Spring-Hibernate Training Program
Session 2

BoTree Consulting

Hibernate Overview

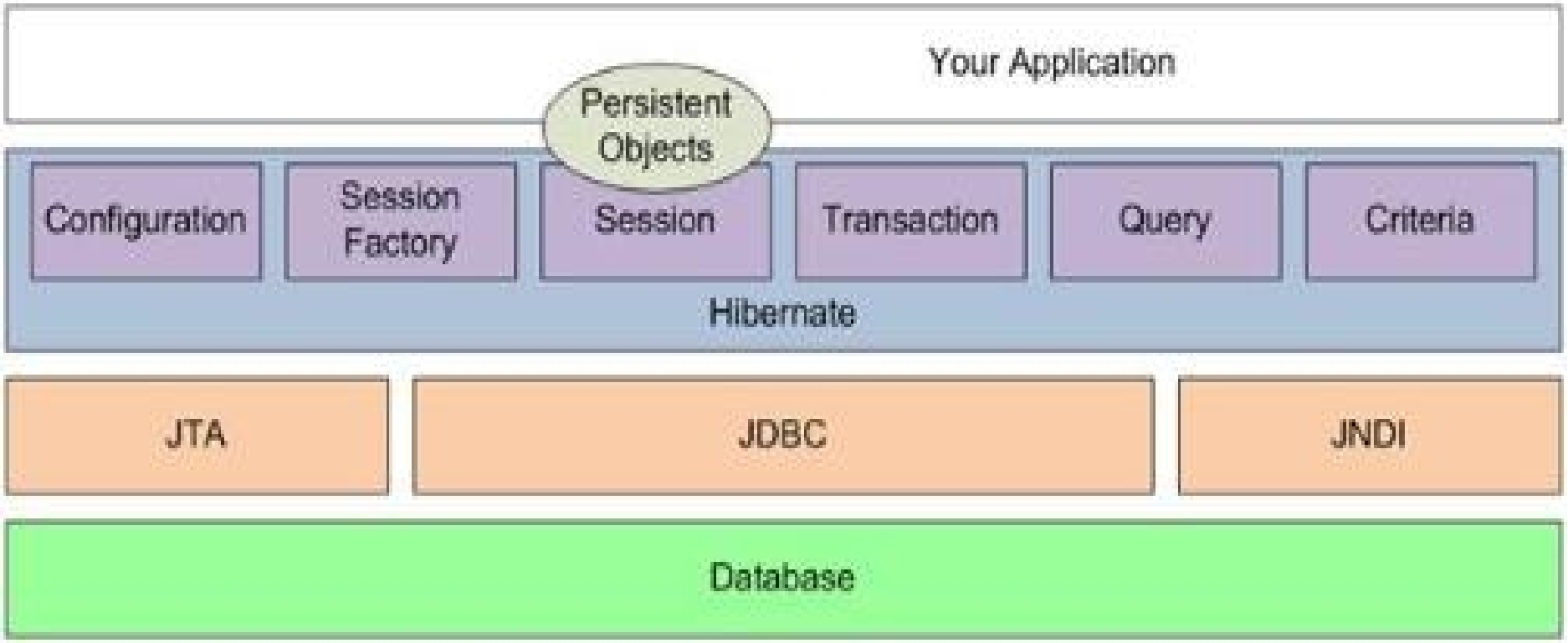
Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an *object-oriented domain model to a traditional relational database*.

Hibernate solves *object-relational impedance mismatch problems* by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate Internals

- Configuration
- Mappings
- HibernateSessionFactory
- HQL
- Criteria API
- Caching

