实验五 Hibernate、Spring 及 SSH 集成

实验内容

学习 Hibernate、Spring 应用的基本开发，及 SSH 的简单集成

实验目的

了解 Hibernate、Spring 框架的机制掌

握 Hibernate、Spring 应用的配置掌握

Spring 中 Bean 注解扫描装配掌握

Hibernate 基本编程

掌握 SSH 的简单集成

环境要求

服务器：Tomcat67.0 或更高集成开发环境：MyEclipse

2014 或更高

实验指导

1. Hibernate 简单使用

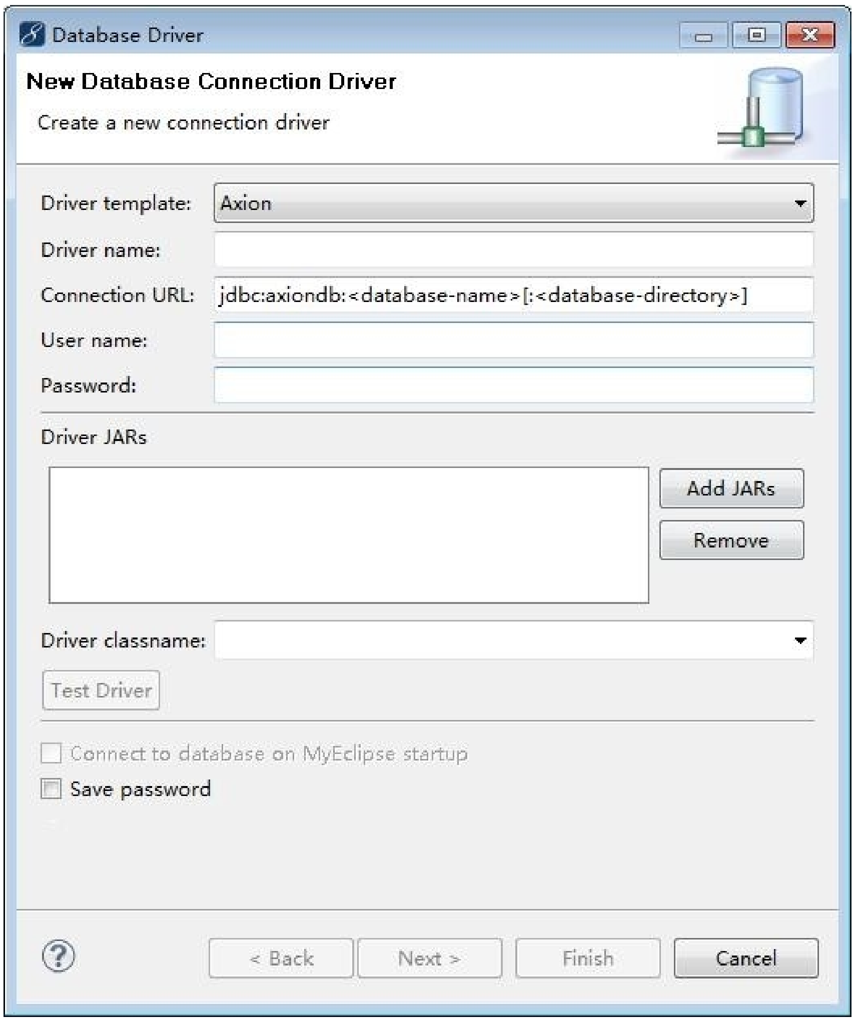
1) 创建表

在 MySQL 数据库服务器上的 test 中创建 student 表，字段如下：

字段名	类型	说明
id	Integer	自增，主键
xh	varchar（10）	
name	varchar（10）	
sex	varchar（2）	
className	varchar（16）	

2) 创建数据库连接

通过菜单“window”->“show view”打开“DB Browser”窗口，在该窗口中点击右键，在弹出菜单总选择“New”，打开如下窗口。


Database Driver

New Database Connection Driver

Create a new connection driver

Driver template: Axion

Driver name:

Connection URL: jdbc:axiondb:<database-name>[:<database-directory>]

User name:

Password:

Driver JARs

Add JARs

Remove

Driver classname:

Test Driver

Connect to database on MyEclipse startup

Save password

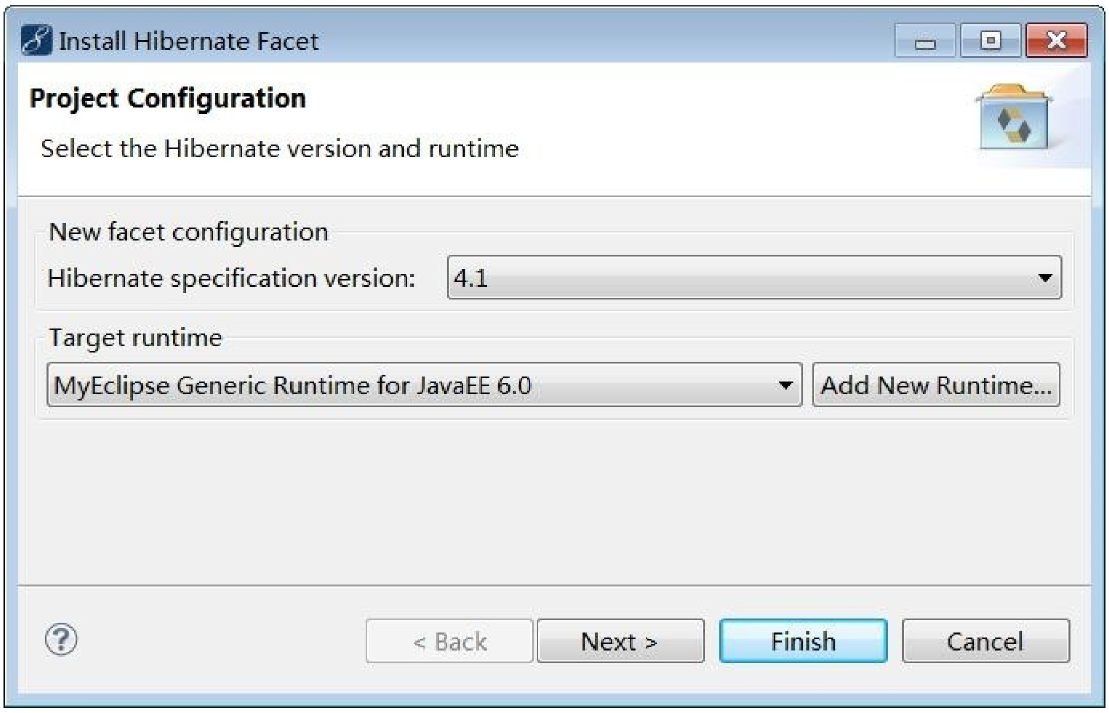
< Back　Next >　Finish　Cancel

在“Driver template”中选择“MySQL”，“Driver name”中输入一个名称（mysql），修改“Connection URL”为：jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8，输入数据库的用户名和口令，

点击“Add JARS”按钮添加 Mysql 的驱动 Jar 包，点击“Finish”按钮完成创建。

3) 创建工程及添加 **Hiberbate** 支持创建一个 Java 工程，名称为 hh。

在左侧“package explore”窗口的“hh”工程名上点击右键，在弹出的菜单中选择“MyEclipse” ->“Project Facets[Capapilities]”->“Install Hibernate Facet”，弹出如下对话框：


