Hibernate

Overview

- Object/relational persistence and query service
- Persists regular Java classes
- Interacts with classes using Java reflection
- Built atop JDBC

Advantages

- Does not force a design on persisted classes
- No interface to implement
- Testing independent of Hibernate is easy
- Refactoring costs much less

O/R Mapping

Initial Setup

- Download Hibernate and Hibernate Extensions
  www.hibernate.org
- Obtain the JDBC driver for your database
- Place all these JARs on your project's class path
- Apache Ant
  ant.apache.org

Initial Setup

- Create the **`hibernate.properties`** file

```
hibernate.dialect=net.sf.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url=jdbc:mysql:///cs491b
hibernate.connection.username=mnelson
hibernate.connection.password=1234
```

Dialects

- DB2
- DB2 AS/400
- DB2 OS390
- PostgreSQL
- MySQL
- Oracle
- Oracle 9/10g
- Sybase
- Sybase Anywhere
- Microsoft SQL Server
- SAP DB
- Informix
- Hypersonic SQL
- Ingres
- Progress
- Mckoi SQL
- Interbase
- Pointbase
- FrontBase
- Firebird

Beginning In The Middle

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
  PUBLIC "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">

<hibernate-mapping>

    <!-- ... -->

</hibernate-mapping>
```

Beginning In The Middle

```
<class name="cs491b.Presentation" table="presentation">

  <id name="id" type="int" column="presentation_id">
    <generator class="native"/>
  </id>

  <property name="subject" type="string" not-null="true"/>

  <property name="date" type="date" not-null="true"/>

  <many-to-one name="student" class="cs491b.Student">

</class>
```

Beginning In The Middle

- Use Schema Export tool to derive relations
- Use Code Generator tool to derive JavaBeans
- Both tools may be invoked:
  - On the command-line
  - Via an Ant Task
  - From within your code

SessionFactory

```
package cs491b;

import net.sf.hibernate.*;
import net.sf.hibernate.cfg.Configuration;

// ...

Configuration config = new Configuration();
config.addClass(Presentation.class);
config.addClass(Student.class);

SessionFactory sessionFactory =
    config.buildSessionFactory();

// ...

sessionFactory.close();
```

Session

```
Session session = sessionFactory.openSession();

try {

  // ...

} finally {
  session.close();
}
```

Session Methods

```
// Find an object with a particular ID
session.load(Presentation.class, 15);

// Find objects with a more complex query
session.find("from cs491b.Presentation as pres " +
    "where pres.date >= ?", myDate, Hibernate.DATE);

// Store new objects
session.save(myPresentation);

// Delete objects from database
session.delete(myPresentation);

// Save changes to objects created or saved by session
session.flush();

// Save changes to other objects
session.update(myPresentation);
```

The Query Builder

```
Query query = session.createQuery(
    "from cs491b.Presentation as pres " +
    "where pres.date >= :date");

// Or move the SQL code to your Hibernate mapping file

query = session.getNamedQuery(
  "cs491b.presentationsOnOrAfter");


query.setDate("date", myDate);

query.list();
```

The Query Builder

```
<query name="cs491b.presentationsOnOrAfter">
  <![CDATA[
    from cs491b.Presentation as pres
    where pres.date >= :date
  ]]>
</query>
```

There's More

- Collections and Associations
  - Lazy loading
  - Ordered collections
- Transactions and Concurrency
- HQL
- Criteria Queries
- Native SQL Queries
- See the Hibernate documentation