Hibernate Architecture



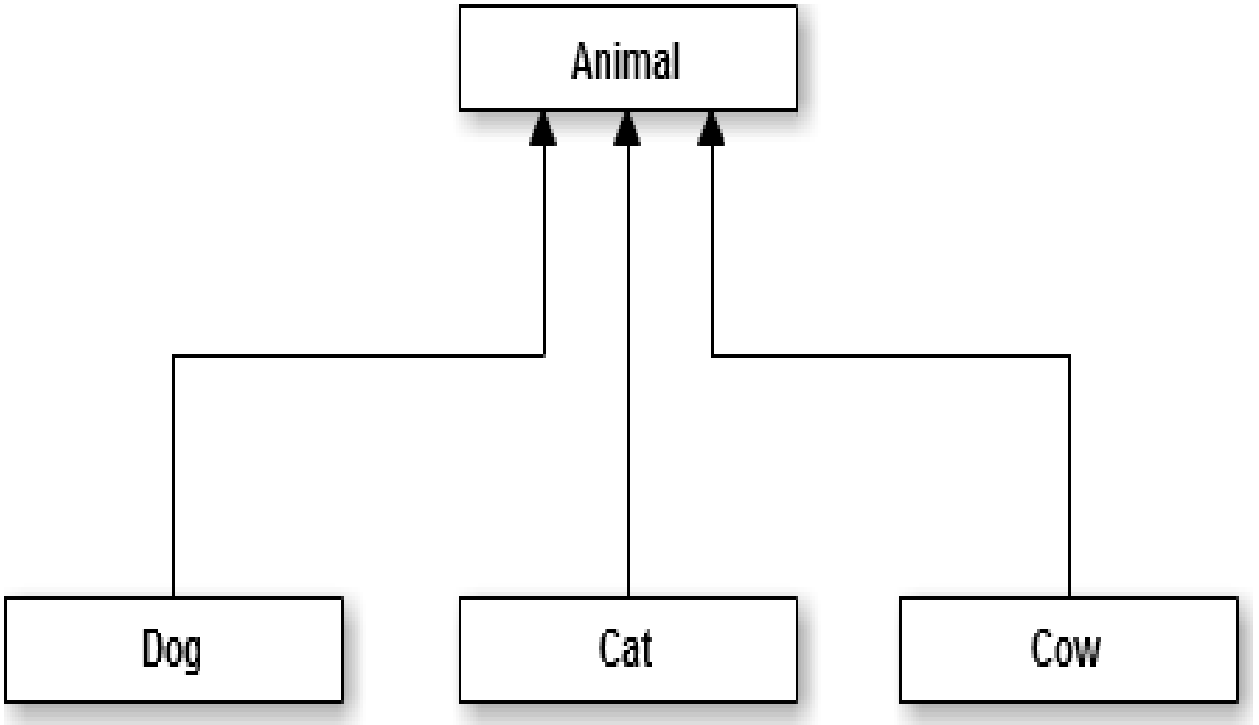
Object Relational Impedance Mismatch

- Identity
  I. Object Identity
  II. Object Equality
  III. Database Identity

Object Relational Impedance Mismatch

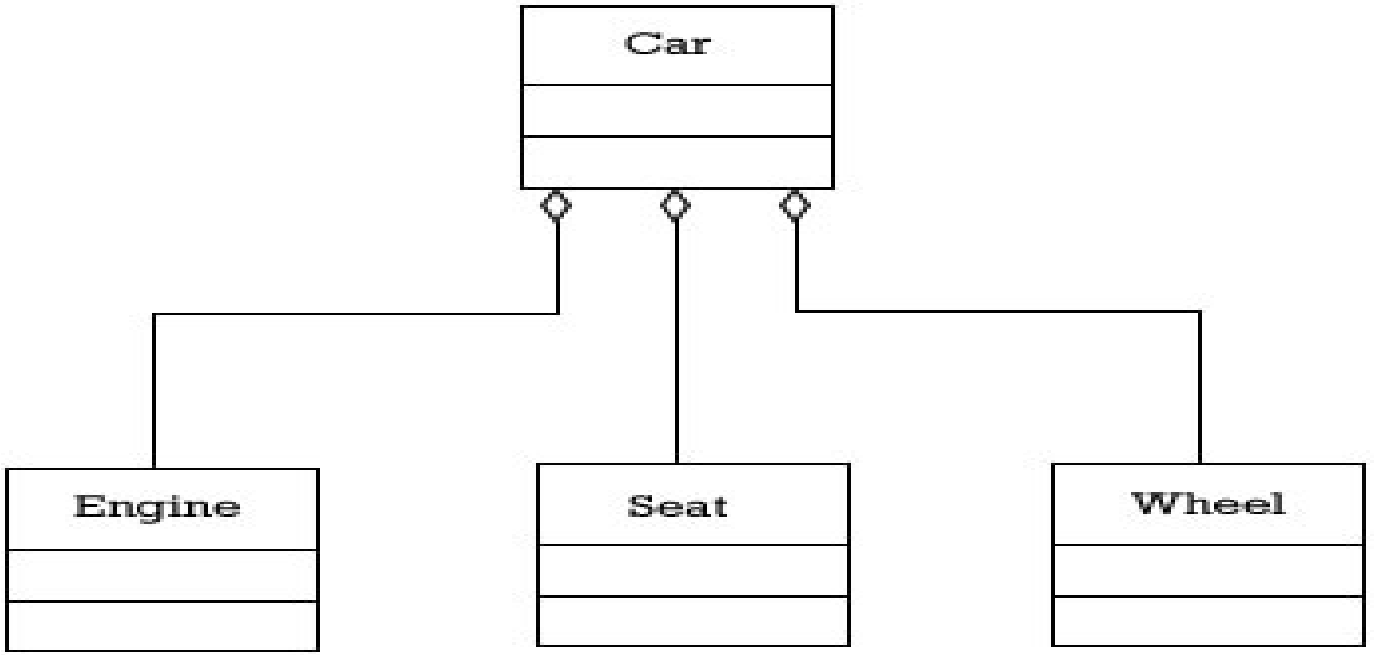
- Organization

  I. Inheritance and Polymorphic Queries



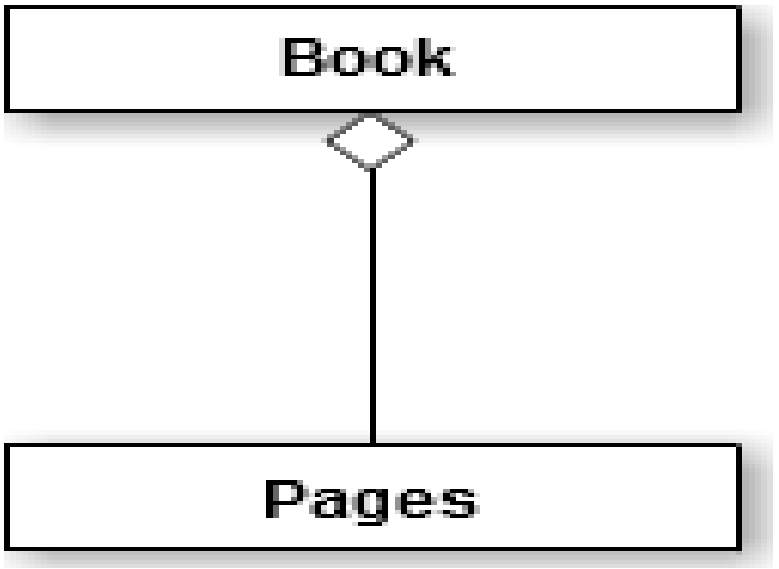
Object Relational Impedance Mismatch

- Organization
  - Association - Aggregation



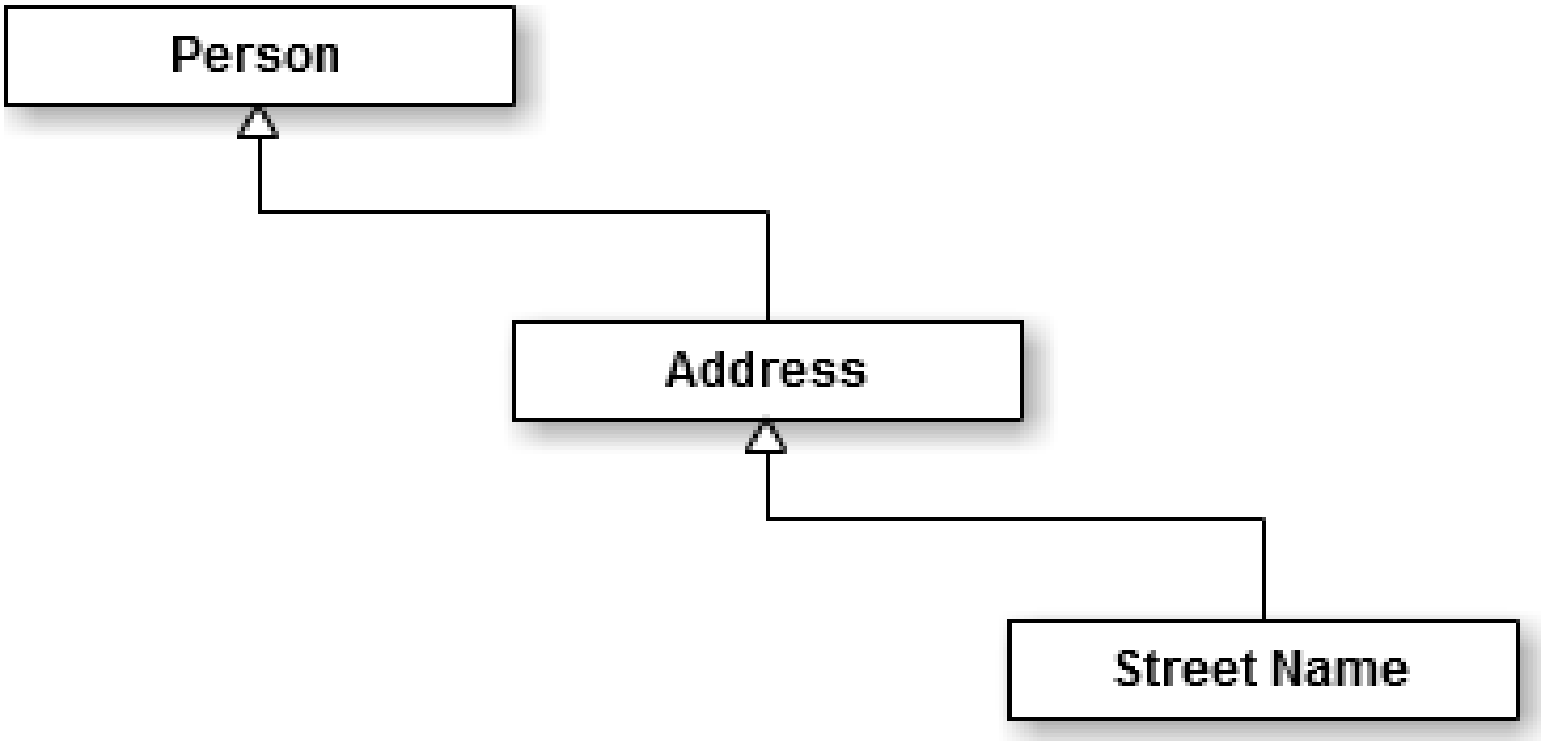
Object Relational Impedance Mismatch

- Organization
  - Association - Composition



Object Relational Impedance Mismatch

- Object Graph Navigation



XML Mapping

```
<hibernate-mapping package="com.botreeconsulting.lms.hibernate.xml.model">
    <class name="Book" table="BOOK" >
        <id name="id" type="long" column="bookId">
            <generator class="sequence" />
        </id>

        <property name="availabilityStatus" not-null="true">
            <type name="org.hibernate.type.EnumType">
                <param name="enumClass">com.botreeconsulting.lms.hibernate.xml.model.AvailabilityStatus</param>
            </type>
        </property>

        <!--
            1. Unidirectional many-to-one on a foreign key
            2. It is possible that there is no publisher detail available when adding a book
               Thus using default not-null="false" on publisher property.
        -->
        <many-to-one name="publisher" column="publisherId" cascade="save-update"/>

        <!--
            1. Uses inverse="false"
            2. The "cascade" attribute tells Hibernate to make any new Author
               instance persistent (that is, save it in the database)
               if the Author is referenced by a persistent Book.

               The cascade attribute is directional: It applies to only one
               end of the association.
        -->
        <set name="authors" table="BookAuthor" cascade="save-update">
            <key column="bookId" />
            <many-to-many column="authorId" class="Author" />
        </set>
    </class>
</hibernate-mapping>
```