Install Hibernate Facet

Project Configuration

Select the Hibernate version and runtime

New facet configuration

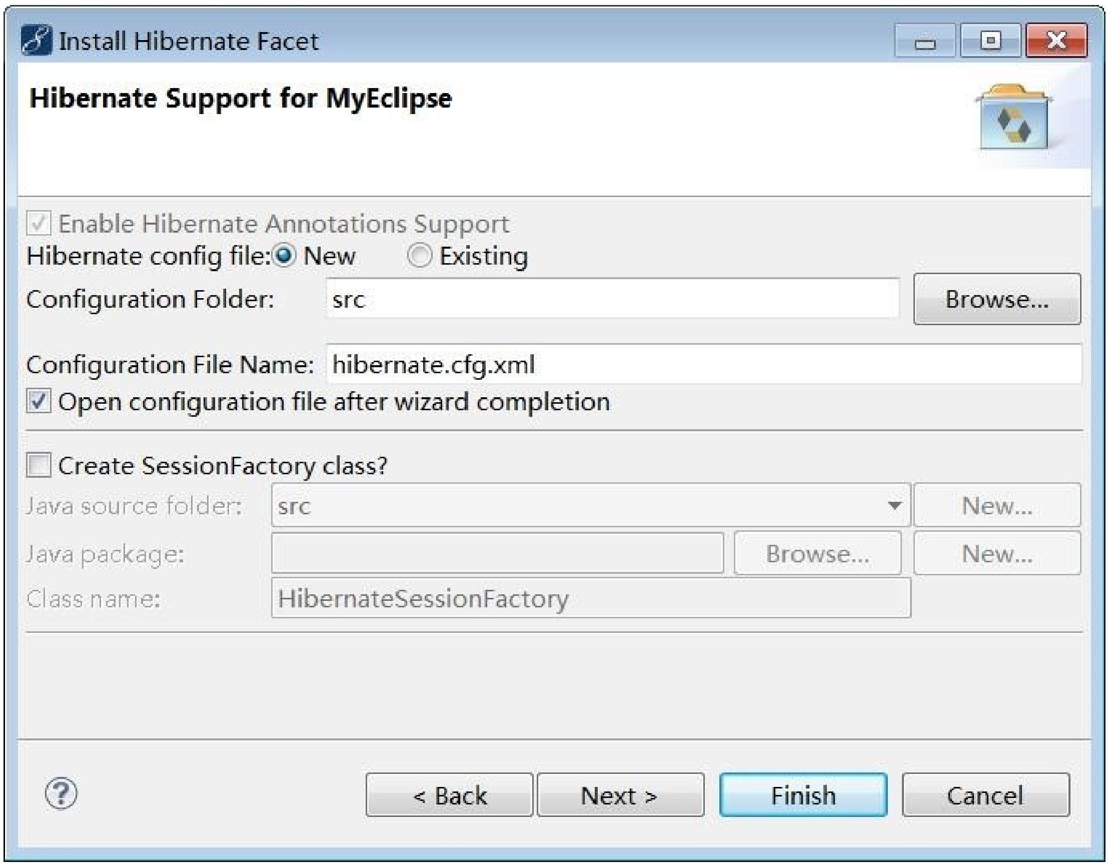
Hibernate specification version: 4.1

Target runtime

MyEclipse Generic Runtime for JavaEE 6.0 | Add New Runtime...

< Back | Next > | Finish | Cancel

在本对话框中选择 Hibernate 的版本及服务器，使用默认选择即可，点击“Next”按钮，弹出如下窗口：


Install Hibernate Facet

Hibernate Support for MyEclipse

[✓] Enable Hibernate Annotations Support

Hibernate config file: [● New] ○ Existing

Configuration Folder: src | Browse...

Configuration File Name: hibernate.cfg.xml

[✓] Open configuration file after wizard completion

[ ] Create SessionFactory class?

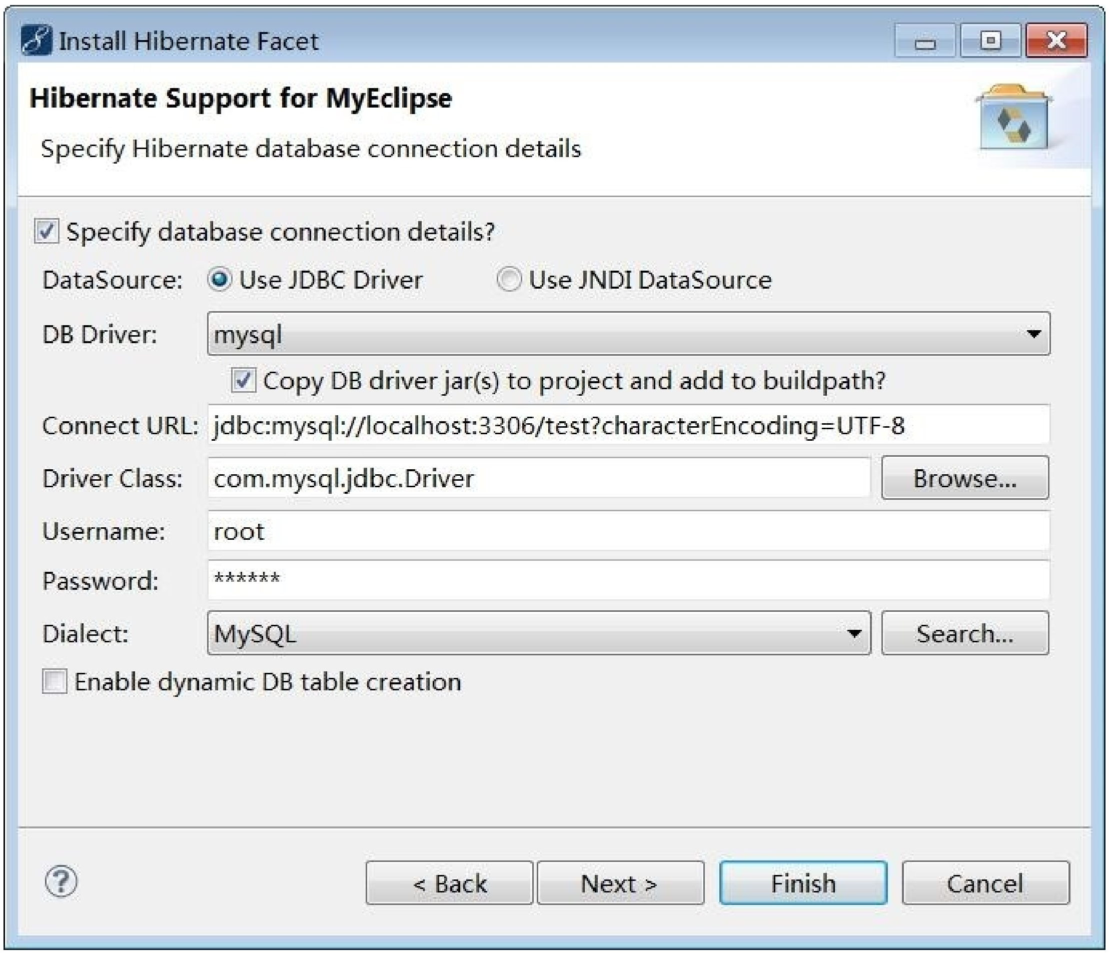
Java source folder: src | New...

Java package: | Browse... | New...

Class name: HibernateSessionFactory

< Back | Next > | Finish | Cancel

在本对话框中上半部分用于指定 Hibernate 的配置文件的位置及名称，使用默认即可；下半部分（Create SessionFactory class）用于创建一个会话工厂工具类，取消选择。点击“Next”按钮弹出如下窗口：



Install Hibernate Facet

Hibernate Support for MyEclipse

Specify Hibernate database connection details

[✓] Specify database connection details?

DataSource: [● Use JDBC Driver] ○ Use JNDI DataSource

DB Driver: mysql

[✓] Copy DB driver jar(s) to project and add to buildpath?

