



Mindtree

Welcome to possible

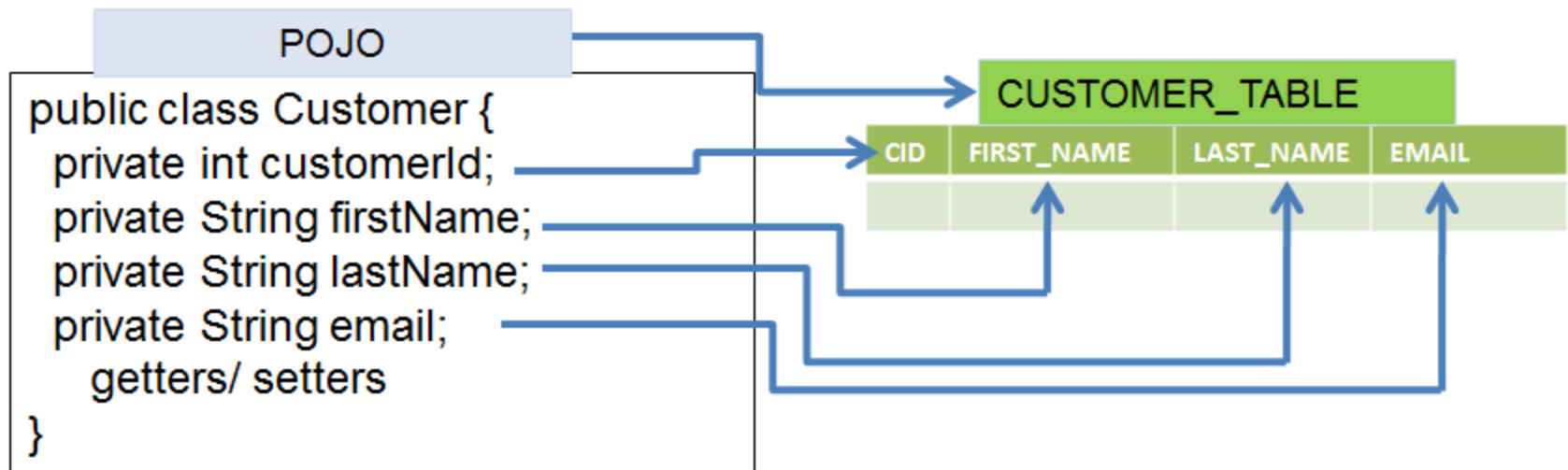
Object Relational Mapping (ORM)

Objectives

- Define ORM.
- ORM Mapping
- Introduction to Hibernate framework
- Hibernate configuration

Object Relational Mapping (ORM)

- ORM is a framework used to persist objects present in application to the tables in a relational database using metadata that describes the mapping between objects and the table.
- A Simple mapping will have a class mapped to a table and java properties are mapped to columns of a table



Object Relational Mapping (ORM)

- ORM solution should:
 - Facility for specifying mapping metadata
 - API for CRUD operations on objects
 - API for queries that refer to classes and properties rather than referring Table and columns.
 - Perform Dirty checking
 - Perform Lazy association fetching.
 - Support for Cache.

Object Relational Mapping (ORM)

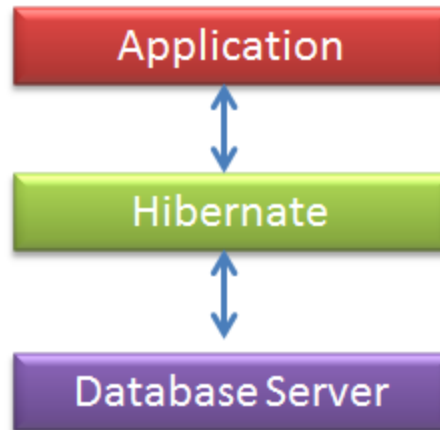
- Why ORM?
 - Productivity
 - ORM eliminates much of the pushing and pulling code from database and helps you to concentrate on business problem.
 - Maintainability
 - ORM reduces lines of code. Less code means easier to refactor.
 - ORM insulates Java side to database side. One model is unaffected by small change in other model.
 - Vendor independence
 - ORM insulates application from underlying database and SQL Dialect, hence it is easier to develop cross-platform application using ORM.

