J2EE vs Java EE

Sanjeeb Sahoo
Staff Engineer, Sun Microsystems
Objective

- Learn the new features of Java Platform, Enterprise Edition (Java EE) that make enterprise application development a much easier task.
Agenda

● Evolution of the Java Platform, Enterprise Edition

● Java EE 5
  – EoD features
  – Demo
  – Comparative Analysis

● How can you participate?

● (Java EE).next

● Q & A
The journey begins...

- May 1995
  - Java programming language
- May 1998
  - EJB
  - Announcement of JPE project
- Guiding principles
  - JCP driven
  - Multi-vendor, multi-platform
  - WORA, CTS
  - 100% Backward compatibility (source & binary)
The journey continues...

- Dec 1999 (J2EE 1.2)
  - Servlet, JSP, EJB, JMS, JTA, JNDI, RMI/IIOP

- Sept 2001 (J2EE 1.3)
  - Robustness
  - EJB 2.0, Connector Architecture

- Nov 2003 (J2EE 1.4)
  - Integration (EIS, Tools)
  - Web Services, Asynchronous Connectors, Management, Deployment
J2EE 1.4

- J2EE is enormously powerful
  - The industry standard for robust enterprise apps
- But that power sometimes gets in the way
  - Too difficult to get started
  - Even simple apps need boring boilerplate
- Can we keep the power… but make typical development tasks simpler?
  - **YES**… and that is the focus of Java EE 5!
Java EE 5 = (J2EE 1.4).next

Make it easier to develop enterprise Java applications

Especially when first getting started with Java EE

Released – June 2006
How did we make it easier?

- Declarative programming
  - Originally - deployment descriptors
  - Now - Java language annotations
- Remove requirements
  - Plain Old Java Objects (POJOs)
  - More and better defaults
    - earlier it was very defensive programming
- New & more powerful frameworks
  - Less work for you to do

Easier to learn and more productive!
Major Features in Java EE 5

- Annotations are *the law*
- Dependency injection
- Simplified web services support
- More web service standards support
- Greatly simplified EJB™ development
- New Java™ Persistence API
- Easy web applications with JavaServer™ Faces
- And fully compatible with J2EE 1.4
Annotations

- What is Java annotation?
  - a Java annotation is a way of adding metadata to Java source code
  - Introduced in J2SE 1.5 (JSR 175)

- How is it defined?
  ```java
default class Entity {
    String name() default "";
}
```
Annotations

• How is it used in source code?
  @Entity public class Employee { ... }

• Is it available at runtime?
  • depends

• How is the metadata accessed at runtime?
  • reflection APIs (e.g.) Class.getAnnotations(),
    Class.getAnnotation(Class), java.lang.annotation.*
Benefits of using annotations

- Reduced need for XML deployment descriptors
- Co-location of source code & configuration information makes it easier to maintain
- Is there any performance impact?
  - Mostly no – depends on container
  - Generation of full deployment descriptor during deployment is a common strategy
Annotations in Java EE 5

- How are they defined?
  - Most of them have @Retention(RUNTIME)

- Where are they defined?
  - As part of different specs, in different packages.
    - `javax.annotation.*`, `javax.annotation.security.*` (JSR 250)
    - `javax.ejb.*`, `javax.interceptor.*`, `javax.persistence.*` (JSR 220)
    - `javax.jws.*`, `javax.jws.soap.*` (JSR 181)
    - `javax.xml.bind.annotation.*` (JSR 222)
    - etc.
Annotations in Java EE 5

• How are they used in Java EE?
  • To map Java classes to databases
  • To map Java classes to XML
  • To define EJBs and their attributes
  • To define Web Services
  • To define security attributes
  • To specify external dependencies
import javax.xml.bind.annotation.*;
@XmlElement(name="employee")
@XmlElement(name="EmployeeType")
public class Employee {
    public String name;
    public int salary;
}
Security Annotations

```java
import javax.annotation.security.*;
import javax.ejb.*;
@Stateless
@RolesAllowed("javaee")
public class HelloEJB implements Hello {
    @PermitAll // overrides class level annotation
    public String hello(String msg) {...}
    public String bye(String msg) {...}
}
```
How to override annotations?