'id' Generation Strategies

- *IDENTITY* : supports identity columns in DB2, MySQL, MS SQL Server, Sybase and HypersonicSQL. The returned identifier is of type long, short or int.
- *SEQUENCE* (called seqhilo in Hibernate) : uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, given a named database sequence.
- TABLE (called MultipleHiLoPerTableGenerator in Hibernate) : uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, given a table and column as a source of hi values. The hi/lo algorithm generates identifiers that are unique only for a particular database.
- AUTO: selects IDENTITY, SEQUENCE or TABLE depending upon the capabilities of the underlying database.

Inheritance – Mapping Classes and Tables

- Table per Class Hierarchy
  - Single Table
  - Discriminator Column
  - This mapping strategy provides good support for polymorphic relationships between entities and for queries that range over the class hierarchy. It has the drawback, however, that it requires that the columns that correspond to state specific to the subclasses be **nullable**.

Inheritance – Mapping Classes and Tables

- Table per Class Hierarchy

```
<class name="Person" table="PERSON">

        <id name="id" type="long" column="id">
            <generator class="sequence"/>
        </id>

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

        <property name="role" not-null="true">
            <type name="org.hibernate.type.EnumType">
                <param name="enumClass">com.botreeconsulting.hibernate.xml.model.Role</param>
            </type>
        </property>
        ...
        <subclass name="Administrator" discriminator-value="ADMIN">
            ...
        </subclass>

        <subclass name="Agent" discriminator-value="AGENT">
            ...
        </subclass>

</class>
```

Inheritance – Mapping Classes and Tables

- Table per Subclass
  - 1 table for Super class
  - 1 table for each Subclass
  - Each Sub Class table has columns specific to that subclass.
  - The primary key column(s) of the subclass table serves as a foreign key to the primary key of the superclass table.
  - It has the drawback that it requires that one or more join operations be performed to instantiate instances of a subclass. In deep class hierarchies, this may lead to unacceptable performance.

Inheritance – Mapping Classes and Tables

- Table per Subclass

```
<class name="Payment" table="PAYMENT">
        <id name="id" type="long" column="ID">
            <generator class="sequence"/>
        </id>
        <property name="firstName" type="string" not-null="true"/>
        <property name="lastName" type="string" not-null="true"/>
        ...
        <joined-subclass name="Administrator" table="ADMINISTRATOR">
            <key column="ID"/>
            ...
        </joined-subclass>
        <joined-subclass name="Agent" table="AGENT">
            <key column="ID"/>
            ...
        </joined-subclass>
</class>
```

Inheritance – Mapping Classes and Tables

- Table per Concrete Class
  - Usually used when the Super class is Abstract or is an Interface
  - 1 table per Concrete Class
  - Poor support for Polymorphic Relationships
  - It typically requires that SQL UNION queries (or a separate SQL query per subclass) be issued for queries that are intended to range over the class hierarchy.

Inheritance – Mapping Classes and Tables

- Table per Concrete Class (With Union)

```
<class name="Person">
        <id name="id" type="long" column="ID">
            <generator class="sequence"/>
        </id>
        <property name="firstName" type="string" not-null="true"/>
        <property name="lastName" type="string" not-null="true"/>
        ...
        <union-subclass name="Administrator" table="ADMINISTRATOR">
            ...
        </union-subclass>
        <union-subclass name="Agent" table="AGENT">
            ...
        </union-subclass>
</class>
```

Inheritance – Mapping Classes and Tables

- Table per Concrete Class (Without Union)

```
<class name="Administrator" table="ADMINISTRATOR" >
        <id name="id" type="long" column="id">
            <generator class="sequence" />
        </id>
        <property name="firstName" type="string" not-null="true"/>
        <property name="lastName" type="string" not-null="true"/>

        <property .....
        <property .....
        <property .....

</class>


<class name="Agent" table="AGENT" >
        <id name="id" type="long" column="id">
            <generator class="sequence" />
        </id>
        <property name="firstName" type="string" not-null="true"/>
        <property name="lastName" type="string" not-null="true"/>

        <property .....
        <property .....
        <property .....

</class>
```

