



Easy Enterprise Apps Starring:  
AspectJ, Hibernate, JBoss, XDoclet, Ant

Rod Cope

# Rod Cope

- **Founder and CTO of OpenLogic**
  - rod.cope@openlogic.com
- **9+ years Java experience**
  - Sun Certified Java Architect
  - J2SE, J2EE, J2ME, Swing, Security, Open Source, etc.
  - GE, IBM, Ericsson, Manugistics, Digital Thoughts, etc.
- **JBoss since EJBoss**
- **Hibernate and AspectJ – 2+ years**

# Goals

- `// Outside of J2EE: No tables, exceptions, tx, or Hibernate`  
`address = new Address( "123 Elm", "Denver", "CO", "80132" );`  
`cust = new Customer( "jd@some.org", "John", "Doe", address );`  
`cust.persist();`
- `// SessionBean: No trace of tables, exceptions, tx, etc.`  
`public List findByLastName( String lastName )`  
`{`  
 `Query query = getNamedQuery( "find-by-last-name" );`  
 `query.setString( "lastName", lastName );`  
  
 `return query.list();`  
`}`

# Overview

- **AspectJ**
  - Wizard of Oz
- **Hibernate**
  - Easy yet powerful object-relational mapping tool
- **JBoss**
  - Flexible application server
- **XDoclet**
  - Code and configuration generator
- **Ant**
  - Coordination master

# Hibernate

- What
  - Object-relationship mapping tool
- Why
  - Fast, stable, flexible, database agnostic

# Customer.java

```
/**
 * Represent a customer with a last name and an address.
 * @hibernate.class table="customer"
 *                 proxy="com.openlogic.Customer"
 *
 * @hibernate.query name="find-by-last-name"
 *                 query="select customer
 *                       from Customer as customer
 *                       where customer.lastName = :lastName
 *                       order by customer.id"
 */
public class Customer
    implements Serializable
{
    private String lastName;
    private Address address;
    ...
}
```

# Customer.java (getters/setters)

```
/** @hibernate.property column="last_name"
 */
public String getLastName()
{
    return this.lastName;
}
public void setLastName( String lastName )
{
    this.lastName = lastName;
}

/** @hibernate.component
 */
public Address getAddress()
{
    return this.address;
}
public void setAddress( Address address )
{
    this.address = address;
}
...
```

# Hibernate Sample

```
// flag to let us know if we caught an exception
boolean throwing = false;

Session session = null;
Transaction tx = null;
try
{
    Configuration cfg = new Configuration().
        addClass( Address.class ).
        addClass( Customer.class );
    SessionFactory factory = cfg.buildSessionFactory();
    session = factory.openSession();

    tx = session.beginTransaction();
    Address address = new Address( "123 Elm", "Denver",
        "CO", "80132" );
    Customer customer = new Customer( "Doe", address );
    session.save( customer );
    tx.commit();
}
```

# Hibernate - catch

```
catch ( Exception e )
{
    // we caught an exception
    throwing = true;
    if ( session != null )
    {
        if ( ( tx != null ) &&
            ( !tx.wasRolledBack() ) &&
            ( !tx.wasCommitted() ) )
        {
            try
            {
                tx.rollback();
            }
            catch ( HibernateException e2 )
            {
                if ( e instanceof HibernateException )
                {
                    throw (HibernateException) e;
                }
                else
                {
                    throw new HibernateException( e );
                }
            }
        }
    }
}
```

# Hibernate - finally

- `finally`

```
{
    if ( session != null )
    {
        try
        {
            session.close();
        }
        catch ( HibernateException e3 )
        {
            // don't complain about a failure to close the
            // session if we were already going to throw
            // something more important
            if ( !throwing )
            {
                // rethrow so the caller knows it failed
                throw e3;
            }
        }
    }
}
```