- XML DD can be used to override annotations
  - Remember, no annotations often mean default values applied by container

- Partial overriding
  - XML DD & annotations can co-exist
  - Not everything can be overridden
  - You must use Java EE 5 version of the XML DD

- Complete overriding
  - metadata-complete = true
Vendor specific annotations

• .java file is not portable
  • To compile, you need vendor specific annotations in classpath

• .class is portable
  • annotations are treated specially by VM during loading
  • Can be used in compilation of other sources
  • Can be used in runtime, but semantics may differ
Dependency Injection

- Example of Inversion of Control
- Container “injects” resources...
  - DataSource, EJB ref, web service ref, persistence units, UserTransaction, env entries,...
  - ... into application ...
    - Fields, methods; public, private, or protected
  - ... in container-managed classes
    - EJBs, servlets, JSF managed beans, web service endpoints, handlers, interceptors, app clients
- Avoids the need to use JNDI
public class MyEJB implements SessionBean {
    private DataSource myDS;
    public void ejbCreate() {
        try {
            InitialContext ctx = new InitialContext();
            myDS = (DataSource) ctx.lookup("employeeDatabase");
        } catch (NamingException ex) {
            // XXX - what to do?
        }
    }
    ...
}

• Plus corresponding deployment descriptor entry
Dependency Injection

```java
cpyackage com.example;

@Stateless @Local(MyEJBInterface.class)
public class MyEJB {
    @Resource
    private DataSource myDS;
    ...
}
```

- Declares a resource named `com.example.MyEJB/myDS`
- And injects it into the `myDS` field
- No deployment descriptor entry needed!
package com.example;

@Stateless @Local(MyEJBInterface.class)
public class MyEJB {
    @Resource(name = "employeeDatabase")
    private DataSource myDS;
    ...
}

• Declares a resource named employeeDatabase
• And injects it into the myDS field
• Still no deployment descriptor entry needed!
package com.example;
@Stateless @Local(MyEJBInterface.class)
public class MyEJB {
    @Resource(mappedName = "OracleDatabase")
    private DataSource myDS;
    ...
}

- Declares a resource that's mapped to the app server's global resource named OracleDatabase
- And injects it into the myDS field
- And still no deployment descriptor entry needed!
Dependency Injection

- Annotations used for Dependency Injection
  - @javax.annotation.Resource
  - @javax.ejb.EJB
  - @javax.xml.ws.WebServiceRef
  - @javax.persistence.PersistenceContext
  - @javax.persistence.PersistenceUnit
Dependency Injection

Possible without using annotations!

```java
package foo;

@Stateless @Remote(MyEJBInterface.class)
public class MyEJB {
    private DataSource myDS;
    ...
}

<resource-ref>
    <injection-target>
        <injection-target-class-name>foo.MyEJB</injection-target-class-name>
        <injection-target-name>myDS</injection-target-name>
    </injection-target>
</resource-ref>
```
Dependency Injection

Is this dependency injection?

```java
package foo;

@Resource(name="jdbc/MyDataSource",
    type=DataSource.class)
public class MyServlet extends HttpServlet{
    ...
}
```
Dependency Injection

- Only for managed classes
- Injection is performed before `@PostConstruct` callback
- In application client container, injection target must be static
- Dependency injection annotations when used at class level do not perform any injections!
  - They just declare dependency
- Any use of mappedName is not only non-portable, but also very installation specific.
Simpler packaging rules

- Packaging is now much simpler
- Many cases don't require deployment descriptors
- library folder
  - like WEB-INF/lib, ear now has a lib folder
  - default name is lib
  - can be overridden by use application.xml
  - makes bundled optional package easy to use
Simpler packaging rules

- Rules and conventions make for simpler packaging
- `.war` files are web applications
- `.rar` files are resource adapters
- `lib` directory contains shared jar files
- `.jar` file with `Main-Class` is an app client
- `.jar` file with `@Stateless` or `@Stateful` or `@MessageDriven` is an EJB module
More Flexibility

- Java Persistence API entity classes can be packaged in
  - EJB jar
  - WEB-INF/classes
  - WEB-INF/lib/*.jar
  - application client jar
  - any non-component jar file in ear
EAR Packaging Example

app.ear

  lib/shared.jar (entities, ejb-interface classes)
  biz/ejb.jar
  ui/web.war
  ui/client.jar

• That's it!
  - No META-INF/application.xml
  - No Class-Path entry in MANIFEST.MF
New APIs and frameworks

- EJB 3.0
- Java Persistence API 1.0
- JAX-WS 2.0
- JAXB 2.0
- StAX – pull parser
- JSF, JSTL
JAXB 2.0 (JSR 222)

- Bidirectional binding
  - From XML schema to Java class
  - From Java classes to XML Schema (new)
    - JAXB 1.0 was XML to Java only
- 100% XML Schema support
- Portable annotation driven architecture
- Databinding for JAX-WS 2.0
Web Services in Java EE 5

- JAX-WS 2.0
  - Follow-on to JAX-RPC 1.0
  - JAX-RPC 2.0 (JSR 224) renamed as JAX-WS 2.0
- Includes JSR 181
  - Web Services Metadata for Java platform
- JAXB 2.0 as standard data binding framework
- Supports latest W3C standards
  - SOAP 1.2, XML Schema 1.0
- Supports latest WS-I standards
  - Basic Profile 1.1, Attachment Profile 1.0
JavaServer Faces 1.2

