ApacheCon

Apache OpenEJB

In-Depth

David Blevins (dblevins@apache.org)

Leading the Wave
of Open Source

The Basics - History

- Timeline
  - 1999 - Founded in Exoffice - EJB 1.1 level
  - 2001 - Integrated in Apple's WebObjects
  - 2002 - Moved to SourceForge
  - 2003 - Integrated in Apache Geronimo
  - 2004 - Moved to Codehaus
  - 2006 - Moved to Apache Incubator
  - 2007 - Graduated Apache OpenEJB
- Specification involvement
  - EJB 2.1 (Monson-Haefel)
  - EJB 3.0 (Blevins)
  - EJB 3.1 (Blevins)

EJB Vision & Philosophy

- EJB can be light
  - EJB as a concept is not heavy, implementations were heavy
- EJB can be simpler
  - Though the API was cumbersome it could be improved
- EJB can be used for plain applications
  - The portability concept can be flipped on end
  - The flexability applications get also provides great flexability to the container to do things differently yet not break compliance

Project Philosophy

- Be as invisible as possible
- Do not create work for users
- Do not burden users unless there is no other way
- Avoid complexity -- KISS
- Do not over abstract or prematurely abstract
- Bend to users -- don’t make users bend to you
- Give ‘em what they want -- not just what was asked for
- Make simple things easy and hard things possible
- Default to the little guy

Doing the Opposite

- Instead of putting the Application into a Container...
  - Put the Container into the Application
- Instead of embedding Tomcat into OpenEJB...
  - Embed OpenEJB into Tomcat
- Instead of putting WARs in EARs to join with EJBs...
  - Put EJBs into WARs

Core features since 1.0

- Embedded EJB Container
  - Embed the container in a plain Java SE vm.
  - Focus on testing and desktop applications, etc.
- Tomcat Integration
  - Embed the Container in Tomcat
  - Focused on giving Tomcat users a little extra (transactions, ejbs, etc.)
- Collapsed EAR
  - Put EJB jars and EJB classes right in WEB-INF/lib or WEB-INF/classes
  - Completely removes packaging and classloading complexity

ApacheCon

Are we crazy?

Leading the Wave
of Open Source

Standardized in EJB 3.1 and Java EE 6

- Embedded EJB Container API
  - Embed the container in a plain Java SE vm.
  - Focus on testing and desktop applications, etc.
- Web Profile
  - Servlets/JSF + EJB Lite + Transactions
  - Focused on giving “web” users a little extra
- EJBs in .war files
  - Put EJB jars and EJB classes right in WEB-INF/lib or WEB-INF/classes
  - Completely removes packaging and classloading complexity

ApacheCon

phew!

Leading the Wave
of Open Source

More new EJB 3.1 features

- @LocalBean view
  - No more interfaces. The last non-POJO requirement, now gone.
- @Singleton beans
  - Almost identical @Stateless bean with pool size of exactly 1
  - @Startup ability identical to Servlet <load-on-startup>
- @Asynchronous bean methods
  - Queue up work in other threads -- finally
  - Can replace some trivial JMS/MDB usage
- @Schedule based timers
  - Based on Quartz/Cron
- A little bit less vendor config
  - @StatefulTimeout
  - @AccessTimeout

EJB/Java EE mixed features

- @ManagedBean
  - The Cheese Pizza bean type
  - Like a @Stateful bean with oddities removed
  - Created on lookup or injection into other components
  - Supports Interceptors and Dependency injection
- Global JNDI
  - java:module
  - java:app
  - java:global
- Standard EJB JNDI names
  - java:module/MyBean
  - java:app/myModule/MyBean
  - java:global/myApp/myModule/MyBean

EJB.next and Java EE.next

- Promote @ManagedBean to a Session bean
- Break up EJB -- separate the toppings
  - @TransactionManagement
  - @ConcurrencyManagement
  - @Schedule
  - @RolesAllowed
  - @Asynchronous
- Allow all annotations to be used as meta-annotations
- Modernize the Connector/MDB relationship
- Interceptor improvements
- Balance API
  - Everything that can be turned on should be able to shut off
- Improve @ApplicationException

Interceptor -- Today

```
    @InterceptorBinding
@Target(value = {ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface Log {
}

@Log
public class FooBean {

    public void somethingCommon(){
      //...

    public void somethingImportant() {
      //...

    public void somethingNoteworthy() {
      //...
}

@Log
public class LoggingInterceptor {

    private java.util.logging.Logger logger =
            java.util.logging.Logger.getLogger("theLogger");

    @AroundInvoke
    public Object intercept(InvocationContext context) throws Exception {
        logger.info("" + context.getMethod().getName());
        return context.proceed();
    }
}
```

Interceptor Improvements

```
    @Log
public class FooBean {

    public void somethingCommon(){
        //...
    }

    @Info
    public void somethingImportant() {
        //...
    }

    @Fine
    public void somethingNoteworthy() {
        //...
    }
}
```