Connect URL: jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8

Driver Class: com.mysql.jdbc.Driver

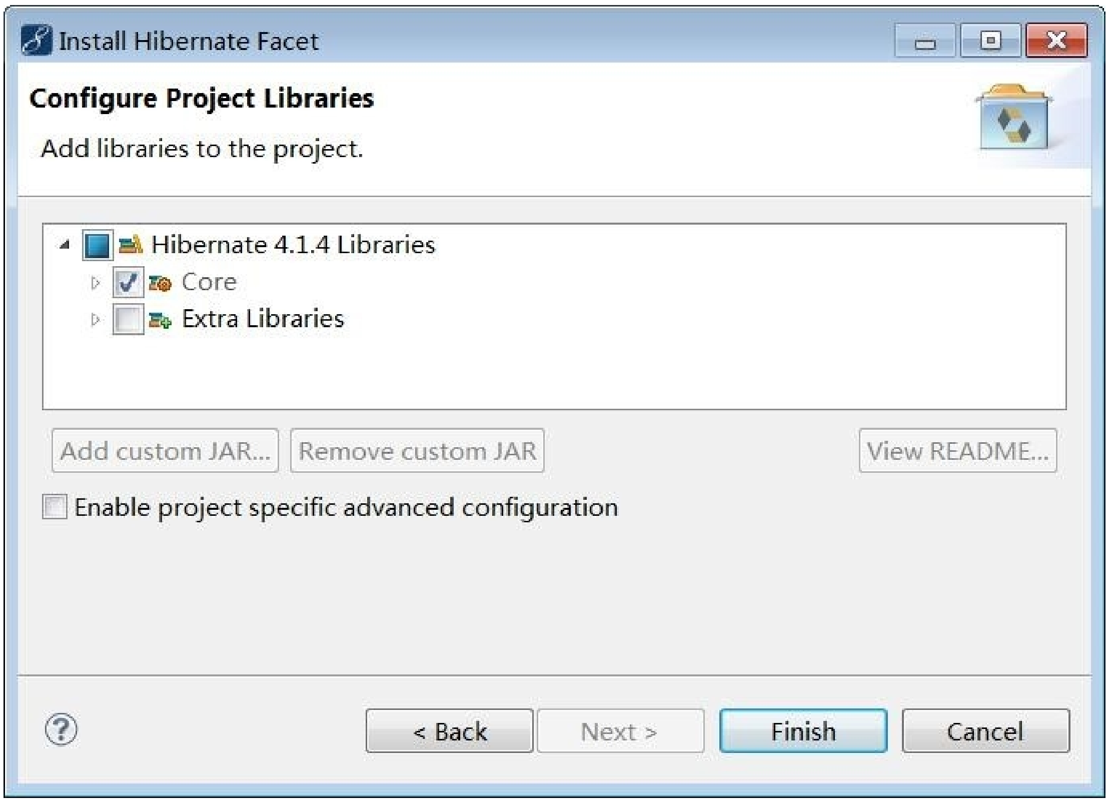
Username: root

Password: ******

Dialect: MySQL

[ ] Enable dynamic DB table creation

本窗口用于选择设置在 Hibernate 中使用数据库的信息，在“DB Driver”中选择我们在第二步创建的数据库连接，点击“Next”按钮，弹出如下对话框：



Install Hibernate Facet

Configure Project Libraries

Add libraries to the project.

- Hibernate 4.1.4 Libraries
  - [✓] Core
  - [ ] Extra Libraries

[ ] Enable project specific advanced configuration

本对话框用于选择所需的 jar 包，使用默认（选择 Core）即可，点击“Finish”按钮即可完成工程

对 Hibernate 的支持。

4) 编写代码

①实体——**student.java**

```
package entity;


import javax.persistence.Entity; import

javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

@Table(name="student")

public class Student {

    @Id

    @GeneratedValue(strategy=GenerationType.AUTO)

    private int id; private String xh; private String name;

    private String className; private String sex; public

    int getId() { return id;

    }

    public void setId(int id)

        { this.id = id;

    }

    public String getXh()

        { return xh;

    }

    public void setXh(String xh)

        { this.xh = xh;

    }
```

```
        public String getName()
            { return name;
        }
        public void setName(String name)
            { this.name = name;
        }
        public String getClassName()
            { return className;
        }
        public void setClassName(String className)
            { this.className = className;
        }
        public String getSex()
            { return sex;
        }
        public void setSex(String sex)
            { this.sex = sex;
        }
        @Override public
        String toString() {
            return "学号： " + xh + "\t"
                + "姓名： " + name + "\t"
                + "性别： " + sex + "\t"
                + "班级： " + className + "\t";
        }
    }
```

②**Hibernate** 配置文件——**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>

        <!--配置数据库-->

        <session-factory>

            <property

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

            <property

name="connection.url">jdbc:mysql://localhost:3306/test?characterEncoding=UTF

-8</property>

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

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

            <property

name="connection.driver_class">com.mysql.jdbc.Driver</property>

            <property name="myeclipse.connection.profile">mysql</property>

            <!--注册实体 -->

            <mapping class="entity.Student"/>

        </session-factory>

        </hibernate-configuration>
```

③测试代码：

```
    import java.io.UnsupportedEncodingException;

    import java.util.List; import java.util.Scanner;

    import org.hibernate.Query; import

    org.hibernate.SessionFactory; import

    org.hibernate.cfg.AnnotationConfiguration;

    import org.hibernate.classic.Session; import

    entity.Student;


    public class Ha {


        /**
```

```
         * @param args

         * @throws UnsupportedEncodingException */

        public static void main(String[] args) throws

UnsupportedEncodingException {


            Ha h = new Ha();

            Scanner sc = new Scanner(System.in);


            while(true){ switch

                (h.menu(sc)) { case 1:

                    h.list();

                    break;

                case 2:

                    Student stu = new Student();

                    stu.setXh(sc.next());

                    stu.setName(sc.next()); byte[] b =

                    stu.getName().getBytes("utf-8");

                    stu.setName(new String(b,"utf-8"));

                    stu.setSex(sc.next());

                    stu.setClassName(sc.next());

                    h.add(stu);

                    h.list();

                    break;

                case 3:

                    h.edit(sc.nextInt(), sc);

                    break;

                case 4:

                    int id = sc.nextInt();

                    h.deleteByKey(id);

                    h.list();

                    break;
                case 5:
                    System.exit(0);
                break; default:
                    break;
                }
            } }

        private SessionFactory sf;

        public Ha(){
            Configuration config = new
Configuration().configure("/hibernate.cfg.xml");
            sf = config.buildSessionFactory();
        }

        private int menu(Scanner sc){
            System.out.println("1.list");
            System.out.println("2.add");
            System.out.println("3.edit");
            System.out.println("4.delete");
            System.out.println("5.exit");
            return sc.nextInt();
        }

        /**
         * 学生信息列表
         */
        private void list(){
            Session session = sf.openSession();
            Query qry = session.createQuery("from Student");
            List<Student> stus = qry.list();
```

```
    for(Student stu:stus){
        System.out.println(stu);
    }
    session.close();
}

/**
 * 添加学生信息
 * @param stu
 */ private void add(Student
stu){ Session session =
sf.openSession();
session.beginTransaction();
session.save(stu);
session.getTransaction().commit();
    session.close();
}

private void edit(int id,Scanner sc){
    Session session = sf.openSession();
    Student stu = (Student) session.get(Student.class, id);
    System.out.println(stu); stu.setXh(sc.next());
    stu.setName(sc.next()); stu.setSex(sc.next());
    stu.setClassName(sc.next()); session.beginTransaction();

    session.update(stu);
    session.getTransaction().commit();
    session.close();
}
```

