



Java EE 6: Renewing the Platform

Roberto Chinnici
JSR-316 Specification Lead
Sun Microsystems, Inc.



Contents

- Goals
- Features
- Component JSRs
- Highlights
- Schedule

Goals for the Java EE 6 Platform

- Easier to use
- More flexible
- Easier to learn
- Easier to evolve going forward

Major New Features in Java EE 6

- Profiles
- Pruning
- Extensibility
- More ease of development

Profiles

- A profile is a targeted bundle of technologies
- Rules set by the Java EE Platform spec
- Profiles can be subsets, supersets, overlapping
- Define immediately a Web Profile
- JCP process for future profiles

Pruning

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

Extensibility

- Embrace open source libraries, frameworks
- Full pluggability in the web container
- No configuration needed for web frameworks
 - > Discover servlet, servlet filters, listeners for a framework
 - > Dynamically add servlets, filters to an application
- Used to support scripting languages

Proposed Components (1)

Full JSRs

- EJB 3.1 (JSR-318) **PFD**
- JPA 2.0 (JSR-317) **PFD**
- Servlet 3.0 (JSR-315) **PFD**
- JSF 2.0 (JSR-314) **COMPLETED**
- JAX-RS 1.0 (JSR-311) **COMPLETED**
- Connector Architecture 1.6 (JSR-322) **PFD**
- Bean Validation 1.0 (JSR-303) **PFD**
- JSR-299 (née Web Beans) **PFD (INCLUSION STILL NOT DECIDED)**

Proposed Components (2)

Maintenance Releases

- JAXB 2.2
- JAX-WS 2.2
- JSR-109 1.3
- JAX-RS 1.1
- JSF 2.1
- EL 1.2, JSP 1.2
- JSR-196 1.1, JACC 1.3
- Common Annotations 1.1 (JSR-250)

Web Profile

- A fully functional mid-sized profile for web apps
- Component technologies:
 - > 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
 - > Bean Validation 1.0
- Possibly:
 - > JSR-299 1.0

Servlet 3.0 Highlights (1)

- Annotations to define web components
 - > `@WebServlet` `@WebFilter` `@WebListener` ...
- Greatly reduced editing of `web.xml`
- Customizable session cookie configuration
 - > `ServletContext.getSessionCookieConfig` method
- Selectable session tracking modes
 - > `ServletContext.setSessionTrackingMode`
 - > `ServletContext.getEffectiveSessionTrackingModes`

Servlet 3.0 Highlights (2)

- Modular web.xml using *fragments*
 - > WEB-INF/lib/mylib.jar → META-INF/web-fragment.xml
- New initializers for extensions
 - > ServletContainerInitializer interface
- Programmatic API for dynamic registration of servlets and filters
 - > ServletContext.addServlet/addFilter methods
- Ability to tweak servlet/filter configuration at startup
 - > ServletRegistration interface

ServletContainerInitializer Sample

```
@HandlesTypes(WebService.class)
public class JAXWSInitializer implements
ServletContainerInitializer {
    public void onStartup(Set<Class<?>> classes, ServletContext ctx) {
        ctx.addServlet("JAXWSServlet", "com.sun.jaxws.JAXWSServlet");
    }
}
```

Servlet 3.0 Highlights (3)

- Resource sharing
 - > WEB-INF/lib/mylib.jar!/META-INF/resources
- Resources under / take precedence on bundled ones
- Annotations for security constraints
 - > @RolesAllowed @DenyAll @PermitAll
- Programmatic login, authentication, logout
 - > HttpServletRequest.login/authenticate/logout methods
- FileUpload
 - > @MultipartConfig
 - > ServletRequest.getPart method

Servlet 3.0 Async Processing (1)

- Use cases: Comet, chat rooms, long waits
- Opt-in system for servlets, filters
 - > `@WebServlet(asyncSupported=true)`
 - > `@WebFilter(asyncSupported=true)`
- Check if available
 - > `ServletRequest.isAsyncSupported()`

Servlet 3.0 Async Processing (2)