# Hibernate - try/catch/finally

```
// flag to let us know if we caught an exception
boolean throwing = false;
```

```
Session session = null;
Transaction tx = null;
```

```
try
{
    Configuration cfg = new Configuration().addClass( Address.class ).addClass( Customer.class );
    SessionFactory factory = cfg.buildSessionFactory();
    session = factory.openSession();
    tx = session.beginTransaction();
```

```
    Address address = new Address( "123 Elm", "Denver", "CO", "80132" );
    Customer customer = new Customer( "Doe", address );
    session.save( customer );
```

```
    tx.commit();
}
catch ( Exception e )
{
    // we caught an exception
    throwing = true;

    if ( session != null )
    {
        if ( ( tx != null ) &&
            ( !tx.wasRolledBack() ) &&
            ( !tx.wasCommitted() ) )
        {
            try
            {
                tx.rollback();
            }
            catch ( HibernateException e2 )
            {
                if ( e instanceof HibernateException )
                {
                    throw (HibernateException) e;
                }
                else
                {
                    throw new HibernateException( e );
                }
            }
        }
    }
}
finally
{
    if ( session != null )
    {
        try
        {
            session.close();
        }
        catch ( HibernateException e3 )
        {
            // don't complain about a failure to close the session
            // if we were already going to throw something more
            // important
            // important
            if ( !throwing )
            {
                // rethrow so the caller knows it failed
                throw e3;
            }
        }
    }
}
```

# Hibernate Summary

- Hard to read
- Hard to debug and maintain
- Ugly!

# Hibernate with AspectJ

- What
  - AspectJ lets us wrap new code around existing code
- Why
  - To encapsulate all that ugly exception handling
- Same code with help from AspectJ and friends:

```
public void createCustomer( String userId )
    throws HibernateException
{
    Address address = new Address("123 Elm", "Denver", "CO", "80132");
    Customer customer = new Customer( "Doe", address );
    customer.persist();
}
```

# Customer and Persistent

```
public class Customer extends Persistent
    implements Serializable { ... }

public abstract class Persistent implements
    Auditable, Lifecycle, Validatable, Serializable
{
    protected Long id;
    /** @hibernate.id generator-class="native"
     *     unsaved-value="null"
     */
    public Long getId(){ return id; }
    public void setId( Long id ) { this.id = id; }
    ...
    public Long persist()
        throws HibernateException
    {
        HibernateSession.currentSession().saveOrUpdate( this );
        return id;
    }
}
```

# HibernateSession.java (current)

```
public class HibernateSession
{
    public static final ThreadLocal sessionHolder =
        new ThreadLocal();

    public static final ThreadLocal countHolder =
        new ThreadLocal();

    public static Session currentSession()
        throws HibernateException
    {
        Session session = (Session) sessionHolder.get();
        return session;
    }

    ...
}
```

# HibernateSession.java (open)

```
public static Session openSession( String userId )
    throws HibernateException {
    try {
        Session session = currentSession();
        if ( session == null ) {
            SessionFactory sf = getSessionFactory();
            if ( sf == null ) {
                sf = (SessionFactory) new InitialContext().lookup( factoryJNDI );
            }
            if ( userId != null ) {
                session = sf.openSession( new AuditInterceptor( userId ) );
            } else {
                session = sf.openSession();
            }
            sessionHolder.set( session );
            countHolder.set( ONE );
        } else {
            int count = ( (Integer) countHolder.get() ).intValue() + 1;
            countHolder.set( new Integer( count ) );
        }
        return session;
    }
    catch ( NamingException e ) {
        throw new HibernateException( "Failed to lookup SessionFactory for userId=" +
            userId, e );
    }
}
```

# HibernateSession.java (close)

```
public static void closeSession() throws HibernateException {
    Integer count = (Integer) countHolder.get();
    if ( count == null ) {
        return;
    }
    if ( count == ONE ) {
        countHolder.set( null );
        Session session = currentSession();
        sessionHolder.set( null );
        if ( session != null ) {
            session.close();
        }
    } else {
        int ic = count.intValue() - 1;
        if ( ic == 1 ) {
            countHolder.set( ONE );
        } else {
            countHolder.set( new Integer( ic ) );
        }
    }
}
```