```
/**
 * 按主键删除
 * @param id
 */
private void deleteByKey(int id){
    Session session = sf.openSession();
    Student stu = (Student) session.get(Student.class, id);
    session.beginTransaction(); session.delete(stu);
    session.getTransaction().commit();
    session.close();
}
}
```

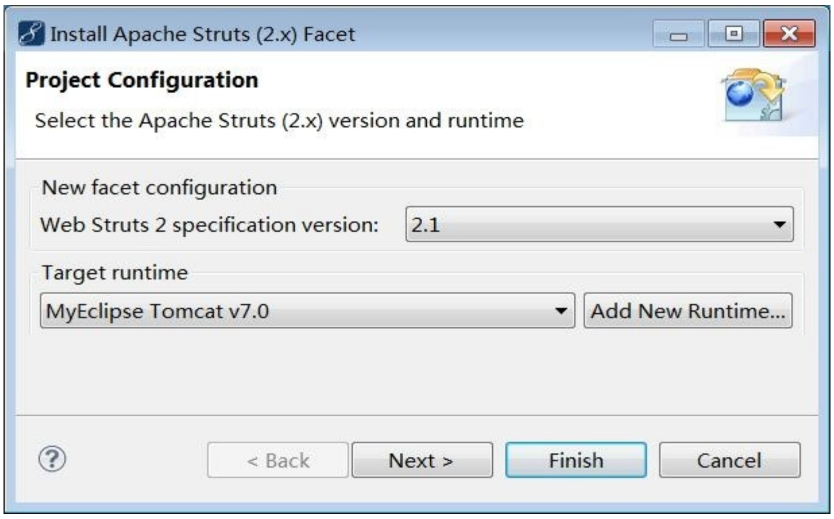
2. SSH 集成实例

功能：学生信息管理系统表同第

1 题。

（1）创建一个 Web 工程 struts1。

（2）Struts 集成在工程名上点击右键，在弹出菜单中选择“MyEclipse”->“Project Facets [Capabilites]”->“Install Apache Struts(2.x) Facet”，弹出的对话框中选择如下图：


Install Apache Struts (2.x) Facet

Project Configuration

Select the Apache Struts (2.x) version and runtime

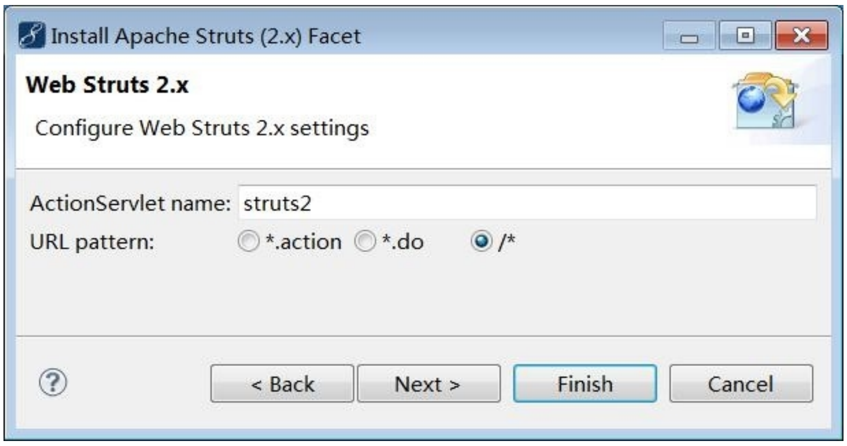
New facet configuration

Web Struts 2 specification version: 2.1

Target runtime

MyEclipse Tomcat v7.0

点击“Next”按钮，在对话框中配置 Struts 的控制器如下图：


Install Apache Struts (2.x) Facet

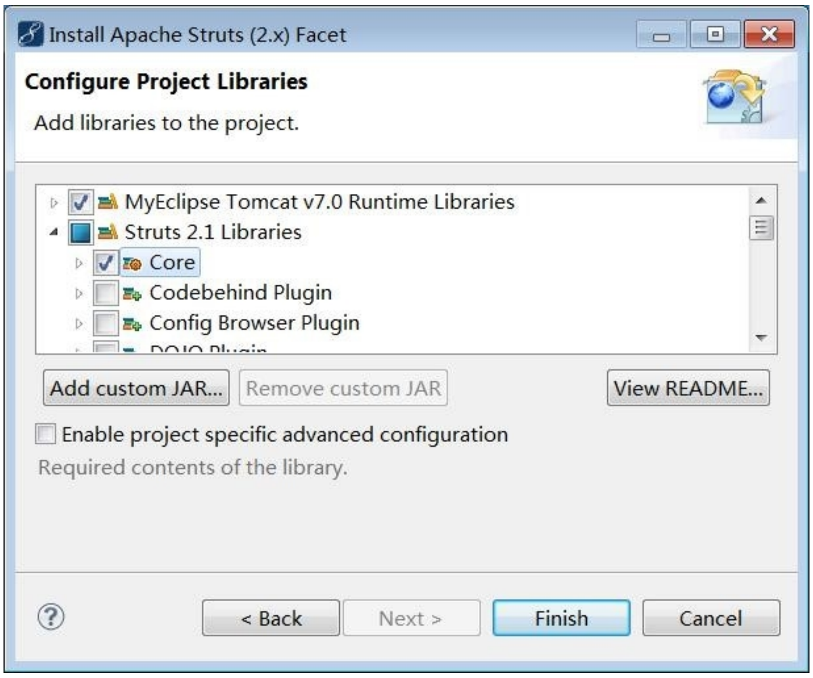
Web Struts 2.x

Configure Web Struts 2.x settings

ActionServlet name: struts2

URL pattern: ○ *.action ○ *.do ● /*

然后点击 Next 按钮显示如下对话框：


Install Apache Struts (2.x) Facet

Configure Project Libraries

Add libraries to the project.

- [✓] MyEclipse Tomcat v7.0 Runtime Libraries
- ■ Struts 2.1 Libraries
  - [✓] Core
  - [ ] Codebehind Plugin
  - [ ] Config Browser Plugin
  - [ ] DOJO Plugin

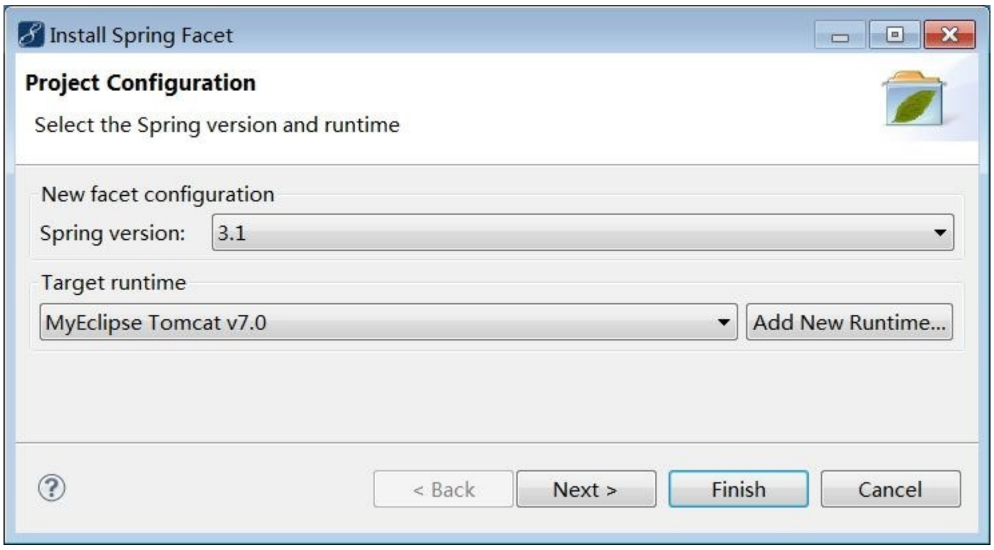
[ ] Enable project specific advanced configuration

Required contents of the library.

选择“Core”和“Spring Plugin”库，然后点击“finish”按钮完成 Struts 的集成。

