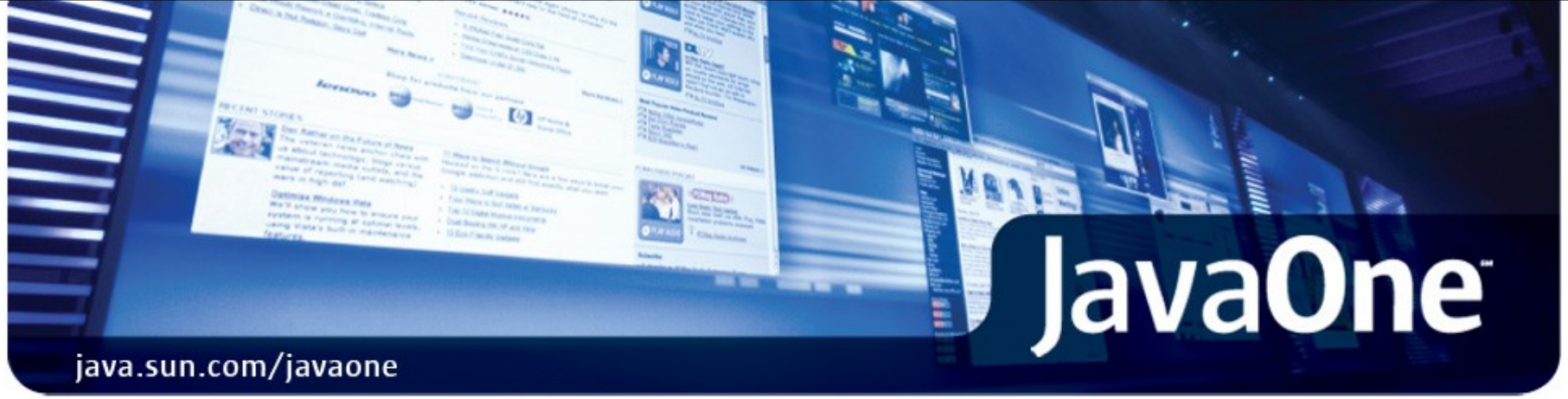


Java™ Servlet 3.0 API: What's new and exciting

Rajiv Mordani
Senior Staff Engineer, Sun Microsystems

TS-5415

Java

Sun microsystems

Learn about the new features in the Java™ Servlet 3.0 API

GOAL

Agenda

- Overview
- Pluggability
- Ease of Development
- Async servlet support
- Security
- Others
- Status
- Summary

Overview

- Java™ Servlet 3.0 API – JSR 315
- Has about 20 members in the expert group with a good mix of representation from the major Java™ EE vendors, web container vendors and individual web framework authors
- Main areas of improvements and additions are -
  - Pluggability
  - Ease of development
  - Async servlet support
  - Security enhancements
- Note: Specification in early draft and things can change
  - The good news is that the community still has time to provide feedback

Agenda

- **Overview**
- **Pluggability**
- **Ease of Development**
- Async servlet support
- Security
- Others
- Status
- Summary

Pluggability

- Make it possible to use framework and libraries with no additional configuration
- Modularizing web.xml to allow frameworks / libraries to have their own entities defined and self-contained within the framework
- Adding APIs to `ServletContext` to allow addition of `Servlets`, `Filters` and `Listeners` to a web application at application startup time.
- Use of annotations to declare all the components within a web application

Pluggability - Modularization of web.xml

- Current users of framework need to edit their application's web.xml to
  - Define a Java™ Servlet provided by the framework (typically a controller Java™ Servlet)
  - Define Filters that the framework needs in order to be used within a web application (logging for example or Filters to implement security constraints)
  - Define listeners so that appropriate action can be taken at different points in the application / component's life cycle.
- Monolithic web.xml can become complex to maintain as the dependencies of the application increases
- Each framework needs to document for the developer what all must be declared in the web.xml

Pluggability – Modularization of web.xml

- Java™ Servlet 3.0 specification introduces the concept of modular web.xml
- Each framework can define it's own web.xml and include it in the jar file's META-INF directory
- The developer needs to include the framework jar in the application
- At deployment the container is responsible for discovering the web.xml fragments and processing them.
- Introduce new element – web-fragment that can define servlets, filters and listeners as child elements.

