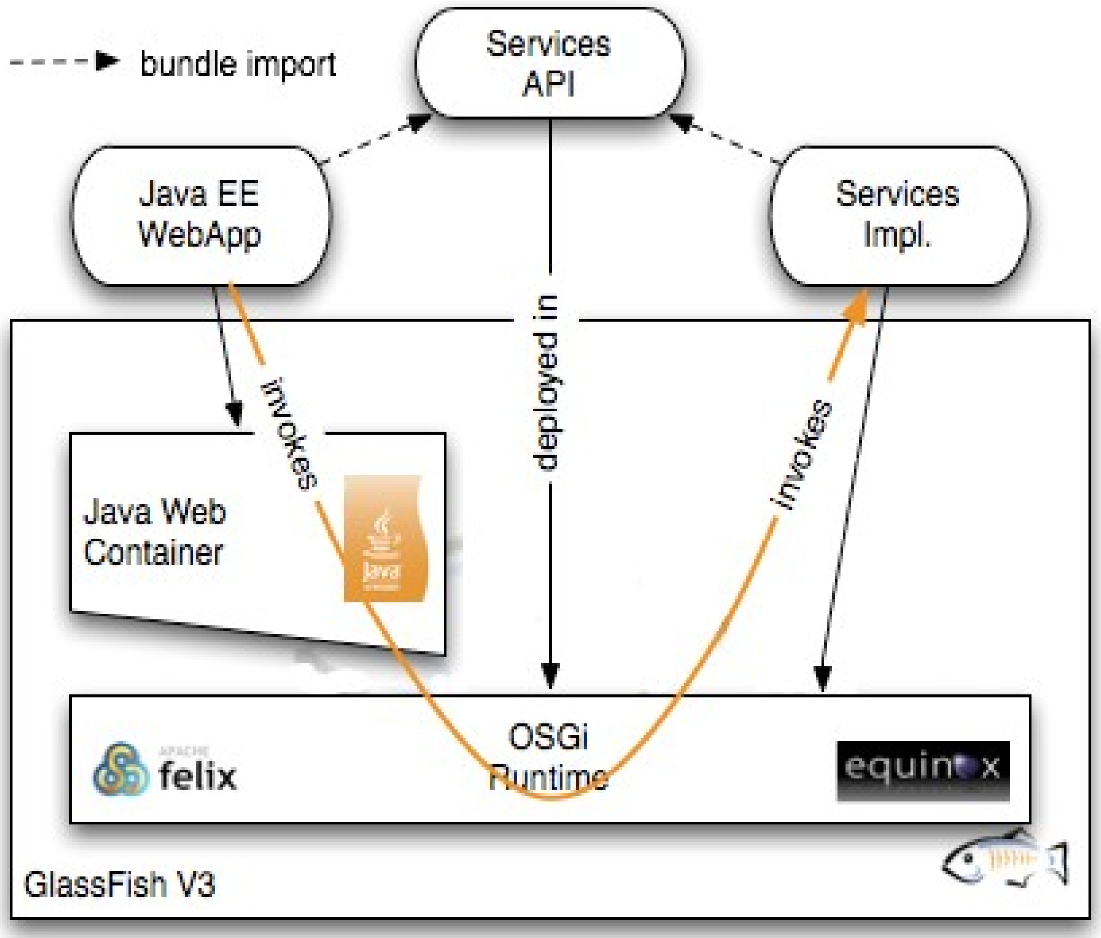
Ship app server inside the application

What's the deal with OSGi?