（3）Spring 集成再在工程名上点击右键，在弹出对话框中选择“MyEclipse”->“Project Facets [Capabilites]”->“Install

Spring Facet”，在弹出对话框中选择 Spring 的版本及服务器：


Install Spring Facet

Project Configuration

Select the Spring version and runtime

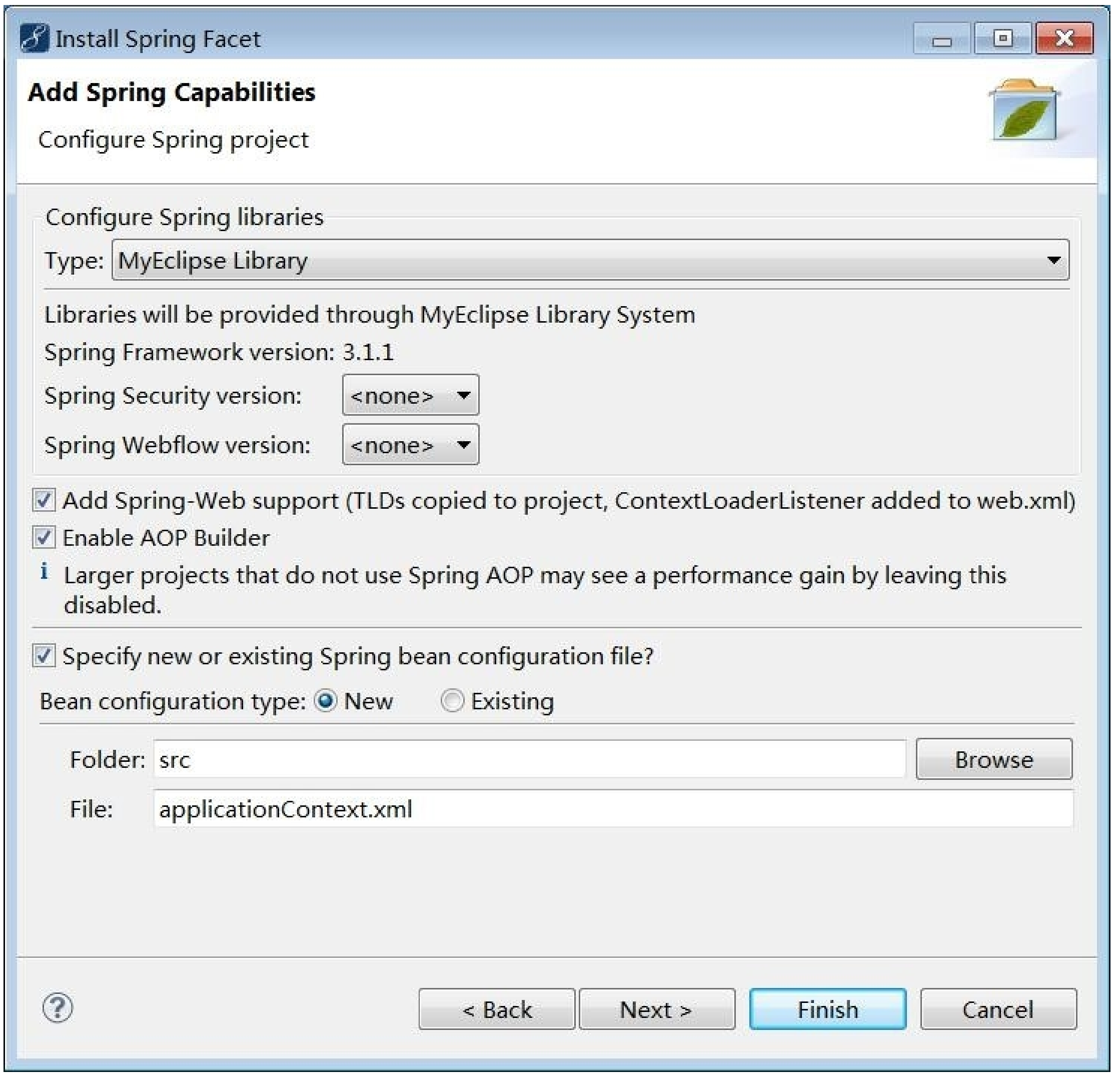
New facet configuration

Spring version: 3.1

Target runtime

MyEclipse Tomcat v7.0

在下一个对话框中，设置 Spring 的配置文件，如下（使用默认值即可）：


Install Spring Facet

Add Spring Capabilities

Configure Spring project

Configure Spring libraries

Type: MyEclipse Library

Libraries will be provided through MyEclipse Library System

Spring Framework version: 3.1.1

Spring Security version: <none>

Spring Webflow version: <none>

☑ Add Spring-Web support (TLDs copied to project, ContextLoaderListener added to web.xml)

☑ Enable AOP Builder

i Larger projects that do not use Spring AOP may see a performance gain by leaving this disabled.

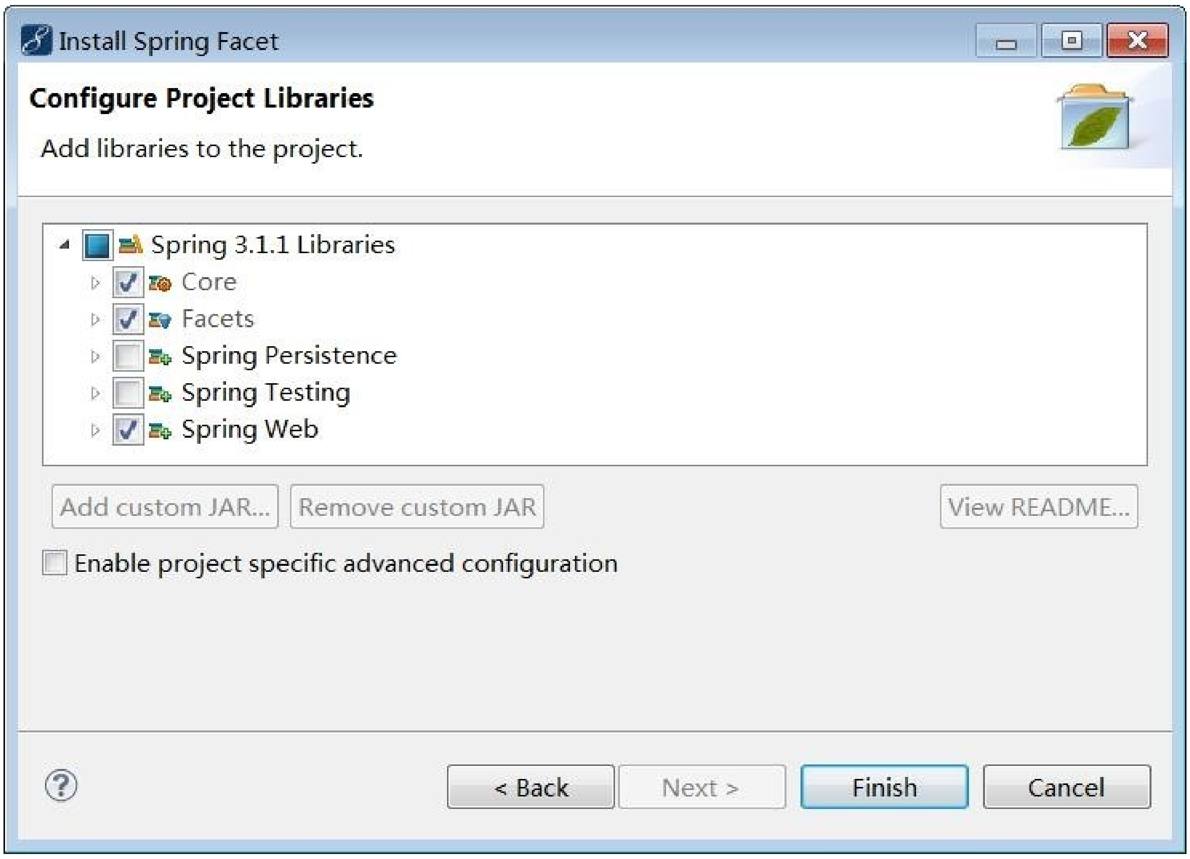
☑ Specify new or existing Spring bean configuration file?

Bean configuration type: [✓ New] Existing

Folder: src

File: applicationContext.xml

点击“Next”，显示对话框如下图，选择“Core”、“Facets”及“Spring Web”，然后点击“Finish”按钮完成 Spring 的集成。


Install Spring Facet

Configure Project Libraries

Add libraries to the project.

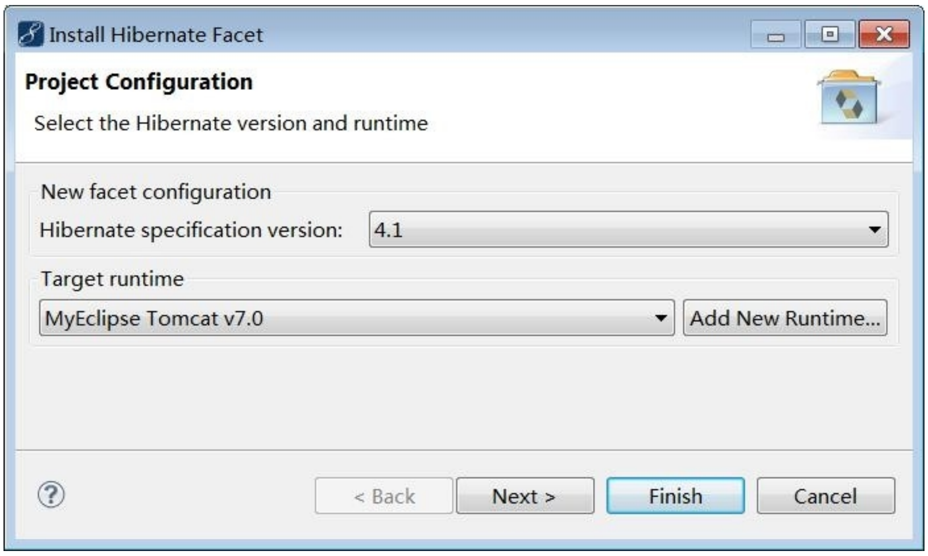
- Spring 3.1.1 Libraries
  - [✓] Core
  - [✓] Facets
  - [ ] Spring Persistence
  - [ ] Spring Testing
  - [✓] Spring Web

[ ] Enable project specific advanced configuration

打开 web.xml 在</web-app>标记之前添加如下内容，使 Spring 生效。