Pluggability - example new elements in web.xml

```
<web-fragment>
  <servlet>
    <servlet-name>welcome</servlet-name>
    <servlet-class>
      WelcomeServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>welcome</servlet-name>
    <url-pattern>/Welcome</url-pattern>
  </servlet-mapping>
...
</web-fragment>
```

Pluggability – Configuration methods in ServletContext

- In addition to web.xml modularization methods added to the ServletContext to declare and configure servlets and filters.
- Can only be called at context initialization time.
- Allows to
  - Declare a new Servlet
  - Define a url mapping for the Servlet declared
  - Declare a Filter
  - Define a url mapping for the Filter
- Enables applications to load Servlets and filters at runtime that are needed

Pluggability – APIs in ServletContext example

```
@ServletContextListener
public class MyListener {
    public void contextInitialized
                (ServletContextEvent sce) {
        ServletContext sc = sce.getServletContext();
        sc.addServlet("myServlet",
                      "Sample servlet",
                      "foo.bar.MyServlet",
                       null, -1);
        sc.addServletMapping("myServlet",
                             new String[]
                             {"/urlpattern/*"});
    }
}
```

Pluggability – APIs in ServletContext example

```
@ServletContextListener
public class MyListener {
    public void contextInitialized
                (ServletContextEvent sce) {
        ServletContext sc = sce.getServletContext();
        sc.addFilter("myFilter",
                     "Sample Filter",
                     "foo.bar.MyFilter",
                      null);
        sc.addFilterMapping("myFilter",
                            new String[]
                            {"/urlpattern/*"},
                           "myServlet",
                           DispatcherType.REQUEST,
                           false);
    }
}
```

Agenda

- **Overview**
- **Pluggability**
- **Ease of Development**
- Async servlet support
- Security
- Others
- Status
- Summary

Ease of Development

- Focus on ease of development in Java™ Servlet 3.0 API
- Enhance Java™ Servlet APIs to use newer language features
- Annotations for declarative style of programming
- Generics for better compile time error checking and type safety
- web.xml optional (was already optional for Java™ EE 5)
  - Restricted to JSPs and static resources only
- Better defaults / convention over configuration

Ease of Development – defining a servlet

- Define a servlet using a `@Servlet` annotation
- Must contain a `url-mapping`
- All other fields optional with reasonable defaults –
  - example the "name" of the servlet is the fully qualified class name if none is specified.
  - Can define the appropriate http methods using the annotations `@GET`, `@PUT`, `@POST`, `@DELETE`, `@HEAD`
  - `@HttpMethod` meta-annotation allows extensions
- Can use the web.xml to override annotation values

Servlet example – 2.5 style

```
public class SimpleSample
extends HttpServlet {
    public void doGet
    (HttpServletRequest req,
     HttpServletResponse res)
    {

    }
}
```

web.xml

```
<web-app>
  <servlet>
    <servlet-name>
       MyServlet
    </servlet-name>
    <servlet-class>
     samples.SimpleSample
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>
      MyServlet
    </servlet-name>
    <url-pattern>
      /MyApp
    </url-pattern>
  </servlet-mapping>
...
</web-app>
```

Ease of development – Defining a servlet

```
@Servlet(urlMapping={"/foo"})
public class SimpleSample {

}
```

Code Sample

```
@Servlet(urlMapping={"/foo", "/bar"},
         name="MyServlet")
public class SampleUsingAnnotationAttributes {
    @GET
    public void handleGet(HttpServletRequest req,
                          HttpServletResponse res)
    {

    }

}
```

Ease of development – Defining a Filter

- Define a Filter using a `@ServletFilter` annotation
- Must contain a `@FilterMapping` annotation
- All other fields optional with reasonable defaults

Code Sample

```
package samples;
import javax.servlet.http.annotation.*;

@ServletFilter
@FilterMapping(urlPattern="/foo")
public class SampleFilter {

    public void doFilter(HttpServletRequest req,
                         HttpServletResponse res)
    {

    }

}
```