- GlassFish runs on top of OSGi (Felix by default)
  - Also runs unmodified on Equinox (and Knopflerfish)
  - GlassFish ships as 200+ bundles
  - Can run without OSGi (Static mode)
  - Can use OSGi management tools (CLI or Web)
  - Can be installed on top of existing OSGi runtime
- Any OSGi bundle will run in GlassFish v3
  - Drop it in `glassfish/modules{/autostart}`
  - Can also `asadmin deploy` it using --`type osgi`
  - GlassFish OSGi admin console

Extending GlassFish v3

OSGi-style – an example, a demo and a picture



- OSGi declarative service
- `Service-Component` entry in the JAR Manifest
- Invoke the service from a servlet using standard `@Resource` injection
- Never use a GlassFish API !
- No need to chose between OSGi and Java EE

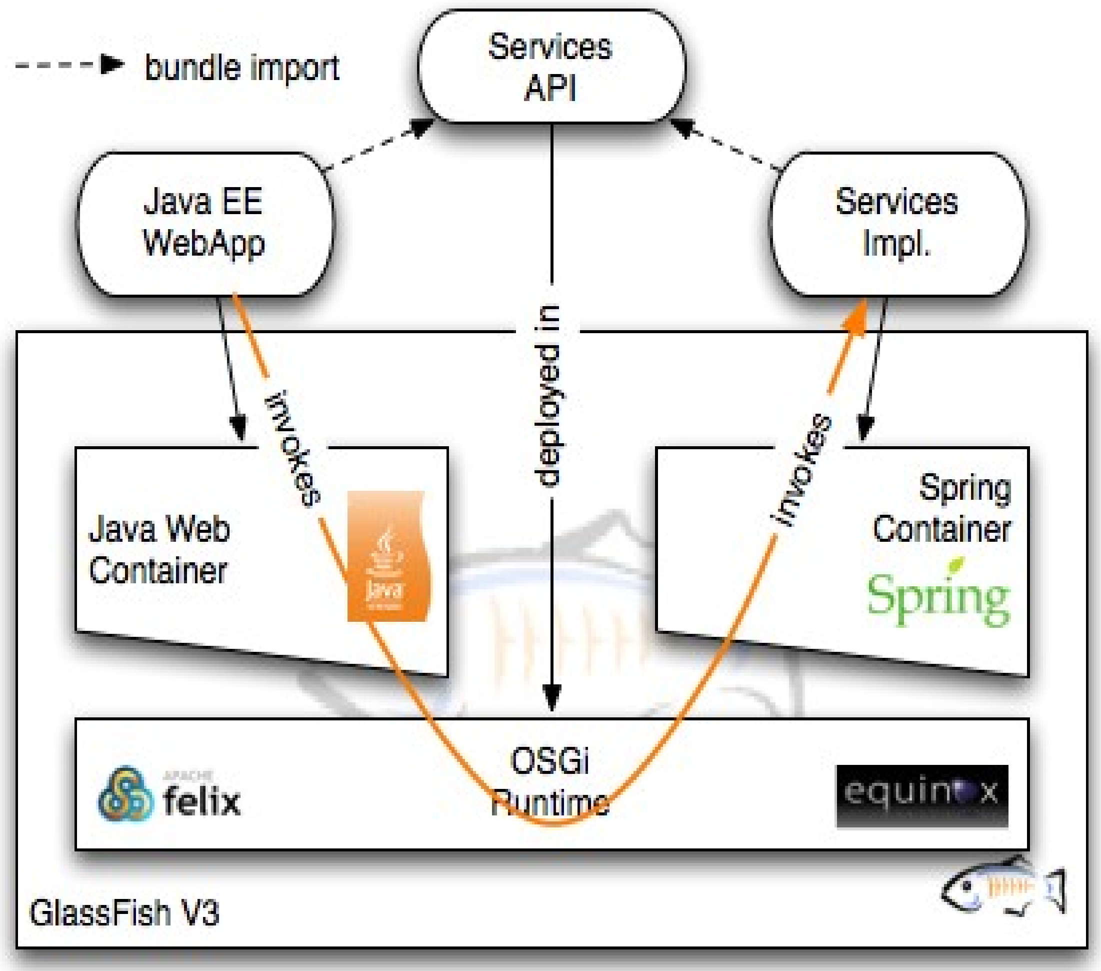
Step by step: http://blogs.sun.com/dochez/entry/glassfish_v3_extensions_part_4