```
<context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:applicationContext.xml</param-value>
   </context-param>
   <!-- 对 Spring 容器进行实例化 -->
   <listener>
      <listener-class>
          org.springframework.web.context.ContextLoaderListener
       </listener-class>
   </listener>
```

（4）Hibernate 集成再在工程名上点击右键，在弹出对话框中选择“MyEclipse” -> “Project Facets[Capapilities]” -> “Install Hibernate Facet”，在弹出对话框中选择 Hibernate 的版本及服务器：


Install Hibernate Facet

Project Configuration

Select the Hibernate version and runtime

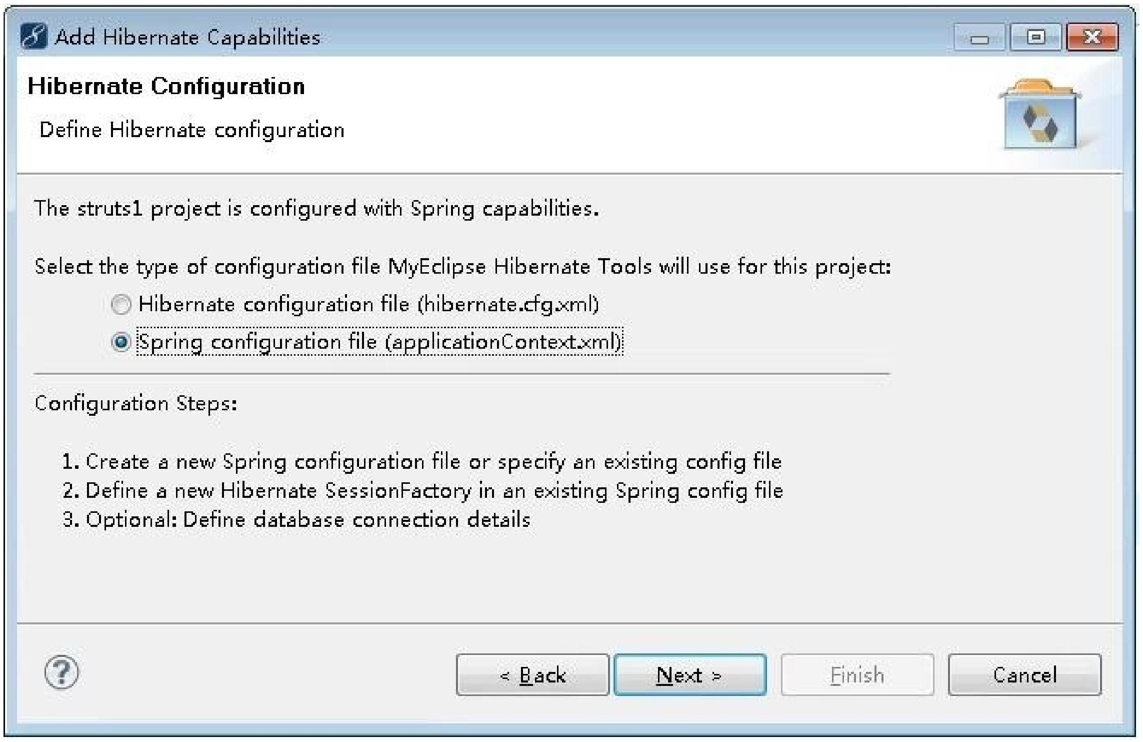
New facet configuration

Hibernate specification version: 4.1

Target runtime

MyEclipse Tomcat v7.0

在下一个窗口中选择如下；


Add Hibernate Capabilities

Hibernate Configuration

Define Hibernate configuration

The struts1 project is configured with Spring capabilities.

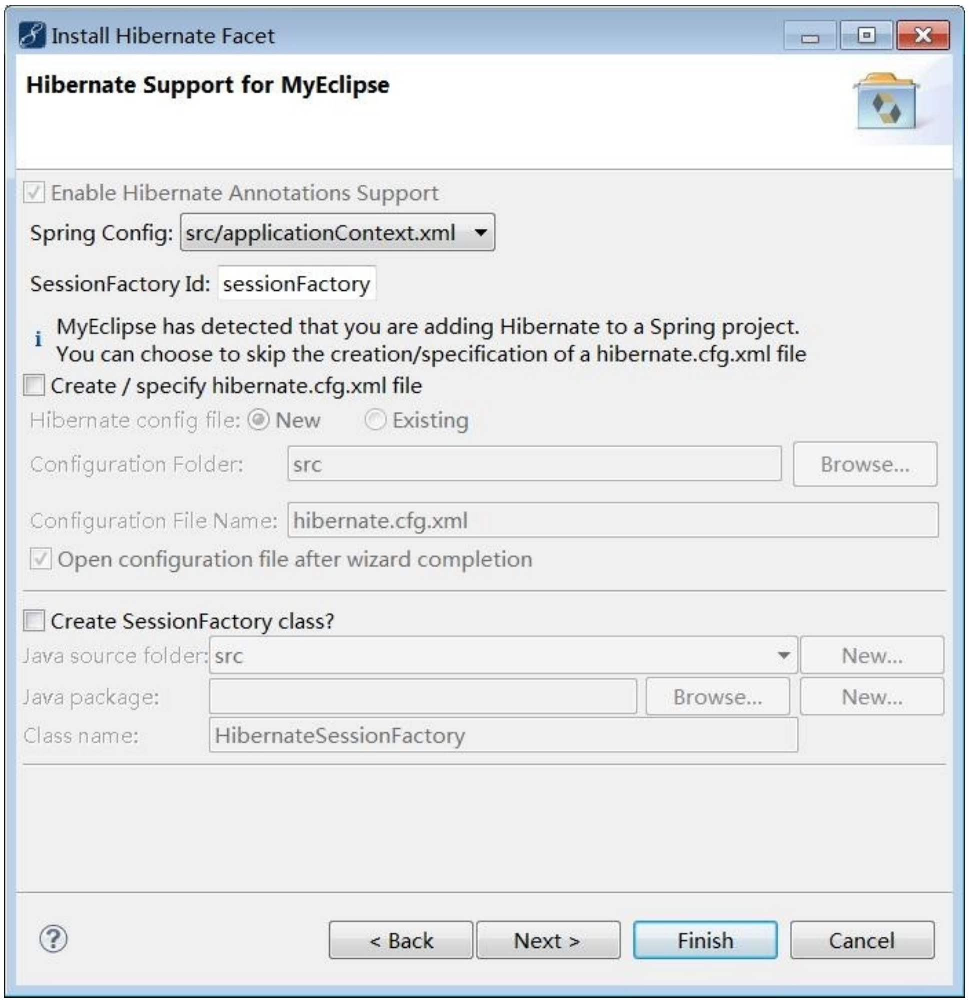
Select the type of configuration file MyEclipse Hibernate Tools will use for this project:

[ ] Hibernate configuration file (hibernate.cfg.xml)
[✓ Spring configuration file (applicationContext.xml)]

Configuration Steps:

1. Create a new Spring configuration file or specify an existing config file
2. Define a new Hibernate SessionFactory in an existing Spring config file
3. Optional: Define database connection details

在下一个窗口中设置 Hibernate 的会话工厂在 Spring 配置文件的 ID，取消创建会话工厂的工具类如下：


Install Hibernate Facet

Hibernate Support for MyEclipse

☑ Enable Hibernate Annotations Support

