Java EE 6 Overview

Reza Rahman
Expert Group Member, Java EE 6
Resin Container Developer
Author, EJB 3 in Action
reza@caucho.com

Java EE 6: New Horizons

- **Java EE 5**
  - **Annotations, convention-over-configuration, freedom from XML**
  - **EJB 3, JPA, JSF, JAX-WS**
- **Java EE 6**
  - **Pruning: Cutting the dead wood**
  - **Profiles: Enabling lightweight application servers**
  - **Innovation: New APIs, new features, further ease-of-use**
- **Managed beans, CDI, JSF 2, EJB 3.1, JPA 2, Servlet 3, JAX-RS, bean validation**

Pruning

- **The goal is to “deprecate” APIs that are out-of-date or have been superseded**
- **Pruned APIs:**
  - **JAX-RPC: Superseded by JAX-WS**
  - **EJB 2.x Entity Beans CMP: Dropped in favor of JPA**
  - **JAXR: UDDI not well used**
  - **Java EE Application Deployment (JSR-88): Poor support**

Profiles

- **Specific sub-sets of Java EE APIs intended for specific types of applications**
- **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 (such as Resin which only implements the Java EE 6 Web Profile)**
- **Only one Profile, the “Web Profile” is initially planned**

Java EE 6 Web Profile

API	Web Profile	Full Profile
Servlet 3	✔	✔
JSF 2	✔	✔
CDI	✔	✔
EJB 3.1*	✔	✔
JPA 2	✔	✔
Bean validation	✔	✔
JTA	✔	✔
JMS		✔
JavaMail		✔
JAX-WS		✔
JAX-RS		✔
JAXB		✔
JCA		✔
JACC		✔

Major API Changes

- **Contexts and Dependency Injection (CDI)**
  - **Next generation dependency injection**
- **Java Server Faces (JSF) 2**
  - **Ease-of-use, technology adoption, new features**
- **Enterprise Java Beans (EJB) 3.1**
  - **Ease-of-use, new features**
- **Java Persistence API (JPA) 2**
  - **More flexibility, new features**
- **Servlet 3**
  - **Ease-of-use, new features**
- **Java API for RESTful Web Services (JAX-RS)**
  - **REST based web services in addition to SOAP support in JAX-WS**
- **Bean Validation**
  - **Expressing application constraints declaratively**

Contexts and Dependency Injection

- **Type-safe generic dependency injection**
- **Automatic context management**
- **Unifies JSF, JPA and EJB 3 programming models**
- **Conversations**
- **Interceptors/decorators**
- **Annotations meta-programming**
- **Portable extensions**

JSF Using CDI

```
<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.price}"/></td>
        </tr>
    </table>
    ...
    <h:commandButton type="submit" value="Add Bid"
        action="#{placeBid.addBid}"/>
    ...
</h:form>
```

JPA Entity as JSF Model

```
@Entity
@Table(name="BIDS")
public class Bid {
    @Id
    @GeneratedValue
    @Column(name="BID_ID")
    private Long id;
    private String bidder;
    private String item;

    @Column(name="BID_PRICE")
    private Double price;
    ...
}
```

EJB 3.1 Session Bean as JSF Event Handler

```
@Stateful @RequestScoped @Named
public class PlaceBid {
    @PersistenceContext
    private EntityManager entityManager;

    @Produces @Named
    private Bid bid = new Bid();

    @Inject @Utility
    private CurrencyTools tools;

    @Audited
    public void addBid() {
        bid.setPrice(tools.round(bid.getPrice()));
        entityManager.persist(bid);
    }
}
```

CDI Managed Bean with Qualifier

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

@Utility
@ApplicationScoped
public class DefaultCurrencyTools
        implements CurrencyTools {
    ...
    public double round(double value) {
        BigDecimal converter =
            new BigDecimal(Double.toString(value));
        converter = converter.setScale(DECIMAL_PLACES,
            BigDecimal.ROUND_HALF_UP);
        return converter.doubleValue();
    }
    ...
}
```

CDI Interceptor

```
@InterceptorBindingType
@Target({TYPE, METHOD})
@Retention(RUNTIME)
public @interface Audited {}

@Audited @Interceptor
public class AuditInterceptor {

    @AroundInvoke
    public Object audit(InvocationContext context)
            throws Exception {
        System.out.println("Entering: "
            + context.getMethod().getName());
        System.out.println("  with args: "
            + context.getParameters());
        return context.proceed();
    }
}
```

Java Server Faces 2

- **First-class support for Facelets as view/custom components/templates**
- **Annotation-driven configuration**
- **Ajax support in the JSF life-cycle**
- **Bookmarking/navigation/parameters**
- **Resources**
- **EL parameters**
- **API/Component improvements**

Facelet Components

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ez="http://java.sun.com/jsf/composite/ezcomp">
<h:head>
    <title>A simple example of EZComp</title>
</h:head>
<h:body>
<h:form>
    <ez:loginPanel id="loginPanel">
        <f:actionListener for="loginEvent"
            binding="#{bean.loginEventListener}" />
    </ez:loginPanel>
</h:form>
</h:body>
</html>
```