Object Relational Mapping (ORM)

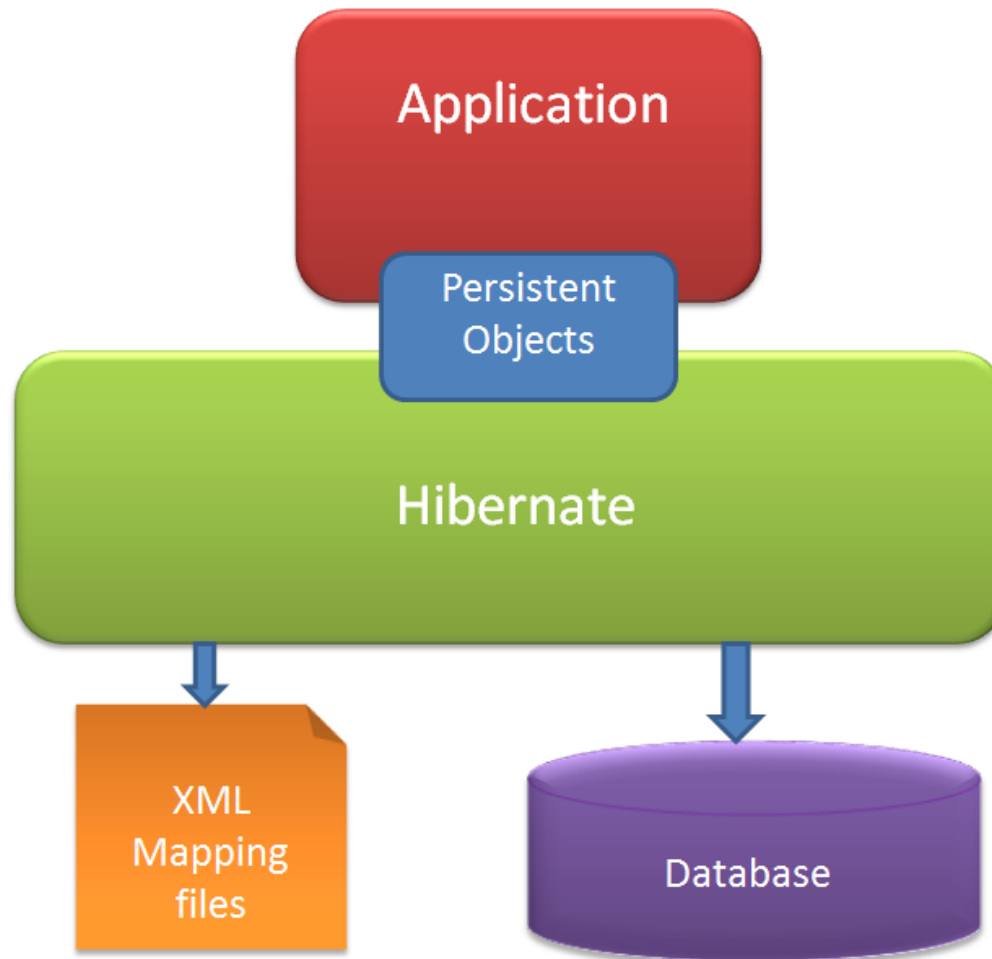
- Some list of frameworks which are available for Java developers?
 - iBATIS
 - iBATIS is not a ORM, it is a Data Mapper.
 - Good choice when a developer does not have control over SQL database schema.
 - Developer writes SQL for CRUD in XML instead of in a application.
 - After the development team moved from ASF to Google code it is named MyBatis.
 - TopLink
 - TopLink is an ORM developed for Smalltalk, BEA Systems added Java Version. TopLink is now acquired by Oracle Corporation.
 - Hibernate
 - Hibernate was started by Gavin King as an alternative to EJB2.x entity bean.
 - Jboss Inc (Red Hat) provide support to Hibernate.
 - Hibernate scores over other ORM frameworks in:
 - Performance
 - Providing extensibility
 - Excellent mapping API even when used with Native SQL.
 - Rich Key generation implementations.

Hibernate

- Hibernate is an open source object/relational mapping (ORM) implementation framework.
- Hibernate mediates the application's interaction with a relational database, leaving the developer free to concentrate on the business problem at hand.



Hibernate Architecture



Hibernate Configuration

1. DTD

Used to validate the configuration file.

2. Root Element

The configuration should have one root element `<hibernate-configuration>`.