Spring Config: src/applicationContext.xml

SessionFactory Id: sessionFactory

MyEclipse has detected that you are adding Hibernate to a Spring project.
You can choose to skip the creation/specification of a hibernate.cfg.xml file

☐ Create / specify hibernate.cfg.xml file

Hibernate config file: [✓ New] Existing

Configuration Folder: src

Configuration File Name: hibernate.cfg.xml

☑ Open configuration file after wizard completion

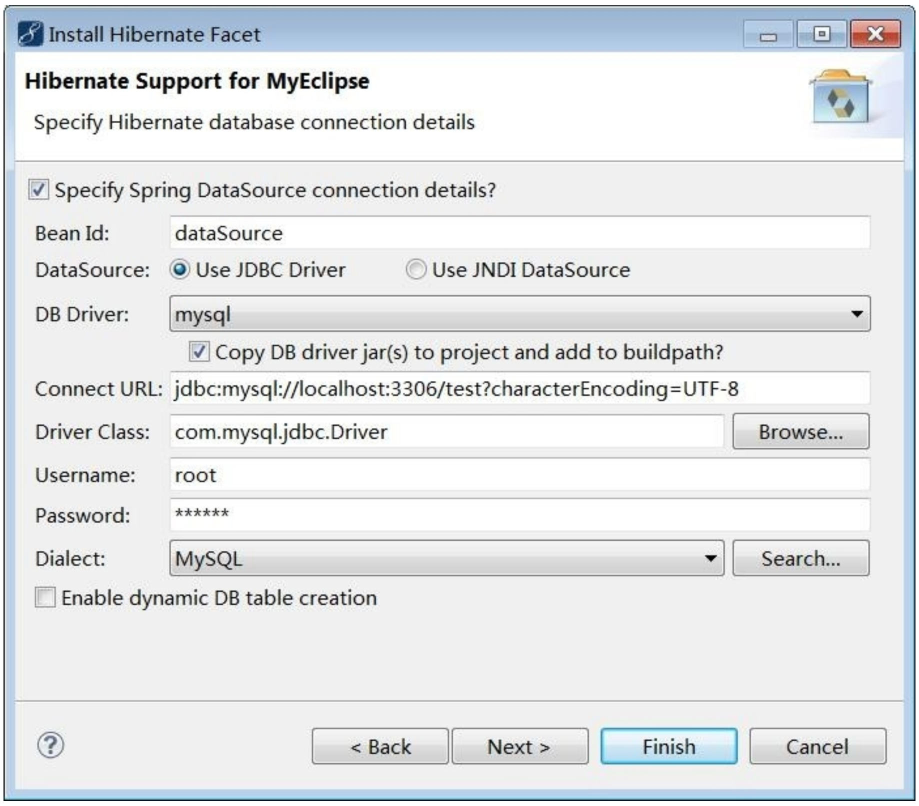
☐ Create SessionFactory class?

Java source folder: src

Java package:

Class name: HibernateSessionFactory

在下一窗口中选择自己的数据库，如图：


Install Hibernate Facet

Hibernate Support for MyEclipse

Specify Hibernate database connection details

☑ Specify Spring DataSource connection details?

Bean Id: dataSource

DataSource: [✓ Use JDBC Driver] Use JNDI DataSource

DB Driver: mysql

☑ Copy DB driver jar(s) to project and add to buildpath?

Connect URL: jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8

Driver Class: com.mysql.jdbc.Driver

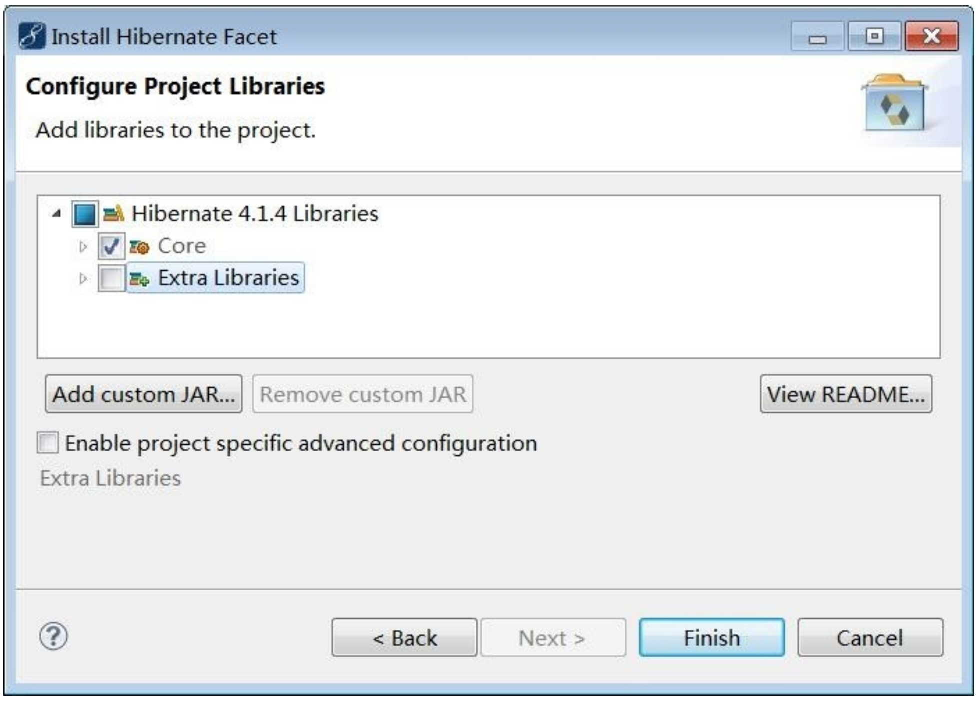
Username: root

Password: ******

Dialect: MySQL

☐ Enable dynamic DB table creation

在下一窗口中，选择 Hibernate 的库，如图。然后点击“Finish”按钮完成 Hibernate 的集成。


Install Hibernate Facet

Configure Project Libraries

Add libraries to the project.

- [■] Hibernate 4.1.4 Libraries
  - [✓] Core
  - [ ] Extra Libraries

[ ] Enable project specific advanced configuration

Extra Libraries

最终自动生成的 Spring 的配置文件——applicationContext.xml，内容为：

