

Connecticut Java User Group

Java EE 6: A Community Update

Reza Rahman

Author, *EJB 3 in Action*

Expert Group Member, Java EE 6 and EJB 3.1

Founder, Cognicellence

October, 2008

Java EE 6: Expanding Horizons

- **Java EE 5**
 - **Ease-of-use, annotations, freedom from XML, API expansion**
 - **EJB 3.0, JPA 1.0, JSF 1.0, JAX-WS 2.0**
 - **The demise of EJB 2.x Entity Beans**
- **Java EE 6**
 - **Pruning: A healthful maturity**
 - **Profiles: One size doesn't fit all**
 - **Extensibility: Making things more friendly for non-standard solutions**
 - **Innovation: New APIs, new features, further ease-of-use**
- **WebBeans, JSF 2.0, EJB 3.1, JPA 2.0, Bean Validation, Servlet 3.0, JAX-RS**
- **SpringSource participating in some JSR groups, including Java EE 6 (JSR 317), of which Rod Johnson is a member**

Pruning

- An important first step in the life-cycle of a mature platform
- The goal is to “deprecate” APIs that are out-of-date or have been superseded
- Candidates for pruning:
 - **JAX-RPC: Superseded by the popular JAX-WS API**
 - **EJB 2.x Entity Beans CMP: Dropped in favor of JPA**
- APIs will be made optional, then removed eventually
- Some larger vendors may choose to keep deprecated APIs for some time

Profiles

- **Specific sub-sets of Java EE APIs intended for specific environments**
- **Each Profile is fully integrated and “just works” out-of-the-box, although integrating add-ons is still possible**
- **Makes creating modular, lightweight Java EE compliant application servers a lot easier**
- **Only one Profile, the “Web Profile” was initially planned**
- **However, there is already broad support for two Profiles intended for web development, one “minimal” and one “intermediate”**
- **Your feedback is needed on what these profiles should really look like!**

Suggested Profiles

API	Minimal Profile	Intermediate Profile	Full Profile
Servlet 3.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JSP 2.2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JSTL 1.2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EL 1.2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JSF 2.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
WebBeans 1.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EJB 3.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JPA 2.0		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JTA 1.1		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
JMS 1.1			<input checked="" type="checkbox"/>
JavaMail 1.4			<input checked="" type="checkbox"/>
JAX-WS 2.2			<input checked="" type="checkbox"/>
JAX-RS 1.0			<input checked="" type="checkbox"/>
JAXR 1.0			<input checked="" type="checkbox"/>
JAXB 2.2			<input checked="" type="checkbox"/>
JCA 1.6			<input checked="" type="checkbox"/>

Extensibility

- This is still at a very preliminary stage
- Extensibility and Profiles work together
- Profiles, particularly the Web Profile, make it possible to extend the platform by integrating non-standard APIs without serious loss of compatibility/fragmentation
- Transparent dependency injection and interceptor attachment, geared towards third-party vendors, can make non-standard technology integration a lot easier without making things more complex for developers
- Could be made as simple as dropping a JAR and annotating classes/interfaces
- Java EE service provider APIs (SPIs) are an existing extension mechanism that could be fully leveraged as well
- **Your feedback is needed on what Java EE extensibility should really look like!**

Java EE Extensibility Example

```
@Stateless
public class PlaceBidBean implements PlaceBid {
    @SqlMapClientBuilder(configuration="sql-map-config.xml")
    private SqlMapClient sqlMapClient;

    public void placeBid(Bid bid) {
        sqlMapClient.insert("insertBid", bid);
    }
}

@Local
public interface PlaceBid {
    public void placeBid (Bid bid);
}
```

Major API Changes in Java EE 6

- **WebBeans 1.0**
 - **JSF/EJB 3/JPA integration**
- **Java Server Faces (JSF) 2.0**
 - **Ease-of-use, technology adoption, new features**
- **Enterprise Java Beans (EJB) 3.1**
 - **Ease-of-use, new features**
- **Java Persistence API (JPA) 2.0**
 - **New features**
- **Bean Validation**
 - **Defining data constraints once per application**
- **Servlet 3.0**
 - **Ease-of-use, new features**
- **Java API for RESTful Web Services (JAX-RS) 1.0**
 - **API for developing REST based web services in addition to SOAP support in JAX-WS**

WebBeans 1.0 (JSR 299)

- **Unifies the JSF, JPA and EJB 3 programming models**
- **Stateful, contextual web development**
- **Conversations**
- **Robust, type-safe dependency injection**
- **Enhancements to the Interceptor model**
- **Annotations meta-programming**