- The Java EE Standard Web Application Framework
- Provides support for fine grained UI event handling
- Provides a clean separation between behavior and presentation
- Framework for building custom reusable UI components
- In-built validation and error reporting framework
- Dependency injection in managed beans
New Web 2.0 Java Pet Store: Built with AJAX-enabled JSF Components

Java Pet Store

Auto-complete

Name: 
Street: 
City: Astronburg
State: Astronburg, PA
Zip: 
Submit

Ratings

Quantity | Description | Price | Rating
--- | --- | --- | ---
1 | Brown Fug | $59.00 | 5
1 | Furry Cat | $89.00 | 5
1 | Grumpy Pelican | $100.00 | 1

RSS Reader

News from BluePrints

SystemNews discusses Java BluePrints AJAX Components

File Upload

Google Maps Mashup

Popup Balloon

News from BluePrints

SystemNews discusses Java BluePrints AJAX Components

News from BluePrints

SystemNews discusses Java BluePrints AJAX Components

News from BluePrints

SystemNews discusses Java BluePrints AJAX Components

News from BluePrints

SystemNews discusses Java BluePrints AJAX Components
Java Persistence API 1.0

• Single persistence API for Java EE and Java SE

• Developed by EJB expert group
  • Builds on years of experience with existing technologies and products

• At least three implementations (all open source):
  • Oracle – GlassFish/TopLink Essentials
  • JBoss – Hibernate
  • BEA – Kodo/OpenJPA
Java Persistence API 1.0

- POJO-based
  - Much simpler than EJB CMP
- Standardized O/R mapping
  - using annotations or XML mapping strategy
- Support for inheritance & polymorphism
- Support for optimistic locking
- Container/Provider pluggability
- Powerful query language
- Support for native queries
Demo
How Much Easier Is It?

- **Adventure Builder**¹
  - J2EE 1.4 – 67 classes, 3284 lines of code
  - Java EE 5 – 43 classes, 2777 lines of code
  - 36% fewer classes to manage!

- **RosterApp**²
  - J2EE 1.4 – 17 classes, 987 lines of code
  - Java EE 5 – 7 classes, 716 lines of code
  - J2EE 1.4 XML files – 9 files, 792 lines
  - Java EE 5 XML files – 1 file, 5 lines
  - 58% fewer classes, 89% fewer XML files to manage!

¹ Source: Debu Panda, Oracle
² Source: Raghu Kodali, Oracle
Agenda

- Evolution of the Java Platform, Enterprise Edition
- Java EE 5
  - How can you participate?
- (Java EE).next
- Q & A
Project GlassFish

Simplifying Java application development with **Java EE 5 technologies**
Includes JAX-WS 2.0, JAXB 2.0, JSF 1.2, EJB 3.0, and Java Persistence 1.0
Supports > 20 frameworks and apps

**Open source** CDDL license
**Basis for the** Sun Java System Application Server PE 9
**Free** to download and **free** to deploy

Building a Java EE 5 open source application server
Java.sun.com/javaee/GlassFish

Source: Sun 2/06 – See website for latest stats
Project GlassFish

- Java EE 5 compliant Application Server
  - Reference Implementation
  - Included in Java EE 5 SDK
- Open Source
  - OSI license – CDDL
  - GPL v2 (coming soon)
- Community at Java.Net
  - Sources, bug DBs, discussions at Java.Net
  - Roadmaps, Architecture Documents
Timeline of Project GlassFish

- Tomcat
- Jasper
- Catalina
- JSTL
- Struts
- Crimson
- XSLTC
- Xalan
- Xerces
- JAXB
- JAX-RPC
- JSF

Timeline:
- J1'04 June 2004
- J1'05 June 2005
- J1'06 May 2006
- Apr 2007 tentative

GlassFish Launch
- V1 final
- V1UR1
- V2 (plan)
Project GlassFish

- GlassFish v1
  - Released! – Victory! Java EE 5 Compliance!
  - UR1 - bug fixes

- GlassFish v2
  - New WS stack, performance, startup time
  - Load balancing, cluster management, failover
  - Some scripting support (see phobos.dev.java.net)

- GlassFish v3
  - Larger architectural changes
  - Better modularization, better scripting support
Enterprise Ready

- clustering based on JXTA
  - see http://shoal.dev.java.net
- HTTP as well as RMI-IIOP Load Balancing and Failover
- In-Memory Replication for Session Persistence
- Sun's distribution (i.e.) SJSAS includes HADB which provides 99.999% availability
- Self management feature
- Generic JMS RA
More than a great AppServer