```
<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx.xsd"

xmlns:tx="http://www.springframework.org/schema/tx">


<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">

   <!--MyEclipse 2014 会丢失此行，自行添加-->

    <property name="driverClassName" value="com.mysql.jdbc.Driver" />

    <property name="url"

        value="jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8">

    </property>

    <property name="username" value="root"></property>

    <property name="password" value="123456"></property>

</bean>
```

```
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate4.LocalSessionFactoryBean"> <property
    name="dataSource">

        <ref bean="dataSource" />
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.MySQLDialect
            </prop>
        </props>
    </property>
</bean>
<bean id="transactionManager"
    class="org.springframework.orm.hibernate4.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
<tx:annotation-driven transaction-manager="transactionManager" />
</beans>
```

（5）创建包：

com.entity：实体包，存放 Hibernate 的实体类；

com.model.action：Action 包，存放 Struts 的 Action

com.model.dao：Dao 包，存放持久化类（数据库操作）； com.model.service：服务包，存放 service 类，此包中的类一般都是将相关的业务处理封装到一个类中，供 action 调用，该类中再调用 Dao 类完成具体的操作。

（6）修改 Spring 配置，让 Spring 对指定对象进行管理。最后的配置文件内容如下：

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:context="http://www.springframework.org/schema/context"
```

```
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.1.xsd
        "
        xmlns:tx="http://www.springframework.org/schema/tx">
    <!-- 开启 Spring 自动扫描，对指定包中的类进行自动装配 -->
    <context:component-scan base-package="model" />

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
    <!--MyEclipse 2014 会丢失此行，自行添加-->
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url"
value="jdbc:mysql://localhost:3306/test?characterEncoding=UTF-8">
        </property>
        <property name="username" value="root"></property>
        <property name="password" value="123456"></property>
    </bean>
    <bean id="sessionFactory"
        class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
        <property name="dataSource">
            <ref bean="dataSource" />
        </property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">
                    org.hibernate.dialect.MySQLDialect
                </prop>
                <prop key="hibernate.show_sql">
                    true
                </prop>
            </props>
        </property>
        <!-- 开启 Hibernate 实体类扫描 -->
        <property name="packagesToScan">
            <list>
                <value>entity</value>
            </list>
        </property>
    </bean>

</beans>
```

（7）相关代码：

Student.java

```
package com.entity;
@Entity //实体注解
public class Student {
    @Id//主键注解
    @GeneratedValue(strategy=GenerationType.IDENTITY)//主键生成策略
    private int id; private String xh; private String name;
    private String className; private String sex;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getXh() {
        return xh;
    }
    public void setXh(String xh)
        { this.xh = xh;

    }
    public String getName()
    { return name; }
    public void setName(String name)
        { this.name = name;
    } public String getClassName()
    { return className;
    } public void setClassName(String className)
    { this.className = className;
    }
    public String getSex()
    { return sex; }
    public void setSex(String sex)
        { this.sex = sex;
    }
}
```

StudentDao.java

```
package com.model.dao;
@Repository //Dao注解
public class StudentDao {
    @Resource //注入会话工厂
    private SessionFactory sf;

    /**
     * 学生列表
     * @return
```

```
 */
public List<Student> studentList(){
    Session session = sf.openSession();
    Query query = session.createQuery("from Student");
    List list = query.list();
    session.close();
    return list;
}

/**
 * 保存添加/修改
 * @param stu
 */ public void saveStudent(Student
stu){ Session session =
sf.openSession();
session.beginTransaction();
if(stu.getId() ==
0){ session.save(stu);
    }
    else{ session.update(st
    u);
    }
    session.getTransaction().commit();
    session.close();
}

/**
 * 根据 ID 得到一个学生信息
 * @param id
 * @return
```

```
     */ public Student getOneById(int
    id){ Session session =
    sf.openSession();

        Student stu = (Student) session.get(Student.class, id);
        session.close();
        return stu;
    }

    /**
     * 删除
     * @param id
     */
    public void deleteById(int id){
        Session session = sf.openSession();
        Student stu = (Student) session.get(Student.class, id);
        if(stu == null)return; session.beginTransaction();
        session.delete(stu); session.getTransaction().commit();
        session.close();
    }
}
```

StudentService.java

```
package com.model.service;
@Service //注解为 Service
public class StudentService {
    @Resource //注入 dao
    private StudentDao stuDao;

    public List<Student> list(){
        return stuDao.studentList();
    }
    public Student getOneById(int
        id){ return stuDao.getOneById(id);
    }

           public void save(Student
       stu){ stuDao.saveStudent(stu);
    }

    public void delete(int id){
        stuDao.deleteById(id);
    }
}
```

StudentAction.java

```
package com.model.action;
@Controller//注解为控制器(Action)
@Scope("prototype")//注解作用范围
public class StudentAction {
    @Resource //注入 studentService
    private StudentService stuService;
    private List<Student> stuList;
    private int stuid; private Student
    stu;

    public String list(){ stuList =
    stuService.list(); return
    "list"; }

    public String
        add(){ stu = new
        Student(); return
        "edit";
    }

    public String edit(){ stu =
        stuService.getOneById(stuid);
        return "edit";
    }

    public String
        save(){ stuService.save(stu)
        ; return "save";
    }

    public String
        delete(){ stuService.delete(stu
        id); return "save";
    }

    public void setStuList(List<Student> stuList)
        { this.stuList = stuList;
    }

    public List<Student> getStuList()
        { return stuList;
    }

    public void setStuid(int stuid)
        { this.stuid = stuid;
    }
```

```
    public int getStuid() {
        return stuid;
    }

    public void setStu(Student stu) {
        this.stu = stu;
    }

    public Student getStu() {
        return stu;
    }
}
```

Struts.xml

```
<struts>
    <package name="mm" extends="struts-default" namespace="/">
        <action name="stu_*" class="studentAction" method="{1}">
            <result name="list">list.jsp</result>
            <result name="edit">edit.jsp</result>
            <result name="save" type="redirectAction">stu_list</result>
        </action>
    </package>
</struts>
```

视图文件自己创建。