Ease of development – Defining a ServletContextListener

- Define a context listener using a `@ServletContextListener` annotation

Ease of Developmnt – ServletContext example

```
@ServletContextListener
public class MyListener {
    public void contextInitialized
                (ServletContextEvent sce) {
        ServletContext sc = sce.getServletContext();
        sc.addServlet("myServlet",
                      "Sample servlet",
                      "foo.bar.MyServlet",
                       null, -1);
        sc.addServletMapping("myServlet",
                             new String[]
                             {"/urlpattern/*"});
    }
}
```

Agenda

- Overview
- Pluggability
- Ease of Development
- Async servlet support
- Security
- Others
- Status
- Summary

Async servlet – Use cases

- Comet style of application
- Async Web proxy
- Async Web services

Async servlet support – popular use case Comet

- Primer
  - Rely on a persistent HTTP connection between server and client
  - Two strategies
    - Streaming - browser opens a single persistent connection to the server for all Comet events each time the server sends a new event, the browser interprets it.
    - Long polling - a new request for each event (or set of events)
  - Standardization efforts as part of the Bayeux protocol
- Implementation specific APIs available today in the various Java™ Servlet containers.
- APIs added to Java™ Servlet 3.0 specification to enable Comet style programming
- Request can be suspended and resumed

Async servlet support - Suspending a request

- Request can be suspended by the application
- Allows the container to not block on a request that needs access to a resource – for example access to a DataSource or wait for a response from a call to a WebService.
- When resumed the Request is re-dispatched through the filters for processing.
  - Results in an additional thread for handling the new request
- The resume method on the request resumes processing
  - Can be used to push timely events in multi-user applications
- The complete method to indicate the completion of request processing
- Can query if a request is suspended, resumed or has timed out.

Async servlet support – methods added to ServletRequest

- Methods added to ServletRequest for suspending, resuming and querying for status -
  - `void suspend(long timeOutMs);`
  - `void resume();`
  - `void complete();`
  - `boolean isSuspended();`
  - `boolean isResumed();`
  - `boolean isTimeout();`

