



Hibernate 3.0

 Enterprise ORM


The Professional Open Source™ Company



Introduction

- Object / relational mapping (ORM) technology makes it easy to build object-oriented applications that access relational databases
- Hibernate is the most popular ORM solution *ever*
- The open source nature of Hibernate finally popularized ORM technology in Java, from the ground up, while most of the industry was looking the other way
- Hibernate is now the poster child for the JBoss *Professional Open Source* model
- Hibernate 3.0 brings new functionality that remove barriers to enterprise-level adoption


The Professional Open Source™ Company



Road Map

- History of Hibernate
- Why Hibernate?
- Market position
- Goals of Hibernate 3.0
- Hibernate and EJB 3.0
- New features of Hibernate 3.0
- Technical Q & A
- Conclusion


The Professional Open Source™ Company



History of Hibernate

- *November 2001* – SourceForge project founded by Gavin King
- *July 2002* – Thousands of downloads per month, buzz in the blogosphere
- *June 2003* – Hibernate 2.0 released, the very first truly full-featured open source ORM solution ever
- *August 2003* – ten thousand downloads per month (industry starts to really take notice)
- *October 2003* – Hibernate joins JBoss
- *April 2004* – one million page views per month on <http://hibernate.org>
- *June 2004* – EJB 3.0 first early draft released, heavily inspired by Hibernate
- *March 2005* – Hibernate 3.0 released


The Professional Open Source™ Company



Why Hibernate?

- Reduce application code by up to 30%
 - ✓ ORM code is much more understandable and much more maintainable
- Improve performance
 - ✓ Hibernate implements many sophisticated performance optimizations, such as aggressive caching
- Take full advantage of relational technology
 - ✓ A key differentiator of Hibernate is its focus upon relational data access instead of a zealotted obsession with “transparent persistence” and datastore agnosticism
- Maturity
 - ✓ As the most-used ORM solution, Hibernate is also extremely well tested in a huge range of environments
- Clear migration path to EJB 3.0


The Professional Open Source™ Company



Market position

- Hibernate is the clear market leader, with overwhelming market share in Java and J2EE applications
 - ✓ javaworld.com survey: 30% of developers use Hibernate
 - ✓ 2 million page views per month on <http://hibernate.org>
 - ✓ 1500 downloads per day
 - ✓ Winner of 2004 Jolt Software Development award
- Four published books covering Hibernate, more in the works
- Adoption has been bottom up – Hibernate is now coming to the attention of enterprise-level decision makers in companies like Metlife, CSC, NCR, Amazon, RouteOne, BNP Paribas, Accenture
- Companies like these have successful applications deployed on Hibernate, and now consider establishing Hibernate as a standard at the enterprise level
- Professional Open Source makes this possible: production support, training, indemnity are crucial concerns
- New features of Hibernate 3.0 target the needs of companies deploying Hibernate across the enterprise


The Professional Open Source™ Company



Goals of Hibernate 3.0

- Capitalize on the success of Hibernate 2.1
- Continue to expand the market for ORM technology
- Expand the problem space for which ORM is a suitable solution
- Support and validate the emerging standard for ORM in Java and J2EE – EJB 3.0
- Solve enterprise *operational* needs – including needs of projects deployed in a hosted environment
- Integrate with the JBoss Enterprise Middleware System (JEMS)


The Professional Open Source™ Company



Hibernate and EJB 3.0

- EJB 3.0 EDR2 (February 2005) defines a standard for two types of product
 - ✓ A "full" EJB container – for traditional monolithic application servers such as WebSphere, along with the new generation of microcontainers such as JBoss 5.0
 - ✓ An object persistence engine that may be used outside an EJB container – for products like Hibernate, TopLink, Kodo
- All major J2EE vendors are involved in writing the spec: SUN, Oracle, JBoss, BEA, IBM, along with object persistence players such as Solarmetric
- Hibernate is at the core of the JBoss EJB 3.0 container (now in a preview release)
- JBoss will also provide a standalone implementation of EJB 3.0 persistence – essentially just compliant API as a wrapper for Hibernate
- EJB 3.0 promises to finally bring ORM technology to a truly mass-market, unlike previous failed standards (ODMG, JDO)


The Professional Open Source™ Company



New features of Hibernate 3.0


- Hibernate Filters, an innovative new technique for dealing with historical, regional and permissioned data
- New suite of Eclipse-based tools for rapid development of data models
- Mapping functionality enhancements for handling exotic legacy data models
- Support for mapping relational data to XML
- Runtime monitoring and management via JMX
- Integration with J2EE container-based security
- Ease of use enhancements, targeting less sophisticated users
- Alignment with EJB 3.0

The Professional Open Source™ Company



New features in action...

The Professional Open Source™ Company




Creating data filters

- Filters add the ability to define a restriction, like a where clause
- Filters are powerful and can be used in many scenarios
- For example, users should only see things they are allowed to see:

```
<filter-def name="accessFilter">
  <filter-param name="userLevel" type="int"/>
</filter-def>
```

The Professional Open Source™ Company



Attaching filters to classes

- Defined filters are enabled for particular classes

```
<class name="SalesRegion" ...>
  ...
  <property name="accessLevel"
    type="int"
    column="access_lvl"/>
  ...
  <filter name="accessFilter">
    <![CDATA[ :userLevel >= access_lvl ]]>
  </filter>
</class>
```

- Filter parameters are set at runtime...