# AuditInterceptor

```
public class AuditInterceptor implements Interceptor, Serializable {
    private String user;
    public AuditInterceptor( String user ) {
        this.user = user;
    }
    public boolean onFlushDirty( Object entity, Serializable id,
                                Object[] currentState,
                                Object[] previousState,
                                String[] propertyNames, Type[] types) {
        if ( entity instanceof Auditable ) {
            AuditInfo ai = ( (Auditable) entity).getAuditInfo();
            try {
                Timestamp timestamp = new Timestamp(System.currentTimeMillis());
                ai.setLastUpdated( timestamp );
            }
            catch( Exception e ) { }
            ai.setUpdatedBy( user );
        }
        return false;
    }
    ...
}
```

# Persistent and Auditing

```
public abstract class Persistent implements
    Auditable, Lifecycle, Validatable, Serializable
{
    ...
    protected AuditInfo auditInfo = new AuditInfo();

    /** @hibernate.component
     */
    public AuditInfo getAuditInfo()
    {
        return this.auditInfo;
    }
    public void setAuditInfo( AuditInfo auditInfo )
    {
        this.auditInfo = auditInfo;
    }
}
```

# AuditInfo

```
public final class AuditInfo implements Serializable
{
    private Timestamp lastUpdated;
    private Timestamp created;
    private String updatedBy;
    private String createdBy;

    /** @hibernate.property
     */
    public Timestamp getLastUpdated()
    {
        return lastUpdated;
    }
    public void setLastUpdated(Timestamp lastUpdated)
    {
        this.lastUpdated = lastUpdated;
    }
    ...
}
```

# HibernateAspect.java

```
public aspect HibernateAspect
{
    pointcut newCustomer( String userId ) :
    call(void createCustomer( String ) throws HibernateException)
        && args( userId );

    void around( String userId ) throws HibernateException :
        newCustomer( userId )
    {
        // see 75 lines of try/catch/finally for full details...
        session = HibernateSession.openSession( userId );
        tx = session.beginTransaction();
        proceed( userId );
        tx.commit();
    }
    ...
}
```

# Hibernate with AspectJ

```
public void createCustomer( String userId )
    throws HibernateException
{
    Address address = new Address("123 Elm", "Denver", "CO", "80132");
    Customer customer = new Customer( "Doe", address );
    customer.persist();
}
```

# Hibernate + AspectJ in JBoss

- What
  - Use Hibernate for BMP inside JBoss
- Why
  - CMP sucks
  - No subqueries, sorts, outer joins, detach/reattach objects, etc.
- How
  - Write SessionBean
  - Enhance HibernateAspect.java
  - Write J2EE test client with DbUnit
  - Generate Hibernate mapping files
  - Configure Hibernate service to work in JBoss
  - Lots of Ant!

# SessionBean - AspectJ/Hibernate

```
/** @ejb.bean name          = "CustomerHelper"
 *          jndi-name      = "ejb/CustomerHelper"
 *          type           = "Stateless"
 * @ejb.transaction-type type = "Container"
 * @ejb.security-role-ref role-name = "admin"
 *          role-link      = "admin"
 */
public class CustomerHelperBean implements SessionBean {
    /** Find customers with the given last name.
     * @return A List of Customer objects that may be empty
     * @ejb.interface-method view-type="both"
     * @ejb.transaction type="Required"
     */
    public List findByLastName( String lastName ) {
        Query query = getNamedQuery( "find-by-last-name" );
        query.setString( "lastName", lastName );
        return query.list();
    }
    public void ejbActivate() throws EJBException {}
    public void ejbPassivate() throws EJBException {}
    public void ejbRemove() throws EJBException {}
    public void setSessionContext(SessionContext sc) throws EJBException {}
    public Query getNamedQuery( String queryName ) { return null; }
}
```