3. Session Factory

All the hibernate configurations should be child elements to `<session-factory>`

4. Property

Property element is used to configure all database specific properties like `driver_class`, `url`, `username`, etc.

5. Mapping

The mapping element is used to add mapping files to configuration

Configuration file: hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration PUBLIC 1
"-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration> 2
<session-factory> 3
<property name="hibernate.connection.driver_class"></property>
<property name="hibernate.connection.url"></property>
  <property name="hibernate.connection.username"></property>
  <property name="hibernate.connection.password"></property> 4
  <property name="hibernate.connection.pool_size">10</property>
  <property name="dialect"></property>
  <mapping resource="" /> 5
</session-factory>
</hibernate-configuration>
```

Hibernate Configuration

- `org.hibernate.cfg.Configuration`
- The Configuration has two parts:
 - The database setup
 - Details like JNDI, `driver_class`, `username`, `password`, `dialect`.
 - The class mapping setup
 - The hibernate mapping files which contains mapping between Java classes and database tables
- `Configuration cfg = new Configuration();`
 - Configuration object is created using information present in `hibernate.cfg.xml`
 - You can add additional information to this object using
 - `setProperty(String propertyName, String value)`

Hibernate SessionFactory

- org.hibernate.SessionFactory
 - A Factory class to create Hibernate Session's
 - SessionFactory instance is singleton per database
 - SessionFactory instance is thread safe (immutable) cache of compiled mappings for a single database.

- Usage:

```
Configuration cfg = new Configuration();  
SessionFactory sessionFactory =  
    cfg.configure().buildSessionFactory();
```

OR

```
SessionFactory sessionFactory =  
    new Configuration().configure().buildSessionFactory();
```

Hibernate Session

- `org.hibernate.Session`
 - Hibernate Session wraps operations related to database.
 - Hibernate Session is single threaded.
 - When a Session object is instantiated, a connection is established to the database
 - Session holds a mandatory (first-level) cache of persistent objects
- Usage:

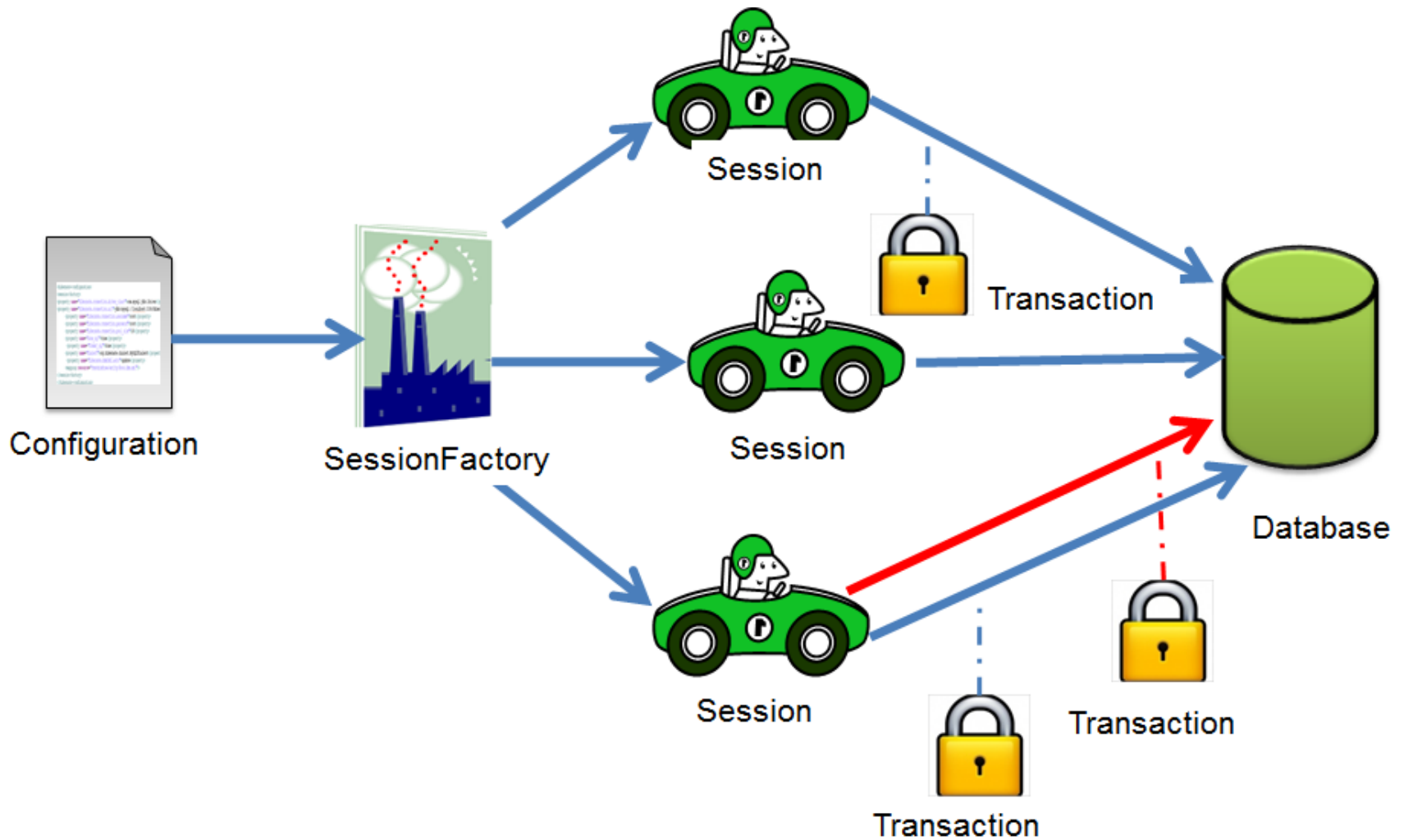
```
Session session = sessionFactory.openSession();
```

Hibernate Transaction

- org.hibernate.Transaction
 - The Transaction interface lets you make sure that all the operations on persisted objects occur in a transaction supported by either JDBC or JTA.

