Works Done:

* Research Spring MVC Framework
  + DispatcherServlet class (front-controller): tất cả các request đều phải thông qua class này.
  + MVC: all the incoming request >>> DispatcherServlet >>> Hander Mapping (xml file) : kiểm tra các request và hướng đến controller tương ứng >>> Controller: handle requests and return an object of ModelAndView >>> Dispatcher: check view resolver in xml file and show return to View (jsp,…).
  + @Controller: this annotation marks a class as Controller.
  + @RequestMapping: this annotation map a request from client to a method ( or class ) tương ứng.
  + Web.xml: config all requests will be forwarded to the DispatcherServlet.
  + Define the bean in xml file to work with DispatcherServlet:
    - <context:component-scan>: where DispatcherServlet will search controller class.
    - Bean class InternalResourceViewResolver: to work with jsp page.
* Research Spring MVC Tiles: chia layout thành nhiều phần riêng biệt, dễ dàng cập nhật từng phần riêng biệt.
* Spring JDBC Templete: provides methods to write the queries directly, save lots of work and time.
  + JDBC – Java DataBase Connectivity
  + Method of Spring JDBC Template:
    - Public int update(String sql) : insert,update, delete records.
    - Public int update(String sql, Object…args): insert, update, delete using PreparedStatement given arguments.
    - Public void execute(String query) : to execute query.
    - Public T execute(Spring sql, PreparedStatementCallback action): execute query by PreparedStatement callback.
    - Public T query(String sql, ResultSetExtractor rse): fetch records using ResultsSetExtractor.
    - Public List query(String sql, RowMapper rse): fetch records using RowMapper.
  + PreparedStatement: to use execute() method with parameters query
    - Public T execute(String sql, PreparedStatementCallback<T>){\*};
    - PreparedStatementCallback: processes the input parameterized and output results.
    - Only one method:
      * \*Public T doInPreparedStatement(PreparedStatement ps) throws SQLException, DataAccessException{}
  + RowMapper:
    - Public T query(String sql, RowMapper<T> rm){\*}
    - \*Public T mapRow(ResultSet rs, int rowNumber) throws SQLException{}
  + SimpleJDBCTemplete
* Research Virtual-Host Apache Tomcat.
  + Create virtual host on local & virtual machine with Tomcat.
  + Configure file hosts: hostname with ipaddress from windows/system32/driver/etc/hosts(local) & /etc/hosts(virtual machine)
  + Configure server.xml in Tomcat: to create virtual host, create different hostname with the same ipaddress.
* RESTeasy:
  + REST: Representational State Transfer
  + Annotation Support In REST
    - @Path: path for a class or method.
    - @GET: get (select) resource from server.
    - @POST: create new resource
    - @DELETE: delete resource
    - @PUT: update resource.
    - @Produces: It specifies the MIME media type of the response of a resource.
    - @Consumes: It specifies the MIME media type of the request to a resource.
    - @PathParam: to extract a parameter from the uri path of resource.
    - @QueryParam: to extract a parameter form the uri path of resource.
* Research Maven
  + Apache Maven is a build tool to support the developer at the whole process of a software project.
  + Features of Maven:
    - Convention over configuration
    - Dependency management
    - Central repository
    - Extensible via plug-ins

**HIBERNATE FRAMEWORK**

**Hibernate Framework**

* To interact the database with java application. Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool which simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



**Advantages of Hibernate Framework**

* Opensource and lightweight.
* Fast performance: because cache ( 1st cache-default & 2nd cache) is internally used in hibernate framework.
* Database Independent query: HQL (Hibernate Query Language) generates the database independent queries.
* Automatic table creation.
* Simplifies complex join: fetch data from multiple tables.
* Provides query statistics and database status.

**Hibernate Architecture**



Hibernate Architecture with mapping file and configuration file.

* Mapping file ( \*.hbm.xml ):

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.javatpoint.mypackage.Employee" table="emp1000">

<id name="id">

<generator class="assigned"></generator>

</id>

<property name="firstName"></property>

<property name="lastName"></property>

</class>

</hibernate-mapping>

employee.hbm.xml

* Configuration file ( hibernate.cfg.xml ):

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hbm2ddl.auto">update</property>

<property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

<property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

<property name="connection.username">system</property>

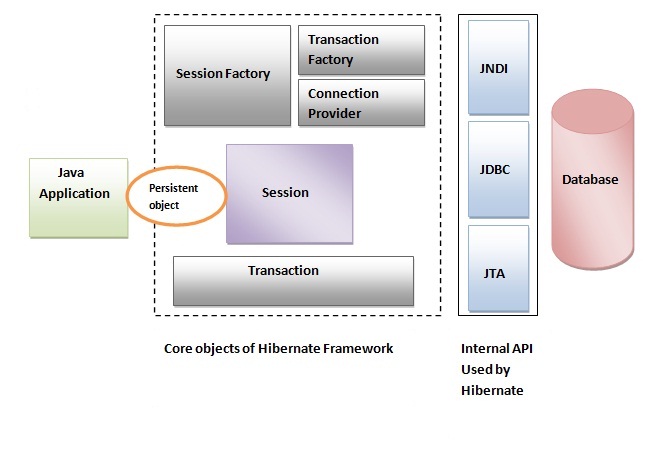
<property name="connection.password">oracle</property>

<property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

<mapping resource="employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

* Session-factory: based on different servers (dialect.Oracle, dialect.MySql, dialect.SQLServer,jdbc:oracle, jdbc:mysql, ….Hibernate use many objects.

**Elements of Hibernate Architecture**

* SessionFactory: is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data, provides factory method to get the object of Session.
* Session: provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It hold a first-level cache (mandatory) of data. It provides methods to insert, update and delete the object, factory methods for Transaction, Query and Criteria.
* Transaction: “được ăn cả, ngã về không”, khi thực thi 1 khối lệnh, 1 câu lệnh xảy ra lỗi thì toàn bộ khối lệnh sẽ bị hủy, dữ liệu được giữ nguyên.
* ConnectionProvider: a factory of JDBC connections. It abstracts the application from DriverManager or DataSource.
* TransactionFactory.

**Hibernate with Annotation**

* Core advantage: don’t need to create mapping file (\*.hbm.xml). Hibernate annotations are used to provide the meta data.
* @Entity: mark this class as an entity.
* @Table: to map the table name where data of this entity is to be persisted. If we don’t use @Table, hibernate will use the class name as the table name by default.
* @Id: marks the identifier for this entity.
* @Column: the details of the column for this property or field. If we don’t use @Column, property name will be used as the column name by default.
* Update later.

**Hibernate Inheritance Mapping**

* Table Per Hierarchy: map the whole hierarchy by single table only. (by xml or annotation)
* Table Per Concrete Class: map table with table having no relations to each other. (by xml or annotation)
* Table Per Subclass: subclass mapped tables are related to parent class mapped table by primary key and foreign key relationship.