# Customer.java

```
/**
 * Represent a customer with a last name and an address.
 * @hibernate.class table="customer"
 *                 proxy="com.openlogic.Customer"
 *
 * @hibernate.query name="find-by-last-name"
 *                 query="select customer
 *                       from Customer as customer
 *                       where customer.lastName = :lastName
 *                       order by customer.id"
 */
public class Customer
    implements Serializable
{
    private String lastName;
    private Address address;
    ...
}
```

# HibernateAspect - Enhanced 1

```
public aspect HibernateAspect {
    interface SessionContextManageable {}

    SessionContext SessionContextManageable.hasc = null;

    public SessionContext SessionContextManageable.
        getSessionContext()
    { return hasc; }

    public void SessionContextManageable.
        saveSessionContext( SessionContext sc )
    { hasc = sc; }

    declare parents : com.openlogic.*Bean implements
        SessionContextManageable;
```

# HibernateAspect - Enhanced 2

```
public aspect HibernateAspect {
    ...
    pointcut setSessionContext( SessionContextManageable scm,
                                SessionContext sc ) :
        target( scm ) &&
        execution( void setSessionContext( SessionContext ) ) &&
        args( sc );

    after( SessionContextManageable scm, SessionContext sc ) :
        setSessionContext( scm, sc )
    {
        scm.saveSessionContext( sc );
    }

    pointcut ejbRemove(SessionContextManageable scm ) :
        target( scm ) &&
        execution( void ejbRemove() );

    after( SessionContextManageable scm ) : ejbRemove( scm )
    {
        scm.saveSessionContext( null );
    }
}
```

# HibernateAspect - Enhanced 3

```
public aspect HibernateAspect {
    ...
    pointcut getNamedQuery( SessionContextManageable scm
                           String name ) :
        target( scm ) &&
        execution( Query getNamedQuery( String ) ) &&
        args( name );

    Query around( SessionContextManageable scm, String name ) :
        getNamedQuery( scm, name )
    {
        return HibernateSession.currentSession().getNamedQuery( name );
    }
}
```

# HibernateAspect - Enhanced 4

```
public aspect HibernateAspect {  
    ...  
    pointcut nonVoidEJBMethod( SessionContextManageable scm ) :  
        target( scm ) &&  
        execution( !void *(..) ) &&  
        !execution( !void ejb*(..) ) &&  
        !execution( SessionContext getSessionContext() );  
}
```

# HibernateAspect - Enhanced 5

```
Object around( SessionContextManageable scm ) : nonVoidEJBMethod( scm )
{
    String method = thisJoinPointStaticPart.getSignature().getName();
    String principal = "unknown";
    if ( ( scm.getSessionContext() != null ) &&
        ( scm.getSessionContext().getCallerPrincipal() != null ) ) {
        principal = scm.getSessionContext().getCallerPrincipal().toString();
    }

    boolean throwing = false;
    Object result = null;
    Session session = null;
    try {
        session = HibernateSession.openSession( principal );
        result = proceed( scm );
        session.flush();
    } catch ( Exception e ) {
        throwing = true;
        try { handleException( session, scm, e, method, false ); }
        catch ( HibernateException he ) { // can't happen }
    }
    finally { handleFinally( session, method, throwing ); }
    return result;
}
```

# HibernateAspect - Enhanced 6

```
void handleException( Session session,
                    SessionContextManageable scm,
                    Exception e, String method,
                    boolean canThrowHibernateException )
    throws HibernateException
{
    if ( session != null ) {
        try { scm.getSessionContext().setRollbackOnly(); }
        catch ( Exception e2 ) { }
    }

    if ( e instanceof EJBException ) {
        throw (EJBException) e;
    } else if ( ( canThrowHibernateException ) &&
                ( e instanceof HibernateException ) ) {
        throw (HibernateException) e;
    } else if ( ( canThrowHibernateException ) &&
                ( e instanceof SQLException ) ) {
        throw new HibernateException( e );
    } else {
        throw new EJBException( e );
    }
}
```