```
Session session = null;
Transaction tx = null;
try {
    session = sessionFactory.openSession();
    tx = session.beginTransaction();
    // CRUD here
    tx.commit();
} catch(HibernateException ex) {
    tx.rollback();
}
```

Hibernate Framework objects



Hibernate Mapping

Book.hbm.xml

```
<hibernate-mapping package="com.mindtree.entity">
  <class name="Book" table="TBOOK">
    <id name="bookId" type="long"
      column="BOOK_ID" unsaved-value="0">
      <generator class="increment" />
    </id>
    <property name="title"
      column="BOOK_TITLE"
      type="string"
      unique="true" length="100"/>
    <property name="price"
      column="BOOK_PRICE"
      type="double" />
  </class>
</hibernate-mapping>
```

Book class

com.mindtree.entity.Book

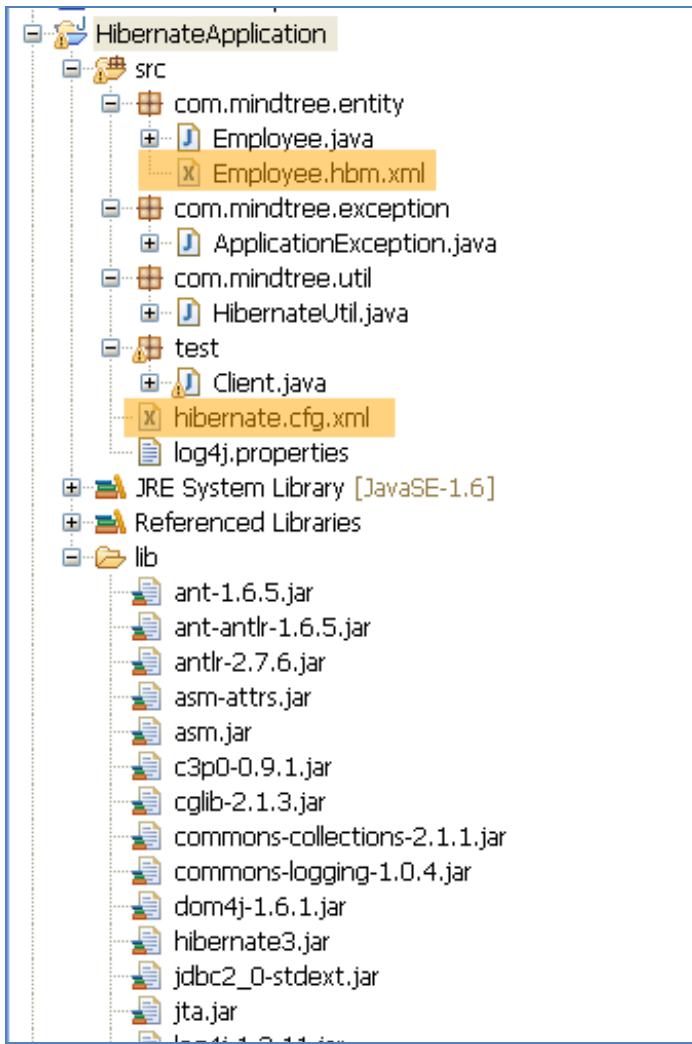
- serialVersionUID: long
- bookId: Long
- title: String
- price: double

- getBookId(): Long
- setBookId(bookId: Long): void
- getTitle(): String
- setTitle(title: String): void
- getPrice(): double
- setPrice(price: double): void

TBOOK table

Field	Type	Null	Key	Default	Extra
BOOK_ID	bigint(20)	NO	PRI		
BOOK_TITLE	varchar(100)	YES	UNI	NULL	
BOOK_PRICE	double	YES		NULL	

Hibernate application folder structure



Configuration file: `hibernate.cfg.xml` should be in “src” folder.

Mapping files: “*.hbm.xml” preferably place them along with your entity classes.

Code Snippet: Using Hibernate API to persist an entity