Demo

Extending GlassFish v3
OSGi-style

Extending GlassFish v3

SpringDM – another example, demo and picture



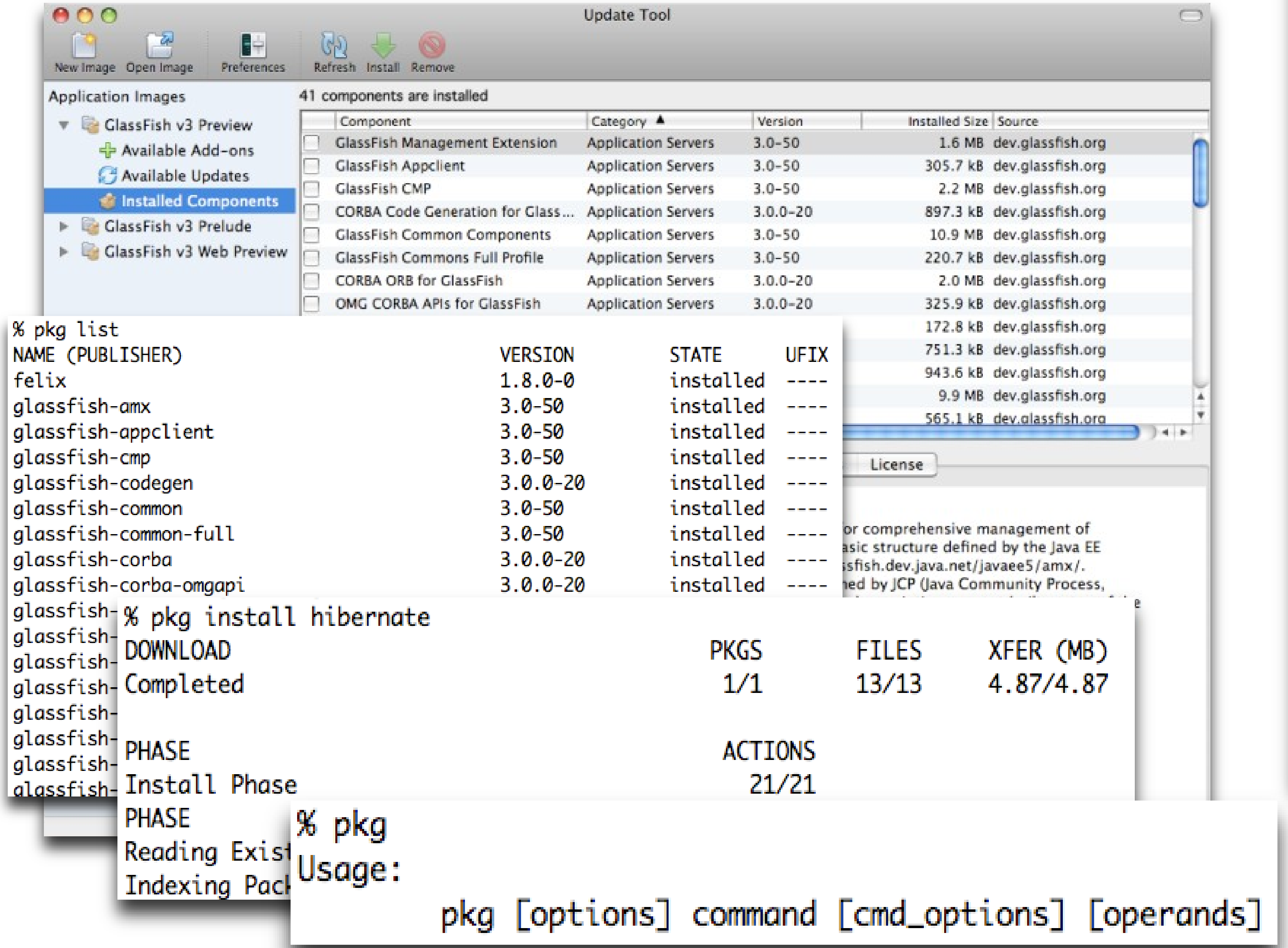
- Extend GlassFish with an unmodified Spring dm container
- Simple Spring bean implementing the service
- Invoke the service from a servlet using standard `@Resource` injection
- Still no use of a GlassFish API
- Single runtime for both Spring and full Java EE