Facelet Custom Component

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:composite="http://java.sun.com/jsf/composite">
<body>
  <composite:interface>
    <composite:actionSource name="loginEvent" />
  </composite:interface>
  <composite:implementation>
    <p>Username: <h:inputText id="usernameInput" /></p>
    <p>Password: <h:inputSecret id="passwordInput" /></p>
    <p><h:commandButton id="loginEvent" value="login"/>
  </composite:implementation>
</body>
</html>
```

JSF Annotations

```
@ManagedBean(name=“accountCreator")
@RequestScoped
public class AccountCreatorBean {
    @ManagedProperty(value="#{user}")
    private User user;

    @EJB
    private AccountService accountService;
    ...
    public String createAccount() {
       ...
    }
}
```

Enterprise Java Beans 3.1

- **EJBs are managed beans with additional services like transactions**
- **Singleton Beans with concurrency control**
- **Cron-style declarative and programmatic Timers**
- **Asynchronous bean invocation**
- **Simplified WAR packaging**
- **Embedded Containers/Testing support**
- **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<BilllingStatus>(
                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

- **Object-relational mapping enhancements**
  - **Embedded objects, collections, maps and ordered lists**
  - **Unidirectional one-to-many mapping**
  - **Join tables for one-to-one, many-to-one**
- **Query and EntityManager API enhancements**
  - **First result, max result, unwrapping, typed results, detach entities**
- **JPQL enhancements**
  - **CASE, NULLIF, COALESCE**
- **Criteria API/Meta-model**
- **Second-level caching**
- **Pessimistic locking**

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;
    ...
}
```

Criteria API

```
CriteriaBuilder criteriaBuilder =
    entityManager.getCriteriaBuilder();
CriteriaQuery<User> criteriaQuery =
    criteriaBuilder.createQuery(User.class);
Root<User> user = criteriaQuery.from(User.class);
criteriaQuery
  .select(user)
  .where(criteriaBuilder.equal(
            user.get(“firstName"), “John”),
         criteriaBuilder.equal(
            user.get(“lastName"), “Smith”));

SELECT user
FROM User user
WHERE user.firstName = ‘John’
AND user.lastName = ‘Smith’
```

Servlet 3

- **Annotations from the ground-up**
- **Modular web.xml fragments in framework library jars**
- **Programmatic addition of Servlets, Filters and Listeners through the ServletContext**
- **Servlet container initializers**
- **Asynchronous processing support in Servlets**

Servlet Annotations

```
@WebServlet(name="PlaceBidServlet"
            urlPatterns={"/bid", "/place-bid"})
public class PlaceBidServlet extends HttpServlet {
    @EJB
    private PlaceBid placeBid;

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) {
        Bid bid = new Bid();
        ...
        placeBid.placeBid(bid);
        ...
    }
}
```

Programmatic Servlet Addition

```
@WebListener
public class ActionBazaarListener
        implements ServletContextListener {
    public void contextInitialized(
        ServletContextEvent event) {
        ServletContext context = event.getServletContext();
        ServletRegistration registration
            = context.addServlet(
                "PlaceBidServlet",
                "actionBazaar.PlaceBidServlet");
        registration.addMapping(
            "PlaceBidServlet",
            new String[]{"/place-bid"});
    }
}
```

Java API for RESTful Web Services

- **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**
- **Integrated with CDI and EJB**

JAX-RS with Session Bean

```
@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));
    }
}
```

Bean Validation

- **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**
  - **JPA 2**

JPA Entity with Bean Validation

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

    @NotNull
    @Size(min=5, max=30)
    public String bidder;

    @NotNull
    @Size(min=10, max=200)
    public String item;

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

Summary

- **Pruning**
  - **Chopping dead wood**
  - **JAX-RPC, EJB 2.x Entity Beans, JAXR, JSR-88, JSR-77**
- **Profiles**
  - **Web profile geared towards majority of Java EE applications**
  - **Lightweight, modular application servers (e.g. Resin)**
- **Innovation**
  - **Managed beans, CDI, JSF 2, EJB 3.1, JPA 2, Servlet 3, JAX-RS, bean validation**

References

- **Introducing Java EE 6**
  - http://java.sun.com/developer/technicalArticles/JavaEE/JavaEE6Overview.html
- **Java EE 6 Tutorial**
  - http://java.sun.com/javaee/6/docs/tutorial/doc/
- **GlassFish**
  - http://java.sun.com/javaee/community/glassfish/
- **Resin**
  - http://www.caucho.com/projects/resin/
- **Java EE 6 Code Examples**
  - http://java.sun.com/javaee/reference/code/