● Tools, including an Eclipse Plug-In
● Samples, Documentation, How-To
● SOA - BPEL Engine, JBI Integration
● A DataBase - Derby
● Pragmatic Approach
  ● Improved Startup time
  ● No Security Manager by Default
● Focus on Popular Frameworks and Applications
  ● Running out of the box
GlassFish & SOA

- BPEL Engine
- JBI runtime
- Java EE service engine

- Yes, you can mark an EJB as a JBI component
GlassFish Wider Impact

• Encouraging Java EE 5 Adoption
• Enabling Java EE 5 Adoption
  • Many Groups Using GF Components
• Raising the Bar for FOSS AS
  • No more “It is an Open Source” excuses!
• Leading the way for more FOSS Middleware
  • Portal
  • OpenDS
  • OpenESB
  • OpenSSO
Global Community
We need your participation

• As Users
  • What's Working? What is Missing?
• As Contributors
  • Bug Reports
  • Documents, Localization
  • Evangelism! Talk to your friends
  • Contribute Bug Fixes and Code!
• Top Priority!
  • Use GF V2 Milestones
How can you build GlassFish?

- Get a java.net user id
- Tools required:
  - maven 1.0.2, JDK 1.5.0, cvs client
- Steps:
  - cvs co glassfish/bootstrap
  - Edit a couple of variables in ~/build.properties
  - maven bootstrap-all build configure-runtime
  - Hey, you have built and installed your own version of GlassFish!
Take a Short Survey to Improve WS Performance in GlassFish

Sameer, one of the lead Web Services Performance guys in our group, sent me this:

Performance is an critical aspect of Web Services. To help us continually improve our Web Service API and toolkit implementations at Sun, please provide your valuable input and developer feedback through this short ten question survey.

Tags: WebServices, Performance, GlassFish

Version Information is required for Persistence in GlassFish b41

Sahool reports that, because of a recent change in the Java Persistence API spec, every persistence.xml file must have a version attribute in the root element, as in:

```xml
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="1.0.0">
  <persistence-unit name="PU1"/>
</persistence>
```

The change should be effective in this week's b41. Check more details in Sahool's Post.

Tags: Versions, GlassFish, JavaPersistence

ADF Faces, MyFaces subproject

Recently Eduardo blogged about Oracle's donation of ADF faces to Apache. Last week Jonas reported that the ADF project enters the Apache incubator which means that developers can start making contributions to the component library. And that Oracle
Project Tango

- JAX-WS 2.0 delivers WS-I basic interop
  - adequate for basic scenarios
  - with simple security model
- Project Tango adds richer protocol support
  - With richer security models, better quality-of-service, etc.
  - WS-Security, WS-Trust, WS-ReliableMessaging, ...
  - Without changing the JAX-WS APIs.
    - your existing apps will benefit without changes!
- Sun Java™ team is working closely with Microsoft
- http://wsit.dev.java.net
NetBeans Enterprise Pack 5.5

- Includes NB 5.5
  - Supports Java EE 5 Applications
  - Java Persistence in Web, EJB and Standalone
  - Includes many wizards
- With SJS AS 9.0 (Sun's distro of GlassFish)
- Plus
  - XML Schema Tools
  - Web Services Orchestration
  - UML Modelling

http://www.netbeans.org
NetBeans & GlassFish Demo
Java EE Futures

- Still too early to say anything definitive
- Everything is subject to approval by the JCP
- We need feedback from you!
  - What's most important to improve
  - What's most important to add
  - What is still not easy enough
  - What did we get right!
Java EE Futures

- JBI, SCA
  - GlassFish already has JBI runtime
- WS Interoperability
- Portlets
- High availability, clustering, etc.
- Scripting
  - JavaScript in JSP pages, servlets
  - AJAX
- Improve existing APIs
Web Tier Futures: AJAX

- We expect two long-term styles with Ajax
- High-level JSF/Ajax components
  - using JavaScript behind the scenes
  - for both whizzy effects and powerful server interaction
- Raw Ajax: hand written client JavaScript
  - increasingly using toolkits: Dojo,Kabuki,Prototype, ...
- Java™ Web Tier will support both styles
Conclusion

atomy EE 5 is here** now**
  – It is a no-brainer for new projects
  – Get it now!

• Download the SDK
  • [http://java.sun.com/javaee](http://java.sun.com/javaee)

• Get involved in the GlassFish community
  • [http://glassfish.dev.java.net](http://glassfish.dev.java.net)

• Please give us feedback
  • [http://forum.java.sun.com](http://forum.java.sun.com)
Credits for the slides

- Bill Shannon (JavaONE 2006 Technical Key Note presentation)
- Eduardo Pelegri-Llopard (GlassFish presentation)
- Sahoo
Thank you,
Sahoo
Sanjeeb.Sahoo@Sun.COM
http://weblogs.java.net/blog/ss141213/
Q & A