The Professional Open Source™ Company

JBoss Filtering in a Session

- First, enable the filter for this Session:

```
Filter f = session.enableFilter("accessFilter");
```

- Then bind arguments to filter parameters:

```
f.setParameter("userLevel", currentUser.getLevel());
```

- All data will be filtered automatically:

```
List filtered = session.createQuery("from SalesRegion");  
SalesRegion region =  
    (SalesRegion)session.get(SalesRegion.class, "1");
```

The Professional Open Source™ Company

JBoss Flexible inheritance mapping

- Table-per-concrete-class:**
You can use <union-subclass> for more efficient fetching (uses a UNION query instead of several SELECTs)
- Table-per-subclass:**
You can now use discriminators with <joined-subclass> mapping
- Table-per-hierarchy:**
You can now use formula-based discriminators with <subclass> mappings
- You can now mix inheritance strategies
this is a side-effect of the new <join> mapping

The Professional Open Source™ Company

JBoss The new <join> element

- The new <join> element is used to map a class to several tables:

```
<class name="Item" table="ITEM">  
  <property name="initialPrice"/>  
  <join table="ITEM_IMAGES">  
    <key column="ITEM_ID" on-delete="cascade"/>  
    <property name="imageName"/>  
    <property name="imageJpegBinary"/>  
    <many-to-one name="uploadedBy" class="User"/>  
  </join>  
</class>
```

- Mostly useful for legacy schemas, but nice for inheritance mappings...

The Professional Open Source™ Company

JBoss Mixing inheritance strategies

- We can now mix <subclass> and <joined-subclass>:

```
<class name="BillingDetails" table="BILLING_DETAILS">  
  <discriminator column="BILLING_TYPE" type="int"/>  
  <subclass name="CreditCard">  
    <property name="cardNumber" column="CC_NUMBER"/>  
  </subclass>  
  <subclass name="BankAccount">  
    <join table="BANK_ACCOUNTS">  
      <key column="BILLING_DETAILS_ID"/>  
      <property name="accountNumber" column="BA_NUMBER"/>  
    </join>  
  </subclass>  
</class>
```

The Professional Open Source™ Company

JBoss Dynamic Models

- A normal mapping with a small difference:

```
<hibernate-mapping>  
  <dynamic-class entity-name="Customer">  
    <id name="id" type="long" column="ID">  
      <generator class="sequence"/>  
    </id>  
    <property name="name" column="C_NAME" type="string"/>  
    <many-to-one name="company" column="COMPANY_ID"/>  
    <bag name="orders">  
      <key column="CUSTOMER_ID"/>  
      <one-to-many class="Order"/>  
    </bag>  
  </dynamic-class>  
</hibernate-mapping>
```

The Professional Open Source™ Company

JBoss Dynamic models in action

- You no longer use classes, but Maps of Maps:

```
Session s = openSession();  
  
// Create a customer  
Map david = new HashMap();  
david.put("name", "David Channon");  
  
// Create a company  
Map foobar = new HashMap();  
foobar.put("name", "Foobar Inc.");  
  
// Link both  
david.put("company", foobar);  
  
// Save both  
s.save("Customer", david);  
s.save("Company", foobar);
```

- This should only be used for prototyping...

The Professional Open Source™ Company

JBoss Hand-written SQL

- You can define your own SQL for everything:

```
<class name="User">
...
<sql-insert>
insert into USER (NAME, ID) values( UPPER(?), ? )
</sql-insert>

<sql-update>
update USER set NAME = UPPER(?) WHERE ID = ?
</sql-update>

<sql-delete>
delete from USER where ID = ?
</sql-delete>
</class>
```

The Professional Open Source™ Company

JBoss Custom SQL for loading

- Loaders refer to named queries:

```
<class name="User">
...
<loader query-ref="loadUser"/>
</class>
```

```
<sql-query name="loadUser">
<return alias="u" class="User" lock-mode="upgrade"/>
select NAME as {u.name},
ID as {u.id}
from USER
where ID =?
for update
</sql-query>
```

The Professional Open Source™ Company

JBoss Events in Hibernate 3.0

- Everything in the core is in events:

```
public class AccessListener extends DefaultLoadEventListener {
    public Object onLoad(LoadEvent event,
        LoadEventListener.LoadType loadType)
        throws HibernateException {
        if ( !MySecurity.isAuthorized( event.getEntityName(),
            event.getEntityId() ) )
            throw MySecurityException("Unauthorized access");
        return super.onLoad(event, loadType);
    }
}
```

- Enabling the new listener in configuration:

```
<session-factory>
...
<listener type="load" class="AccessListener"/>
```

The Professional Open Source™ Company

JBoss Conclusion

- 2005 will bring an accelerated rate of enterprise-level adoption of Hibernate due to the following factors:
 - ✓ Success of existing projects using Hibernate
 - ✓ Continued popularity in the Java developer community
 - ✓ Availability of professional training and support
 - ✓ New features of Hibernate 3.0 addressed directly to the concerns of enterprise-level decision makers
 - ✓ Emergence of a compelling standard in EJB 3.0
 - ✓ Delivery of an integrated, open source middleware stack by JBoss, and continued growth of JBoss market share

The Professional Open Source™ Company