Step by step: http://blogs.sun.com/dochez/entry/glassfish_v3_extensions_part_4

OSGi + Java EE = Hybrid Apps

- GlassFish as the modular runtime
  - Assembled spontaneously
  - Admin tools (Web & CLI)
- Implementation of Java EE related OSGi services & standards
  - OSGi RFC's
- Support for Java EE 6 platform
  - e.g. JPA, EJB, JDBC, JTA, ... as OSGi services
- Web Application Bundle (WAB)
  - WAR + OSGi metadata + Web-ContextPath header

Update Center



41 components are installed

Component	Category	Version	Installed Size	Source
GlassFish Management Extension	Application Servers	3.0-50	1.6 MB	dev.glassfish.org
GlassFish Appclient	Application Servers	3.0-50	305.7 kB	dev.glassfish.org
GlassFish CMP	Application Servers	3.0-50	2.2 MB	dev.glassfish.org
CORBA Code Generation for Glass…	Application Servers	3.0.0-20	897.3 kB	dev.glassfish.org
GlassFish Common Components	Application Servers	3.0-50	10.9 MB	dev.glassfish.org
GlassFish Commons Full Profile	Application Servers	3.0-50	220.7 kB	dev.glassfish.org
CORBA ORB for GlassFish	Application Servers	3.0.0-20	2.0 MB	dev.glassfish.org
OMG CORBA APIs for GlassFish	Application Servers	3.0.0-20	325.9 kB	dev.glassfish.org
			172.8 kB	dev.glassfish.org
			751.3 kB	dev.glassfish.org
			943.6 kB	dev.glassfish.org
			9.9 MB	dev.glassfish.org
			565.1 kB	dev.glassfish.org

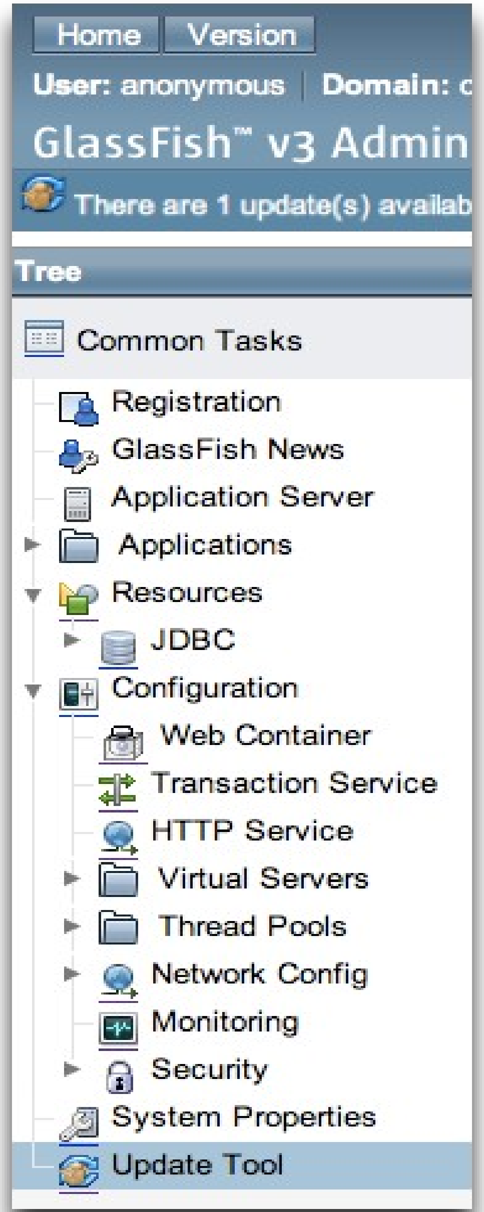
or comprehensive management of
asic structure defined by the Java EE
sfish.dev.java.net/javaee5/amx/.
ned by JCP (Java Community Process,

```
% pkg list
NAME (PUBLISHER)                     VERSION      STATE      UFIX
felix                                1.8.0-0      installed  ----
glassfish-amx                        3.0-50       installed  ----
glassfish-appclient                  3.0-50       installed  ----
glassfish-cmp                        3.0-50       installed  ----
glassfish-codegen                    3.0.0-20     installed  ----
glassfish-common                     3.0-50       installed  ----
glassfish-common-full                3.0-50       installed  ----
glassfish-corba                      3.0.0-20     installed  ----
glassfish-corba-omgapi               3.0.0-20     installed  ----
glassfish-
glassfish-
glassfish-
glassfish-
glassfish-
glassfish-
glassfish-
glassfish-
```

```
% pkg install hibernate
DOWNLOAD                          PKGS      FILES     XFER (MB)
Completed                          1/1      13/13     4.87/4.87

PHASE                                 ACTIONS
Install Phase                           21/21
PHASE
Reading Exist
Indexing Pack
```

```
% pkg
Usage:
        pkg [options] command [cmd_options] [operands]
```