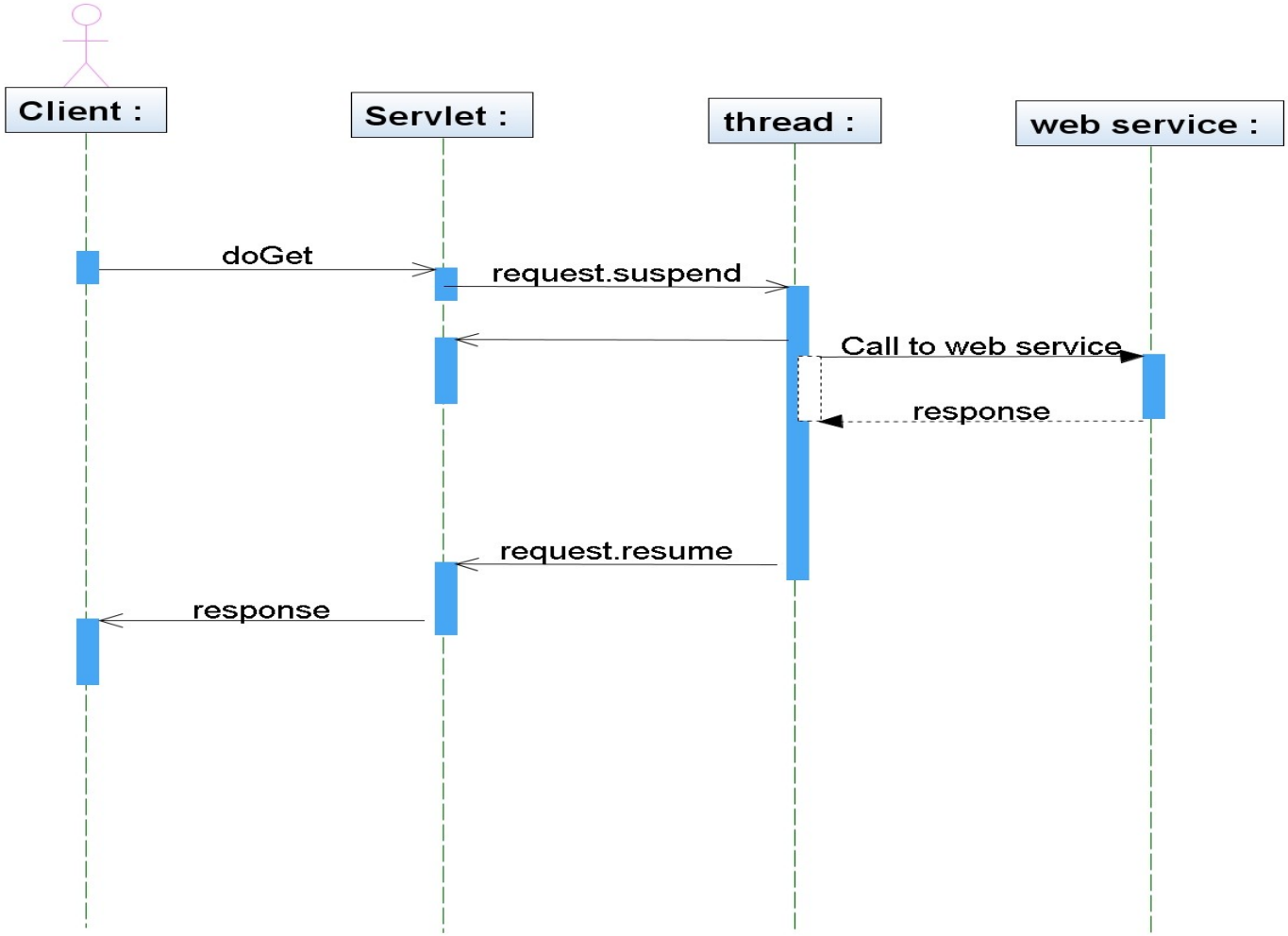
Async servlet support – Events on RequestListener

- Corresponding events fired for changes to request processing.
- Notification for suspend, resume and complete available for developers via the ServletRequestListener
- Methods added to **`ServletRequestListener`**
  - `void requestSuspended(ServletRequestEvent rre);`
  - `void requestResumed(ServletRequestEvent rre);`
  - `void requestCompleted(ServletRequestEvent rre);`

Async servlet support – methods added to Response

- Methods added to ServletResponse for disabling, enabling and querying for status -
  - `void disable();`
  - `void isDisabled();`
  - `void enable();`

Async servlet to web services call



Agenda

- Overview
- Pluggability
- Ease of Development
- Async servlet support
- Security
- Others
- Status
- Summary

Security

- Ability to login and logout programmatically
- Methods added to **`ServletRequest`** to force a login and ability to logout
- Still being discussed in the EG.
- Proposal to add login and logout method to
  - **`HttpServletRequest`**
  - **`HttpServletRequestWrapper`**
  - **`HttpSession (logout only)`**

Security – login and logout

- Login method intended to allow an application or framework to force a container mediated authentication from within an unconstrained request context
- Login requires access to the `HttpResponse` object to set the www-authenticate header.
  - Available through new methods added to the request to give access to the corresponding response object
- logout methods are provided to allow an application to reset the authentication state of a request without requiring that authentication be bound to an `HttpSession`
- Still in discussion in the Expert Group and not closed upon.