```
Book book = new Book();
book.setTitle("Hibernate in Action");
book.setPrice(450.55);

org.hibernate.cfg.Configuration cfg =
    new org.hibernate.cfg.Configuration();
org.hibernate.SessionFactory sessionFactory =
    cfg.configure().buildSessionFactory();
org.hibernate.Session session = null;
org.hibernate.Transaction tx = null;

try {
    session = sessionFactory.openSession();
    tx = session.beginTransaction();
    Long bookId = (Long) session.save(book);
    System.out.println("Book saved with ID :" + bookId);
    tx.commit();
} catch (HibernateException e) {
    e.printStackTrace();
    tx.rollback();
}
```



Mindtree

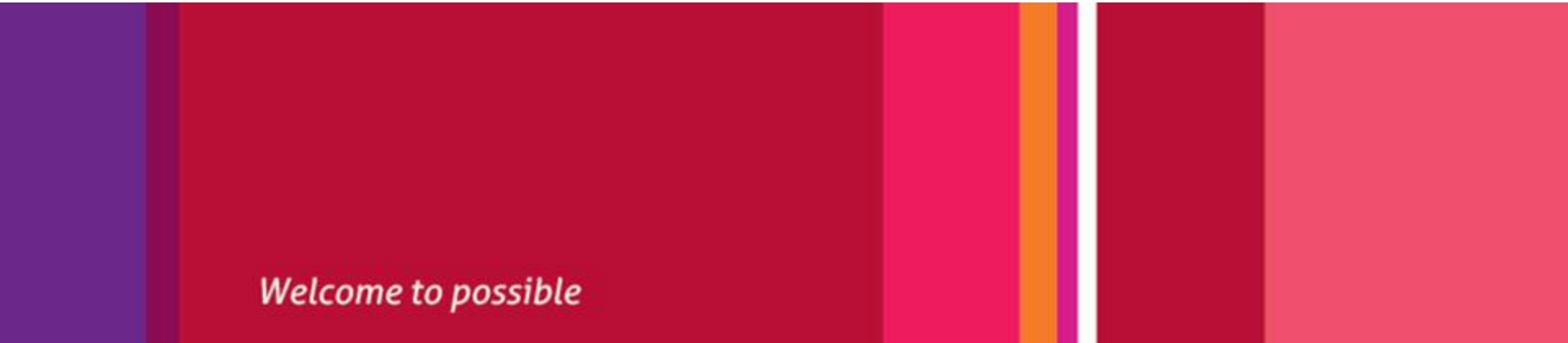
Welcome to possible

References

Contains the reference that will supplement the self learning and will be needed for completing the assignments & practice questions

References

- Books:
 - Java Persistence with Hibernate
 - Manning Publications
 - Professional Hibernate
 - Wrox Publications
- Documentation
 - Hibernate Documentation
 - <http://www.hibernate.org/docs.html>



Welcome to possible

India | USA | UK | Germany | Sweden | Belgium | France | Switzerland | UAE | Singapore | Australia | Japan | China