Demo

GlassFish à la Carte

GlassFish à la carte

- Unzip 5-MB bootstrap
  - Install core IPS packages

GlassFish à la carte

- Unzip 5-MB bootstrap
  - Install core IPS packages
- Define the repository to use

GlassFish à la carte

- Unzip 5-MB bootstrap
  - Install core IPS packages
- Define the repository to use
- Install individual packages
  - Start with core `glassfish-nucleus` package
  - Drags dependencies of course

GlassFish à la carte

- Unzip 5-MB bootstrap
  - Install core IPS packages
- Define the repository to use
- Install individual packages
  - Start with core `glassfish-nucleus` package
  - Drags dependencies of course
- Install umbrella package (a distro really)
  - Enough to run a JAX-RS/EJB31 demo

GlassFish à la carte

- Unzip 5-MB bootstrap
  - Install core IPS packages
- Define the repository to use
- Install individual packages
  - Start with core `glassfish-nucleus` package
  - Drags dependencies of course
- Install umbrella package (a distro really)
  - Enough to run a JAX-RS/EJB31 demo
- Create GlassFish server instance
- Deploy application

GlassFish à la carte

- Unzip 5-MB bootstrap
  - Install core IPS packages
- Define the repository to use
- Install individual packages
  - Start with core `glassfish-nucleus` package
  - Drags dependencies of course
- Install umbrella package (a distro really)
  - Enough to run a JAX-RS/EJB31 demo
- Create GlassFish server instance
- Deploy application
- Run!

Monitoring and Management
Beyond web console and `asadmin`

- Dynamic and non-intrusive monitoring of events from any GlassFish runtime classes
  - BTrace integration new!
    - Portable, dynamic and safe tracing tool for Java
    - Btrace annotations and API to write scripts
  - Probe Providers defined in java or XML new!
    - Default providers & roll out your own
  - RESTful interface new!
  - DTrace for end-to-end new!
- Still exposed via JMX
  - `jconsole` and `visualvm` as natural clients