Agenda

- Overview
- Pluggability
- Ease of Development
- Async servlet support
- Security
- Others
- Status
- Summary

Others – HttpOnly Cookie support

- Added support for HttpOnlyCookies
- Prevents access to the cookie from client side scripting code
- Prevents cross-site scripting attacks.
- Method added to `Cookie` to set and query if it is an HttpOnly cookie

Others – Session tracking cookie configuration

- Ability to set the session tracking cookie configuration for the corresponding `ServletContext`
- Supports multiple Session tracking mode – COOKIE, URL, SSL

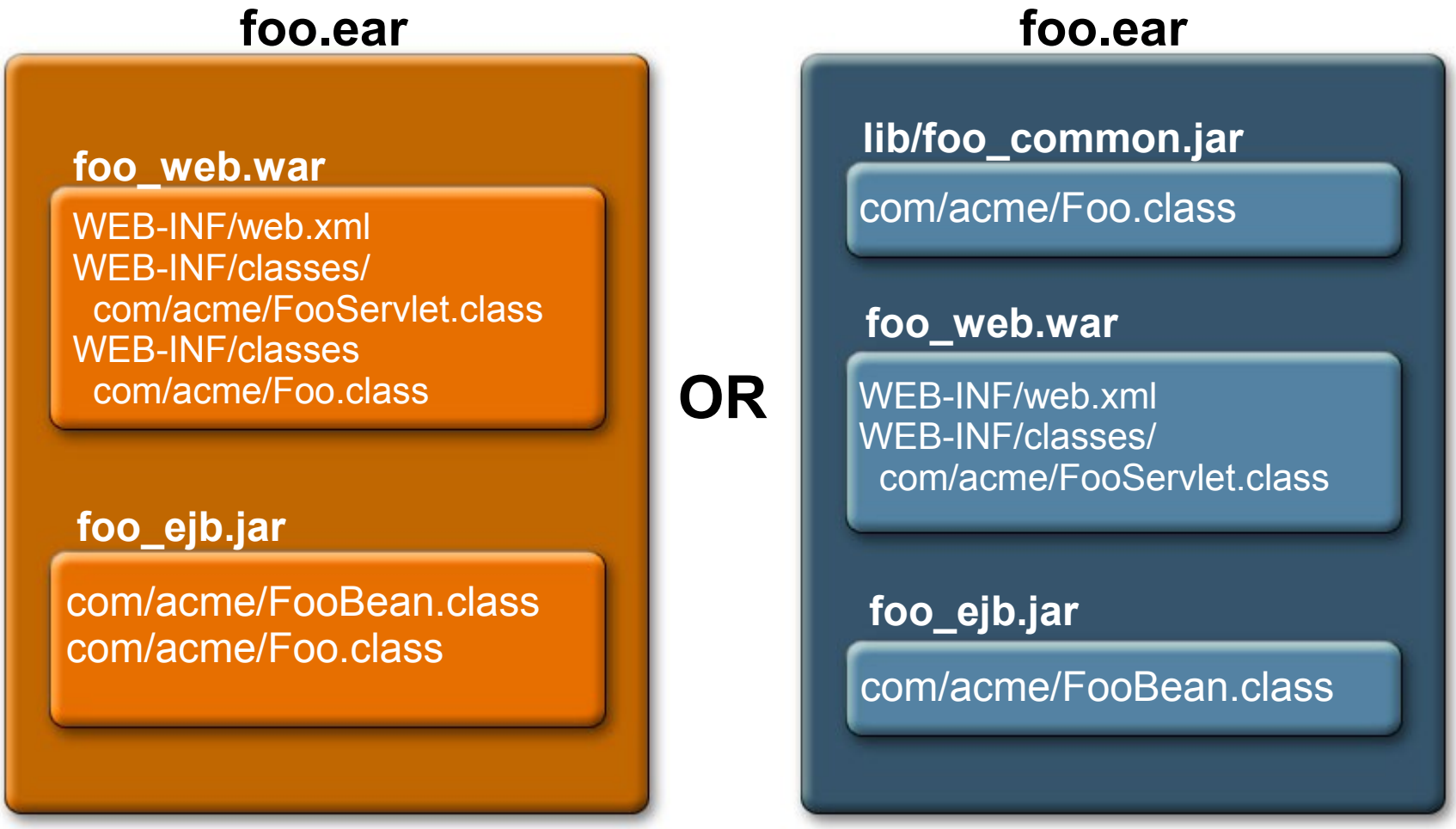
Pending discussion in the expert group

- Miscellaneous items to be done for Java™ Servlet 3.0 API
  - File upload
  - Container wide init-params
  - Clarifications from previous releases
  - Enablement of JAX-RS / JSF 2.0 (if any changes needed)