# HibernateAspect - Enhanced 7

```
void handleFinally( Session session, String method,
                   boolean throwing )
{
    if ( session != null )
    {
        try
        {
            HibernateSession.closeSession();
        }
        catch ( Exception e )
        {
            if ( !throwing )
            {
                throw new EJBException( e );
            }
        }
    }
}
```

# SessionBean - Recap

```
/** @ejb.bean name          = "CustomerHelper"
 *      jndi-name          = "ejb/CustomerHelper"
 *      type               = "Stateless"
 * @ejb.transaction-type type = "Container"
 * @ejb.security-role-ref role-name = "admin"
 *      role-link          = "admin"
 */
public class CustomerHelperBean implements SessionBean {
    /** Find customers with the given last name.
     * @return A List of Customer objects that may be empty
     * @ejb.interface-method view-type="both"
     * @ejb.transaction type="Required"
     */
    public List findByLastName( String lastName ) {
        Query query = getNamedQuery( "find-by-last-name" );
        query.setString( "lastName", lastName );
        return query.list();
    }
    public void ejbActivate() throws EJBException {}
    public void ejbPassivate() throws EJBException {}
    public void ejbRemove() throws EJBException {}
    public void setSessionContext(SessionContext sc) throws EJBException {}
    public Query getNamedQuery( String queryName ) { return null; }
}
```

# Test Client - DbUnit

```
public class TestCustomer extends DatabaseTestCase {
    static String databaseDriver      = null;
    static String databaseURL         = null;
    static String databaseUserID      = null;
    static String databasePassword    = null;

    static {
        databaseDriver      = System.getProperty( "driver_class" );
        databaseURL         = System.getProperty( "url" );
        databaseUserID      = System.getProperty( "username" );
        databasePassword    = System.getProperty( "password" );
    }

    protected IDatabaseConnection getConnection() throws Exception {
        Class driverClass = Class.forName( databaseDriver );
        Connection jdbcConnection = DriverManager.getConnection(
            databaseURL, databaseUserID, databasePassword );
        return new DatabaseConnection( jdbcConnection );
    }

    protected IDataset getDataSet() throws Exception {
        return new FlatXmlDataSet( new InputStreamReader(
            getClass().getClassLoader().getResourceAsStream( "data.xml" ) ) );
    }
}
```

# Test Client - Simple Test

```
public void testCustomersAddress()  
{  
    List customers = CustomerHelperUtil.getHome().  
        create().findByLastName( "Doe" );  
    assertEquals( "Wrong number of customers",  
        1, customers.size() );  
  
    Customer cust = (Customer) customers.get( 0 );  
    assertEquals( "Wrong city for customer",  
        "Denver", cust.getAddress().getCity() );  
}
```

# Test Client - Auditing

```
public void testAuditing() { auditChange(); checkAuditChange(); }

public void auditChange() {
    UsernamePasswordHandler handler = new UsernamePasswordHandler(
        "adminuser", "password" );
    LoginContext lc = new LoginContext( "admin", handler );
    lc.login();

    InitialContext initialContext = new InitialContext();
    Object homeRef = initialContext.lookup( ProductHelperHome.JNDI_NAME );
    ProductHelperHome home = (ProductHelperHome)
        PortableRemoteObject.narrow( homeRef, ProductHelperHome.class );
    ProductHelper helper = home.create();

    List products = helper.findByName( "Great Stuff 1.0" );
    assertEquals( "Wrong number of products", 1, products.size() );

    Product product = (Product) products.get( 0 );
    product.setPrice( new BigDecimal( "15.95" ) );
    helper.persist( product );

    initialContext.close();
    lc.logout();
}
```

# Test Client - Auditing Test

```
public void checkAuditChange()  
{  
    List products = ProductHelperUtil.getHome().create().  
        findByName( "Great Stuff 1.0" );  
    assertEquals( "Wrong number of products",  
        1, products.size() );  
  
    Product product = (Product) products.get( 0 );  
    assertEquals( "Updated by wrong userId",  
        "adminuser",  
        product.getAuditInfo().getUpdatedBy() );  
}
```