RESTful admin

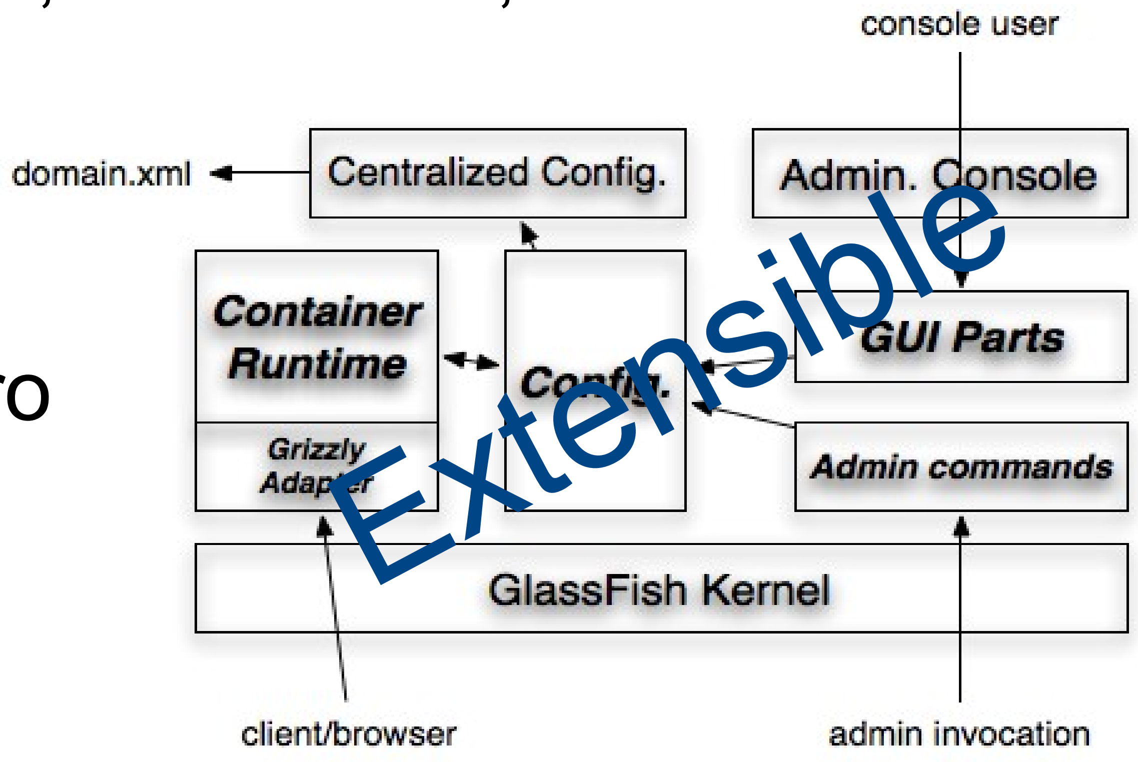
- JAX-RS/Jersey + Grizzly to provide REST interfaces to :
  - Configure runtime (via GET, POST, DELETE)
  - Invoke commands (restart, stop, deploy, etc..)
  - Monitoring (GET only)
- Available from :
  - `http://localhost:4848/management/domain`
  - `http://localhost:4848/monitoring/domain`
- Use REST clients as Admin GUI substitute
  - Use you favorite glue/scripting language or tool
- Data offered as either XML, HTML or JSON
- Extensible

Demo

RESTful admin

A lot more ...

- Dynamic languages
  - Rails, Grails, Django, Scala/Lift...
- Async Web
  - Comet, Atmosphere, WebSockets, ...
- Full support for :
  - mod_jk
  - WebDAV, CGI, SSI
- Web Services Metro
  - .Net 3.5 interop
- OpenMQ