Mapping Relations

- One to One

```
<one-to-one name="propertyName" class="ClassName"
cascade="cascade_style" constrained="true|false"
fetch="join|select"
property-ref="propertyNameFromAssociatedClass"
access="field|property|ClassName" formula="any SQL expression"
lazy="proxy|no-proxy|false" entity-name="EntityName"
node="element-name|@attribute-name|element/@attribute|."
embed-xml="true|false" foreign-key="foreign_key_name" />
```

Mapping Relations

- One to One

```
<class name="Person" table="PERSON">
        <id name="id" type="long" column="PERSON_ID">
            <generator class="sequence"/>
        </id>

        <property name="firstName" type="string" not-null="true"/>
        <property name="lastName" type="string" not-null="true"/>
        ....
        <one-to-one name="address" class="com.botreeconsulting.StockDetail"
            cascade="save-update"></one-to-one>
</class>

<class name="Address" table="ADDRESS">
        <id name="id" type="long" column="PERSON_ID">
            <generator class="foreign">
                <param name="property">PERSON</param>
            </generator>
        </id>
        ....
        <property name="streetName" type="string" not-null="true"/>
        <property name="cityName" type="string" not-null="true"/>
        ....
        <one-to-one name="person" class="com.botreeconsulting.Person"
            cascade="save-update"></one-to-one>
</class>
```

Mapping Relations

- Many to One / One To many

<many-to-one name="propertyName" column="column_name" class="ClassName" cascade="cascade_style" fetch="join|select" update="true|false" insert="true|false" property-ref="propertyNameFromAssociatedClass" access="field|property|ClassName" unique="true|false" not-null="true|false" optimistic-lock="true|false" lazy="proxy|no-proxy|false" not-found="ignore|exception" entity-name="EntityName" formula="arbitrary SQL expression" node="element-name|@attribute-name|element/@attribute|." embed-xml="true|false" index="index_name" unique_key="unique_key_id" foreign-key="foreign_key_name" />

Mapping Relations

- One to Many / Many to One

```
<class name="Book" table="BOOK" >
        <id name="id" type="long" column="bookId">
            <generator class="native" />
        </id>
        ....
        <many-to-one name="publisher" column="publisherId" cascade="save-update"/>
</class>

<class name="Publisher" table="PUBLISHER" >
        <id name="id" type="long" column="publisherId">
            <generator class="native" />
        </id>
        ....
</class>
```

Mapping Relations

- Many to Many

```
<element column="column_name" formula="any SQL expression"
type="typename" length="L" precision="P" scale="S" not-
null="true|false" unique="true|false" node="element-name"/>
```

```
<many-to-many column="column_name" formula="any SQL
expression" class="ClassName" fetch="select|join" unique="true|
false" not-found="ignore|exception" entity-name="EntityName"
property-ref="propertyNameFromAssociatedClass" node="element-
name" embed-xml="true|false"/>
```

Mapping Relations

- Many To Many

```
<class name="Book" table="BOOK" >
      <id name="id" type="long" column="bookId">
            <generator class="native" />
      </id>
      <property name="isbn" type="string" not-null="true" unique="true"/>
      <property name="title" type="string" not-null="true"/>

      <set name="authors" table="BookAuthor" cascade="save-update">
            <key column="bookId" />
            <many-to-many column="authorId" class="Author" />
      </set>

</class>

<class name="Author" table="AUTHOR" >

      <id name="id" type="long" column="authorId">
            <generator class="native" />
      </id>
      <property name="firstName" type="string" not-null="true"/>
      <property name="lastName" type="string" not-null="true"/>

      <set name="books" table="BookAuthor" inverse="true" cascade="save-update">
            <key column="authorId" />
            <many-to-many column="bookId" class="Book" />
      </set>

</class>
```

End of
Spring – Hibernate Training Program
Session 2