# Ant

- Your friend and mine
- Use it and love it
- 1,240 lines of Ant in this project

# Ant: Compile

```
<taskdef
  resource="org/aspectj/tools/ant/taskdefs/aspectjTaskdefs.properties">
  <classpath refid="lib.class.path"/>
</taskdef>

<!-- Run AspectJ compiler against all non-test source -->
<iajcc srcdir="${basedir}"
  includes="src/java/**,src/ejb/**,target/gen-src/**"
  destdir="${build.java.classes.dir}"
  target="1.2"
  classpathref="compile.class.path"
  verbose="false"/>

<!-- Compile test source separately - don't want it aspected -->
<javac srcdir="${test.src.dir}"
  destdir="${build.test.classes.dir}"
  classpathref="compile.class.path"
  debug="on"
  deprecation="on"/>
```

# Ant: Generate

```
<taskdef name="hibernatedoclet"  
    classname="xdoclet.modules.hibernate.HibernateDocletTask"  
    classpathref="gen.class.path"/>  
<hibernatedoclet destdir="${build.gen-hibernate-map.dir}"  
    mergedir="${xdoclet.src.dir}"  
    force="true" verbose="false">  
    <fileset dir="${java.src.dir}">  
        <include name="**/*.java"/>  
        <exclude name="**/*Aspect.java"/>  
    </fileset>  
  
    <hibernate version="2.0"/>  
  
    <jbossservice destdir="${build.jboss-service.dir}"  
        jndiname="java:/HibernateFactory"  
        servicename="Hibernate"  
        dialect="net.sf.hibernate.dialect.MySQLDialect"  
        datasource="java:/DefaultDS" showSql="false"/>  
</hibernatedoclet>
```

# Ant: Patch

```
<!-- Add this to the generated jboss-service.xml
<loader-repository>
  com.openlogic:loader=sample.ear
</loader-repository>  -->

<replaceregexp
  file="${build.jboss-service.dir}/jboss-service.xml"
  match="&gt;([^\&lt;]*)&lt;mbean"
  replace="&gt;\1&lt;loader-
  repository&gt;com.openlogic:loader=sample.ear&lt;/loader-
  repository&gt;\1&lt;mbean"
  flags="s"/>
```

Contents of jboss-app.xml:

```
<jboss-app>
  <loader-repository>
    com.openlogic:loader=sample.ear
  </loader-repository>
</jboss-app>
```

# Ant: Schema Export

```
<fileset id="mapping.files" dir="${build.gen-hibernate-map.dir}">
  <include name="**/*.hbm.xml" />
</fileset>
<pathconvert refid="mapping.files" property="hibernate.mappings"
  pathsep=" "/>
<java classname="net.sf.hibernate.tool.hbm2ddl.SchemaExport"
  fork="true">
  <arg line="--text"/>
  <arg line="--delimiter=";" />
  <arg line="--format"/>
  <arg line="--output=${build.gen-schema.file}"/>
  <arg line="${hibernate.mappings}"/>
  <classpath refid="schema.class.path" />
  <sysproperty key="hibernate.dialect"
    value="net.sf.hibernate.dialect.MySQLDialect"/>
  <sysproperty key="hibernate.connection.driver_class"
    value="${database.driver}"/>
  <sysproperty key="hibernate.connection.url"
    value="${database.url}"/>
  <sysproperty key="hibernate.connection.username"
    value="${database.userid}"/>
  <sysproperty key="hibernate.connection.password"
    value="${database.password}"/>
  <sysproperty key="hibernate.show_sql"
    value="false"/>
</java>
```

# Ant: DbUnit

```
<taskdef name="dbunit"  
    classname="org.dbunit.ant.DbUnitTask"  
    classpathref="dbunit.class.path" />  
  
<mkdir dir="${build.jar.dir}" />  
<dbunit driver="${database.driver}"  
    url="${database.url}"  
    userid="${database.userid}"  
    password="${database.password}"  
    classpathref="lib.class.path">  
    <export dest="${dbunit.data.xml.file}">  
        <table name="customer" />  
        <table name="user_role" />  
        <table name="product" />  
        <table name="purchase" />  
        <table name="line_item" />  
    </export>  
</dbunit>
```