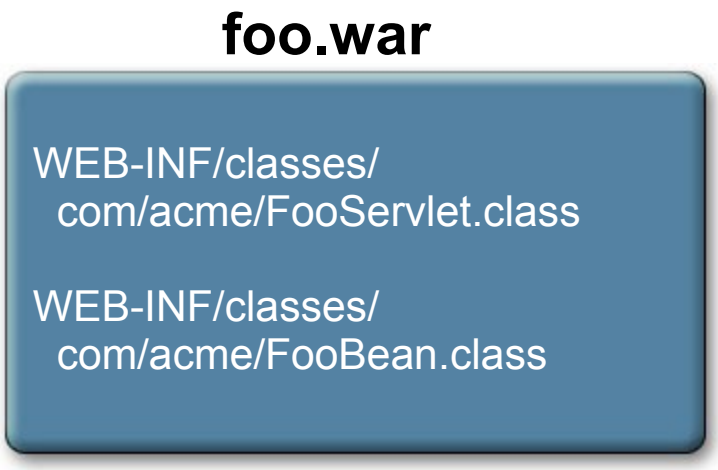
Java EE profiles

- Java EE 6 specification introducing notion of profiles
- Targeting a web profile for Java EE 6
- Web profile to be based on Servlets and JSPs
- Still being discussed in the Java EE 6 expert group
- Roberto solicited feedback from the community
- Varying opinions of what should be in the profile
- Still need to close on in the Java EE 6 expert group

Web/EJB Technology Application in Java™ EE Platform 5



Web/EJB Technology Application in Java™ EE Platform 6



EJB in a WAR file

- Goal is to remove an artificial packaging restriction
  - NOT to create a new flavor of EJB component
- EJB component behavior is independent of packaging
- Full EJB container functionality available

Agenda

- Overview
- Pluggability
- Ease of Development
- Comet
- Security
- Status
- Summary

Status

- Currently in Early Draft Review
- Public Review in summer of this year
- Proposed final draft and final release aligned with Java™ EE 6
- Early access to bits of implementation to be available via Project GlassFish

Agenda

- Overview
- Pluggability
- Ease of Development
- Comet
- Security
- Status
- Summary

Summary

Lot of exciting things happening in the Java™ Servlet 3.0 API

- Pluggability for frameworks
- Ease of Development for developers
- Comet support to enable modern web 2.0 style applications
- Security enhancements to enable programmatic login / logout
- Miscellaneous improvements for better developer experience

- Make the life of framework developers and users much easier
- Implementation being done in open source as part of GlassFish project

GlassFish Community

Open Source, Enterprise Ready & Extendable

- **GlassFish V3 Tech Preview 2 Available now!**
  - Modular OSGi architecture – easy to deploy, Develop and Extend
- **GlassFish V2 – Production Ready**
  - Fastest open source app server with Clustering, High Availability, Load Balancing
  - Support for jMaki, Comet, Ajax, Ruby and Groovy
- **GlassFish ESB**
  - Core SOA functions now embedded in GlassFish
- **GlassFish Communications App Server**
  - SIP servlet technology for converged services

Always free to download, deploy and distribute

- **GlassFish Partner Initiative**
  - Expanding the ecosystem for partners.
- **Enterprise and Mission Critical Support**
  sun.com/software/products/appsrvr
- **GlassFish Unlimited Pricing**
  - Fixed price, unlimited deployments
  - Combine w/ MySQL Unlimited
- **Tools Integration**
  - NetBeans and Eclipse

glassfish.org

For More Information

- Official JSR Page
  - http://jcp.org/en/jsr/detail?id=315
- JAX-RS – TS 5425
- JSF 2.0 – TS 5979
- Mailing list for webtier related issues - webtier@glassfish.dev.java.net
- Rajiv's blog
  - http://weblogs.java.net/blog/mode

THANK YOU

Rajiv Mordani
Senior Staff Engineer, Sun Microsystems

TS-5415