JSF Page Using WebBeans

```
<h:form>
  <table>
    <tr>
      <td>Bidder</td>
      <td><h:inputText value="#{bid.bidder}"/></td>
    </tr>
    <tr>
      <td>Item</td>
      <td><h:inputText value="#{bid.item}"/></td>
    </tr>
    <tr>
      <td>Bid Amount</td>
      <td><h:inputText value="#{bid.bidPrice}"/></td>
    </tr>
  </table>
  ...
  <h:commandButton type="submit" value="Add Bid"
    action="#{placeBid.addBid}"/>
  ...
</h:form>
```

JPA Entity WebBeans Component

```
@Entity
@Component
@Named("bid")
@Table(name="BIDS")
public class Bid {
    @Id
    @GeneratedValue
    @Column(name="BID_ID")
    public Long bidId;
    public String bidder;
    public String item;

    @Column(name="BID_PRICE")
    public Double bidPrice;
}
```

EJB 3.1 Session Bean WebBeans Component

```
@Stateful
@RequestScoped
@Component
@Named("placeBid")
public class PlaceBidBean {
    @PersistenceContext
    private EntityManager entityManager;

    @Current
    private Bid bid;

    public void addBid() {
        entityManager.persist(bid);
    }
}
```

Java Server Faces 2.0 (JSR 314)

- **Simplified custom components**
- **First-class Facelet support**
- **First-class Ajax support**
- **Annotation-driven configuration**
- **Convention over configuration**
- **Optional faces-config.xml**
- **RAILS_ENV style development stages**
- **Resources**
- **Data validation (JSR 303)**
- **Improving performance**
- **REST (JSR 311) support**
- **Many others**

JSF Project Stages

```
<context-param>  
  <param-name>javax.faces.PROJECT_STAGE</param-name>  
  <param-value>Development</param-value>  
</context-param>
```

javax.faces.application.ProjectStage

- Production
- Development
- UnitTest
- SystemTest
- Extension

JSF Resources

Resource path:

```
[ /META-INF ] /resources /  
[ localePrefix / ] [ libraryName / ] [ libraryVersion / ]  
resourceName [ /resourceVersion ]
```

```
Resource resource = FacesContext.getApplication()  
    .getResourceHandler()  
    .createResource(resourceName[ , libraryName ] );
```

```
# { resource [ ' <libraryName> : <resourceName> ' ] }
```

```
@ResourceDependency(name="style.css", library="actionbazaar")  
public class HeaderTabs extends UIComponentBase {  
    ...  
}
```

Enterprise Java Beans 3.1 (JSR 318)

- **Session Bean optional interfaces**
- **Singleton Beans with concurrency control**
- **Cron-style declarative and programmatic Timers**
- **Asynchronous bean invocation**
- **Simplified WAR packaging**
- **Java SE support**
- **Standardized Global JNDI naming**
- **EJB Lite**

Cron-like Declarative Timers

`@Stateless`

```
public class NewsletterGeneratorBean {  
    @Resource  
    private Session mailSession;  
  
    @Schedule(second="0", minute="0", hour="0",  
              dayOfMonth="1", month="*", year="*")  
    public void generateMonthlyNewsletter() {  
        ...  
    }  
}
```

Asynchronous Session Bean

```
@Stateless
public class OrderBillingBean {
    ...
    @Asynchronous
    public Future<BillingStatus> billOrder(Order order) {
        try {
            bill(order);
            return new AsyncResult<BillingStatus>(
                BillingStatus.COMPLETE);
        } catch (BillingException be) {
            return new AsyncResult<BillingStatus>(
                BillingStatus.BILLING_FAILED);
        }
    }
    ...
}
```

Asynchronous Invocation Client

@EJB

```
private OrderBillingBean orderBilling;
...
Order order = new Order();
...
Future<BillingStatus> future = orderBilling.billOrder(order);
...
BillingStatus status = future.get();
...
if (status == BillingStatus.COMPLETE) {
    notifyBillingSuccess(order);
} else if (status == BillingStatus.BILLING_FAILED) {
    notifyBillingFailure(order);
}
```

Java Persistence API 2.0 (JSR 317)

- **Object-relational mapping enhancements**
 - **Collections, maps and ordered lists**
 - **Unidirectional one-to-many mapping**
 - **Join tables for one-to-one, many-to-one**
- **Criteria API**
- **Bean validation (JSR 303)**
- **EntityManager and Query API additions**
 - **Locking, loading Entity, clearing Entity, max results, first result**
- **Second-level caching**
- **More standard properties and hints**
 - **JDBC, locking, caching**

Mapping Collections

```
@Entity
@Table(name="USERS")
public class User {
    @Id
    @GeneratedValue
    @Column(name="USER_ID")
    public long userId;
    public String userName;
    @Column(name="BIRTH_DATE")
    public Date birthDate;
    ...
    @ElementCollection
    @CollectionTable(name="ALIASES")
    @Column(name="ALIAS")
    public Set<String> aliases;

    @ElementCollection
    public Map<String, String> photos;
}
```