# Ant: JUnit Testing

```
<junit printsummary="on" haltonfailure="off">
  <jvmarg value="-Djava.security.manager"/>
  <sysproperty key="java.security.policy"
    value="\${config.src.dir}/app.policy"/>
  <sysproperty key="java.security.auth.login.config"
    value="\${config.src.dir}/auth.conf"/>

  <sysproperty key="hibernate.dialect"
    value="net.sf.hibernate.dialect.MySQLDialect"/>
  <sysproperty key="hibernate.connection.driver_class"
    value="\${database.driver}"/>
  <sysproperty key="hibernate.connection.url"
    value="\${database.url}"/>
  <sysproperty key="hibernate.connection.username"
    value="\${database.userid}"/>
  <sysproperty key="hibernate.connection.password"
    value="\${database.password}"/>
  <sysproperty key="hibernate.show_sql"
    value="false"/>

  <!-- Normal JUnit stuff after this point -->
  ...
</junit>
```

# Security Configuration - App

- Contents of app.policy:

```
grant {  
    permission java.security.AllPermission;  
};
```

- Contents of auth.conf:

```
admin {  
    org.jboss.security.ClientLoginModule required;  
};
```

- Contents of jndi.properties:

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory  
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces  
java.naming.provider.url=jnp://localhost:1099
```

# Security Configuration - JBoss

- Contents of JBoss conf/login-config.xml

```
<application-policy name = "admin">
  <authentication>
    <login-module
code="org.jboss.security.auth.spi.DatabaseServerLoginModule"
flag = "required">
      <module-option name = "dsJndiName">
        java:/DefaultDS
      </module-option>
      <module-option name = "principalsQuery">
        select password from users where user_id=?
      </module-option>
      <module-option name = "rolesQuery">
        select role, 'Roles' from user_role where user_id=?
      </module-option>
    </login-module>
  </authentication>
</application-policy>
```

# Security Configuration - DB

## Example Using MySQL:

```
create table if not exists users
```

```
(  
  user_id          VARCHAR(50)  BINARY  NOT NULL,  
  password         VARCHAR(50)  BINARY  NOT NULL,  
  PRIMARY KEY (user_id)  
) TYPE=InnoDB;
```

```
create table if not exists user_role
```

```
(  
  user_id          VARCHAR(50)  BINARY  NOT NULL,  
  role             VARCHAR(50)  BINARY  NOT NULL,  
  INDEX user_id_index (user_id),  
  INDEX role_index  (role),  
  FOREIGN KEY      (user_id)    REFERENCES users(user_id)  
) TYPE=InnoDB;
```

```
insert into users values ( "adminuser", "password" );
```

```
insert into user_role values ( "adminuser", "admin" );
```

# Trouble in Paradise

- AspectJ FUD
  - Who knows what it's doing?
  - What if it breaks in some weird way?
  - It's not ready for production
  - Patent issues
- What's the alternative?
  - Hand-coding
    - cut/paste/debug hell
    - Changes?
  - Custom code generation
    - Huge convoluted set of code
    - Changes?

# Conclusion

- **Development Time**
  - Less than half with AspectJ
- **Performance**
  - 90+% of non-aspected code
- **Recommendations**
  - Ready for small, non-mission-critical projects
  - Heavy unit testing and load testing
  - Try it! Faster, easier, more fun to write apps with
    - AspectJ, Hibernate, XDoclet, and Ant

# References and Links

- Sample Code
  - <http://www.openlogic.com/presentations>
- AspectJ
  - <http://www.eclipse.org/aspectj/>
- Hibernate
  - <http://www.hibernate.org>
- XDoclet
  - <http://xdoclet.sourceforge.net/xdoclet/>
- Ant
  - <http://ant.apache.org>
- JBoss
  - <http://www.jboss.org>

## Any Questions?