Overall Presentation Goal

Learn about the new features in the Java Platform, Enterprise Edition 6
Speaker’s qualifications

- Roberto Chinnici is a Senior Staff Engineer at Sun Microsystems
- Roberto Chinnici is the specification lead for the Java EE 6 platform (JSR-316)
- Roberto Chinnici speaks frequently on the Java EE platform and scripting languages
Overall Goal for Java EE 6

Make the platform:

- Easier to use
- More flexible, adaptable
- Easier to learn
- Easier to evolve going forward
Timeline

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

J2EE 1.3
- CMP, Connector Architecture

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

Web Services

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

Java EE 6
- EJB Lite
- Restful WS
- Web Beans
- Extensibility

Profiles

Java EE 6 Web Profile

Ease of Development

Robustness

Enterprise Java Platform

JPE Project
Major New Features

- Profiles
- Pruning
- Extensibility
- Ease of development
Profiles

- Profiles are targeted bundles of technologies
- (Simple) rules set by platform spec
- Profiles can be subsets, supersets or overlapping
- Decoupling of specs to allow more combinations
- First profile: the Web Profile
- Future profiles defined in the Java Community Process
Pruning

- Make some technologies optional
- Reduce the real and conceptual footprint
- Same rules as proposed by the Java SE
  - “pruned now, optional in the next release”
- Pruned technologies will be marked in the javadocs
- Current pruning list:
  - JAX-RPC, EJB Entity Beans, JAXR, JSR-88
Extensibility

- Embrace open source libraries and frameworks
- Zero-configuration, drag-and-drop for web frameworks
  - Servlets, servlet filters, context listeners for a framework get discovered and registered automatically
- Completely general solution using web fragments
- Scripting languages can use the same mechanism
Ease of Development

- Ongoing concern
- This time focus is the web tier
- Lots of opportunities in other areas, e.g. EJB

General principles:

- Annotation-based programming model
- Traditional API for advanced users
- Reduce or eliminate need for deployment descriptors
- Get technologies to work together well
Proposed Components (1)

Full JSRs

- **EJB 3.1 (JSR-318)** PUBLIC REVIEW DONE
- **JPA 2.0 (JSR-317)** IN PUBLIC REVIEW
- **Servlet 2.0 (JSR-315)** PUBLIC REVIEW SOON
- **JSF 2.0 (JSR-314)** IN PUBLIC REVIEW
- **JAX-RS 1.0 (JSR-311)** COMPLETED
- **Connectors 1.6 (JSR-322)** IN PUBLIC REVIEW
- **Web Beans 1.0 (JSR-299)** IN PUBLIC REVIEW
- **Bean Validation 1.0 (JSR-303)** PUBLIC REVIEW “SOON”

Not finalized yet. The Java EE 6 platform expert group has the final word on the contents.
Proposed Components (2)

Maintenance Releases

- JAXB 2.2
- JAX-WS 2.2
- JAX-RS 1.1
- EL 1.2
- JSP 1.2

Others

- Authentication SPI (JSR-196)
- Concurrency utilities for Java EE (JSR-236)

Not finalized yet. The Java EE 6 platform expert group has the final word on the contents.
Servlet 3.0 Highlights

- Annotation-based programming model
  - @WebServlet @ServletFilter etc.
- Modular web.xml descriptor:
  - WEB-INF/lib/mylibrary.jar → META-INF/web-fragment.xml
- Annotations and web fragments are merged
- Programmatic API for dynamic registration of servlets
Useful for Comet, chat rooms, long waits

Must declare `@WebServlet(asyncSupported=true)`

Then call:

```java
AsyncContext ctx = ServletRequest.startAsync(req, res);
```

AsyncContext can then either:

- `forward(String path)`
- `start(Runnable action)`

AsyncContext.start(Runnable) must be paired with:

- `complete()`
EJB 3.1 Highlights

- Singleton beans: @Singleton
- No interface view: one source file per bean
- Calendar timers: @Schedule(dayOfWeek="Mon,Wed")
- Non-persistent timers (tied to a JVM)
- Async business methods: @Asynchronous
  - Methods must return void or a Future<T>
- Global JNDI names for beans
  - java:global/(app)/(module)/(bean)#(interface)
Singletons break a prior assumption in EJB

Reflected in complexity when defaults are not enough

Added concurrency management annotations

@ConcurrentManagement

- Container (default)
- Bean : @Lock @AccessTimeout
EJB 3.1 Lite

- Session beans (stateless, stateful, singletons)
- Transactions
- Security
- Interceptors
- ejb-jar.xml
- Embeddable container API: EJBCocontainer
- OK to bootstrap on Java SE
- Beans looked up by global JNDI name
Web Beans 1.0

- Adds contextual services to the platform
- Unifies JSF managed beans and EJB components
- Dependency-injection framework:
  ```java
  @LoggedIn User user;
  ```
  - Binding types to select among alternatives
  - Scopes (session, singleton, conversation, …)
  - Deployment types to select families of beans (e.g. mocks)
- Programmatic API: Manager, Bean, Context, …
No generic type literals in the language

List<String>.class  WRONG

Workaround in Web Beans, JAX-RS, others:

TypeLiteral t = new TypeLiteral<List<String>>() {};

For annotations with elements too:

AnnotationLiteral a = new AnnotationLiteral<PayBy>() {
    public value() { return CHEQUE; }
}

Implemented using Class.getGenericSuperclass()
Removing Barriers

- EJB components can be placed in web applications
  - No need to wrap them in an EJB jar archive

WebBeans has a unified EL resolver

- Place @Named("foo") on a bean to give it a name
- Use "#{foo}" to refer to it from JSF

Unify @Resource-style injection with Web Beans

Apply javax.interceptor more widely
Web Profile

- Much discussion in the expert group and outside it
- Latest proposal: a fully functional mid-sized profile

Accepted:

- Servlet 3.0, JSP 2.1, JSR-45, EL 1.2, JSTL 1.2, JSF 2.0
- EJB Lite 3.1, JTA 1.1, JPA 2.0, JSR-250

Controversial:

- JAX-RS 1.1, Web Beans 1.0, JSR-303
Schedule

- Public reviews to be completed in January
- Proposed final draft at the end of January/February
- Final release at the end of May
- Implementation as GlassFish V3
  - Some modules in early access now
  - Other modules will become available in the near future
  - Mostly complete in early March
Summary

- Smaller, more agile platform: profiles, pruning, extensibility
- Ease of development still a major focus area
- Component specs ready for public review
- Open source implementation in GlassFish V3
Concluding statement

Learn about Java EE 6.
Send feedback on the component technologies.
Download GlassFish V3 prelude.
Participate.
Contribute.
Enjoy!
Thanks for your attention!

http://java.sun.com/javaee