- Entering async mode
 - > `ServletRequest.startAsync()` → `AsyncContext`
- Choice of how to process the request
- Redispatch to self
 - > `actx.dispatch()`
- Redispatch to other servlet
 - > `actx.dispatch(servletName)`
- Process asynchronously
 - > `actx.start(aRunnable)` then `actx.complete()` later

EJB 3.1 Highlights

- Singleton beans
 - > `@Singleton @ConcurrencyManagement @Startup`
- No-interface view
- Calendar timers
 - > `@Schedule(dayOfWeek="Fri", hour="12")`
- Non-persistent timers
 - > `@Schedule(minute="*/10", hour="*", persistent=false)`
- Async business methods
 - > `@Asynchronous public Future<Result> compute(...)`

EJB 3.1 Lite

- Simple, modern subset of EJB for use outside of the full platform
- Contents:
 - > Session beans (stateful, stateless, singletons)
 - > Transaction and security attributes
 - > Interceptors
 - > ejb-jar.xml
- Embeddable container API
- Bootstraps on Java SE
- Beans looked up in JNDI by global name

EJB Bootstrap Sample

```
public class BankTester {  
    public static void main(String[] args) {  
  
        EJBContainer container = EJBContainer.createEJBContainer();  
  
        Bank bank = (Bank) container  
                                .getContext()  
                                .lookup("java:global/bank/BankBean");  
        testAccountCreation(bank);  
        // ...  
    }  
}
```

EJB Components in a Web Module

- EJB components can be defined directly inside a web module (war file)
- Things to keep in mind:
 - > java:comp is shared by all components
 - > A single class loader is used
 - > Local view scoped to the web module
 - > Descriptor WEB-INF/ejb-jar.xml
or META-INF/ejb-jar.xml in a jar inside WEB-INF/lib
 - > No entity beans

Multiple JNDI scopes

- Familiar with `java:comp`
- Now adding:
 - > `java:module` module-scoped, like inside a war file
 - > `java:application` application-scoped
 - > `java:global` scoped to multiple applications
- Usable throughout
 - > `@Resource(lookup="java:module/env/db") DataSource db;`
- EJBs are automatically registered
 - > `java:global/myApp/myModule/MyBean!com.acme.MyInterface`
 - > `java:module/MyBean // shorthand`

Bean Validation (JSR-303)

- Now included in Java EE 6
- Integrated with JSF 2.0 and JPA 2.0
- Examples:
 - > `@NotNull @Size(max=40) private String streetName;`
 - > `@NotNull @Valid private Country country;`
- TraversableResolver for object graph validation
- Validation API
 - > `Set<ConstraintViolation> Validator.validate(Object)`
- Validator and ValidatorFactory injectable

JSR-299 (formerly Web Beans)

- Type-based dependency injection
 - > `@LoggedIn User user;`
 - > `@Produces @LoggedIn User getLoggedInUser() { ... }`
- Context management
 - > `@RequestScoped`, `@SessionScoped`, `@ConversationScoped`, ...
- Heavy use of meta-annotations
 - > `@BindingType public @interface LoggedIn {`
- Powerful SPI
 - > `BeanManager`, `Bean`, `InjectionPoint`, `Context`

JSR-299 Basics

- Resolution is by *bean type* and *binding type(s)*
 - > bean type + set(binding types) → bean
- Not string based!
- @LoggedIn User = (User, @LoggedIn) → bean
- Beans are enabled/disabled in batches via *deployment types* *
- Put as much info as possible on bean class
 - > @ConversationScoped @Current

```
public class ShoppingCart {}
```

JSR-299 Example

```
@ConversationScoped @Current  
public class ShoppingCart { ... }
```

```
@PayByCreditCard  
public class CreditCardPaymentProcessor implements PaymentProcessor  
{ ... }
```

```
public class Checkout {  
    private @Current ShoppingCart cart;  
    private @PayByCreditCard PaymentProcessor payByCard;  
  
    public void checkout() { ... }  
}
```

Schedule

- Proposed final draft for almost all specs
- Final release September 2009
- Implementation: GlassFish V3

Q&A



Java EE 6: Renewing the Platform

Roberto Chinnici

roberto.chinnici@sun.com