Unidirectional One-to-Many Relationship

```
@Entity
public class User {
    @Id @GeneratedValue
    public long id;
    public String userName;
    ...
    @OneToMany
    @JoinColumn(name="USER_ID")
    public Set<Phone> phones;
}
```

```
@Entity
public class Phone {
    @Id @GeneratedValue
    public long id;
    public String type;
    public String number;
    ...
}
```

Bean Validation (JSR 303)

- **Specify constraints only once across application layers**
- **Constraint**
 - **Restriction on a bean, field or property**
 - **Not null, between 10 and 45, valid email, etc**
 - **Evaluated automatically by a framework**
- **Useful in other Java SE/Java EE APIs**
 - **JSF 2.0**
 - **JPA 2.0**
 - **EJB 3.1**
 - **WebBeans**

JPA Entity with Bean Validation

```
@Entity
@Table(name="BIDS")
public class Bid {
    @Id
    @GeneratedValue
    @Column(name="BID_ID")
    public Long bidId;

    @NotNull
    @Length(max=30)
    public String bidder;

    @NotNull
    @Length(max=200)
    public String item;

    @Column(name="BID_PRICE")
    @NotNull
    @Min(value=0.0, message="price negative")
    public Double bidPrice;
}
```

Servlet 3.0 (JSR 315)

- Annotations from the ground-up
- Optional web.xml
- Intelligent defaults
- Modular web.xml fragments in framework library jars
- Programmatic addition of Servlets, Filters and Listeners through the ServletContext
- Asynchronous processing support in Servlets
 - Suspend and resume requests

Servlet Annotations Example

```
@Servlet(name="PlaceBidServlet"  
         urlMapping={"/bid", "/place-bid"})  
public class PlaceBidServlet {  
    @EJB  
    private PlaceBid placeBid;  
  
    @GET  
    public void doGet(HttpServletRequest request,  
                      HttpServletResponse response) {  
        Bid bid = new Bid();  
        ...  
        placeBid.placeBid(bid);  
        ...  
    }  
}
```

Programmatic Servlet Addition

```
@ServletContextListener
public class ActionBazaarListener {
    public void contextInitialized(
        ServletContextEvent event) {
        ServletContext context = event.getServletContext();
        context.addServlet(
            "PlaceBidServlet",
            "Place bid servlet",
            "actionBazaar.PlaceBidServlet",
            null, -1);
        context.addServletMapping(
            "PlaceBidServlet",
            new String[]{"/place-bid"});
    }
}
```

Java API for RESTful Web Services (JSR 311)

- **Web services through REST instead of SOAP**
- **REST counterpart of JAX-WS**
- **Gets rid of low-level code so you can focus on core logic**
- **Annotations from the ground-up**
- **Could be integrated with WebBeans, EJB and JPA**

Session Bean using JAX-RS

```
@Stateless
@Path("/webservices")
public class PlaceBidBean {
    @PersistenceContext
    private EntityManager entityManager;

    @PUT
    @Path("/bid/{bidder}")
    public void placeBid(
        @PathParam("bidder")
        String bidder,
        @QueryParam("item")
        String item,
        @QueryParam("bid_price")
        Double bidPrice) {
        entityManager.persist(
            new Bid(bidder, item, bidPrice));
    }
}
```

Summary

- **Pruning**
 - **JAX-RPC, EJB 2.x Entity Beans CMP**
- **Profiles**
 - **Minimal and intermediate Web Profiles?**
- **Extensibility**
 - **Generalized dependency injection and interceptor attachment?**
 - **Additional SPIs?**
- **Innovation**
 - **WebBeans, JSF 2.0, EJB 3.1, JPA 2.0, Bean Validation, Servlet 3.0, JAX-RS**
- **Your help is needed**
 - **Send comments to jsr-316-comments@jcp.org!**
 - **Copy me at reza_rahman@lycos.com**

References

- Profiles in the Java EE 6 Platform, http://weblogs.java.net/blog/robc/archive/2008/02/profiles_in_the_1.html
- Java EE 6, <http://jcp.org/en/jsr/detail?id=316>
- WebBeans, <http://jcp.org/en/jsr/detail?id=299>
- JSF 2.0, <http://jcp.org/en/jsr/detail?id=314>
- EJB 3.1, <http://jcp.org/en/jsr/detail?id=318>
- JPA 2.0, <http://jcp.org/en/jsr/detail?id=317>
- Bean Validation, <http://jcp.org/en/jsr/detail?id=303>
- Servlet 3.0, <http://jcp.org/en/jsr/detail?id=315>
- JAX-RS, <http://jcp.org/en/jsr/detail?id=311>