GlassFish – Practical

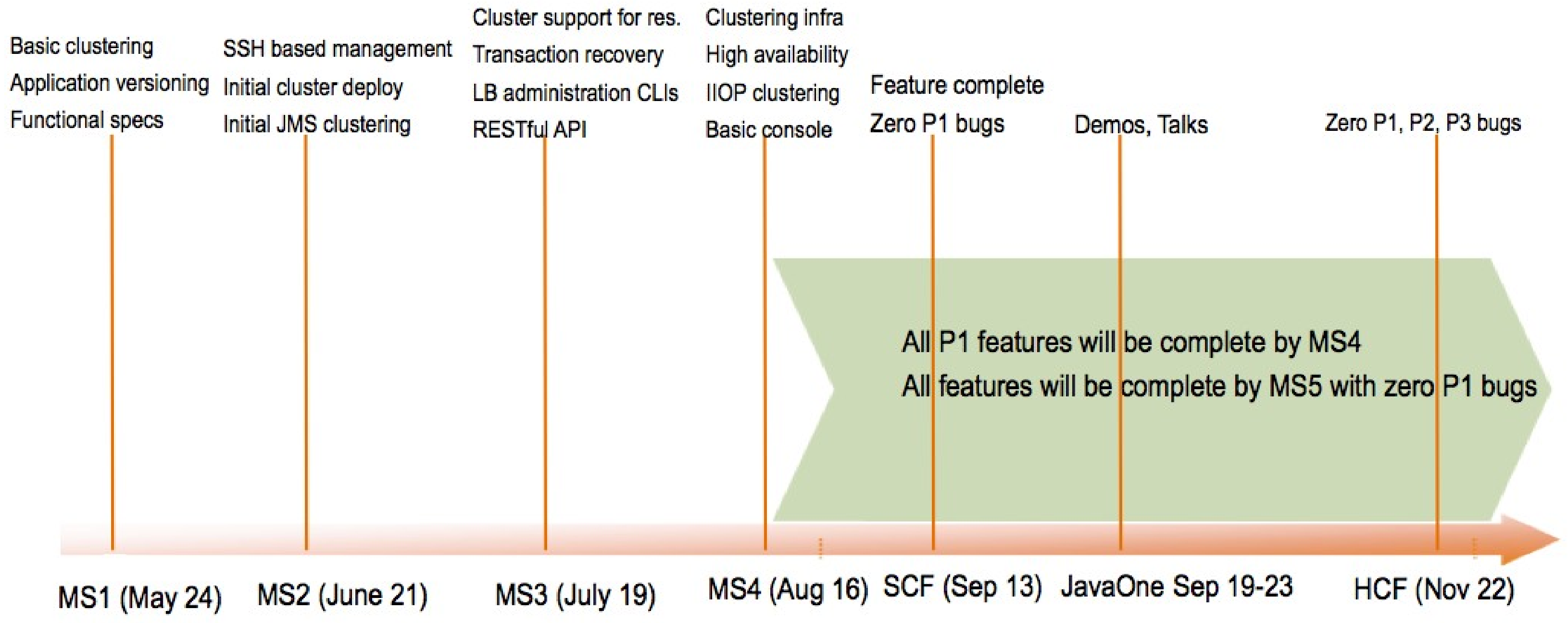
- Get it from http://glassfish.org
  - GlassFish 3.0.1 available (use UC to update from v3)
  - Also from http://www.oracle.com/goto/glassfish (AddOns software integrated for eval.)

- Choice !
  - Eclipse or NetBeans (or vi...)
  - Felix or Equinox

- Graphical Installer, Zip version
- Download size starting at 33MB
  - Can move from Web profile to full platform

A glance at GlassFish OSE 3.1

- High-level goal
  - Combine the benefits from 2.1.1 and 3.0
    - Clustering, replication and centralized admin from 2.1.1
    - OSGi modularity and Java EE 6 from 3.x
- Milestone-driven development



Some GlassFish 3.1 highlights

- Basic clustering (M1)
- App. versioning (M1)
- RESTful API (M1)
- Stabilize Embedded
- Shoal over grizzly
- Metro RM & SecConv session failover
- Retain SFSB/EJB Timer across redeploys
- Application-scoped resources
- WebSockets (via Grizzly)
- More Enterprise OSGi
- Updated IDE plugins
- Improved CDI integration
- Technology refresh: JSF, CDI, Grizzly, OSGi, JPA, Jersey, Bean Validation, Metro, UC, etc.
- WS-I compliance: BP 1.2/2.0, BSP, 1.1, RSP 1.0
- Much more...

Questions

alexis.mp@sun.com
http://blogs.sun.com/alexismp
twitter: alexismp