Interceptor Improvements

```
    @Log
public class LoggingInterceptor {

    private java.util.logging.Logger logger =
            java.util.logging.Logger.getLogger("theLogger");

    @AroundInvoke
    public Object finest(InvocationContext context) throws Exception {
        logger.finest("" + context.getMethod().getName());
        return context.proceed();
    }

    @Info
    public Object info(InvocationContext context) throws Exception {
        logger.info("" + context.getMethod().getName());
        return context.proceed();
    }

    @Fine
    public Object fine(InvocationContext context) throws Exception {
        logger.fine("" + context.getMethod().getName());
        return context.proceed();
    }
}
```

Meta-Annotations

```
    @RolesAllowed({"SuperUser", "AccountAdmin", "SystemAdmin"})
@Stereotype
@Target(METHOD)
@Retention(RUNTIME)
public interface Admins {}


@Schedule(second="0", minute="0", hour="0", month="*", dayOfWeek="*", year="*")
@Stereotype
@Target(METHOD)
@Retention(RUNTIME)
public @interface Hourly {}


@Schedule(second="0", minute="0", hour="0", month="*", dayOfMonth="15,Last", year="*")
@Stereotype
@Target(METHOD)
@Retention(RUNTIME)
public @interface BiMonthly {}

@Singleton
@TransactionManagement(CONTAINER)
@TransactionAttribute(REQUIRED)
@ConcurrencyManagement(CONTAINER)
@Lock(READ)
@Interceptors({LoggingInterceptor.class, StatisticsInterceptor.class})
@Stereotype
@Target(TYPE)
@Retention(RUNTIME)
public @interface SuperBean {}
```

Meta-Annotations

```
    @Singleton
@TransactionManagement(CONTAINER)
@TransactionAttribute(REQUIRED)
@ConcurrencyManagement(CONTAINER)
@Lock(READ)
@Interceptors({LoggingInterceptor.class, StatisticsInterceptor.class})
public class MyBean {

    @Schedule(second="0", minute="0", hour="0", month="*", dayOfWeek="*", year="*")
    public void runBatchJob() {
        //...
    }

    @Schedule(second="0", minute="0", hour="0", month="*", dayOfMonth="15,Last", year="*")
    public void sendPaychecks() {
        //...
    }

    @RolesAllowed({"SuperUser", "AccountAdmin", "SystemAdmin"})
    public void deleteAccount(String accountId) {
        //...
    }
}
```

Meta-Annotations

```
@SuperBean
public class MyBean {

    @Hourly
    public void runBatchJob() {
        //...
    }

    @BiMonthly
    public void sendPaychecks() {
        //...
    }

    @Admin
    public void deleteAccount(String accountId) {
        //...
    }
}
```

Testing

Embeded / Testing Principles

- Be as invisible as possible
- No special classloaders required
- No files
  - All Configuration can be done in the test or via properties
  - No logging files
  - No database files (in memory db)
- No ports
  - Remote EJB calls done with “intra-vm” server
  - JMS done via embedded broker with local transport
  - Database connections via embedded database
- No JavaAgent
  - Avoidable if not using JPA or if using Hibernate as the provider
  - OpenJPA to a limited extent

What can you test?

- EJBs
  - @Stateless
  - @Stateful
  - @Singleton
  - @MessageDriven
  - @ManagedBean
  - Interceptors
  - Legacy EJB 2.x and earlier
- Views
  - @Remote
  - @Local
  - @LocalBean
  - @WebService (requires a port)

What can you test? (cont.)

- Container Provided resources
  - DataSources
  - EntityManagers and EntityManagerFactories
  - JMS Topics/Queues
  - WebServiceRefs
  - Any Java EE Connector provided object
- Services
  - Timers
  - Transactions
  - Security
  - Asynchronous methods

What can’t you test?

- Servlets
- Filters
- Listeners
- JSPs
- JSF Managed Beans
- Non-EJB WebServices

Unique Testing Features

- Most spec complete embedded container
- Fast startup (1 - 2 seconds)
- Test case injection
- Overriding
  - Configuration overriding
  - Persistence Unit overriding
  - Logging overriding
- Test centric-descriptors
  - test-specific ejb-jar.xml or persistence.xml, etc.
- Validation
  - Compiler-style output of application compliance issues
  - Avoid multiple “fix, recompile, redeploy, fail, repeat" cycles
- Descriptor output -- great for xml overriding

Demo

Tomcat Integration

- Supports Tomcat 5.5.x and 6.x (7.x in progress)
- Drop in war file or bundle (currently nicknamed Tomtom)
- Great for even when EJBs are not needed
- Servlet enhancements
  - Transaction
  - JPA
  - JMS
  - JAX-WS
  - Connector
  - EJB
  - JSF (trunk)
  - CDI (trunk)

Tomcat Integration (cont.)

- Deployment formats (drop into webapps/ dir)
  - WAR (can include EJBs in wars, aka “Collapsed EAR”)
  - EAR
  - Plain EJB jar
- Remote EJB clients can talk over
  - HTTP
  - HTTPS
- Uses Tomcat’s security for both Servlets and EJBs
- Plans for Web Profile certification

Demo

Questions?

Apache

CON

thank you!

openejb.apache.org

Leading the Wave
of Open Source