J2EE 应用开发总复习

1. JSP、Ajax、JS 英文全称是什么。

2. Struts 的工作原理是什么？

3．Struts 的 Action 能映射为什么扩展名的文件，映射后的文件怎么调用(使用)。

4．熟悉 Struts Action 的缺省行为函数的的特点，即从访问权限、返回值类型、函数名称、函数参数是什么。

5. 在 Struts 2.X 框架下，Action 类的基类是什么？

6. 在 Struts 2.X 框架下，如何实现多 Action 操作。

7. 在 Struts 2.X 框架下，配置 result 的常用类型有哪些。（熟悉、弄懂课堂演示里的 type 配置为 dispatcher、redirect、redirectAction 和 plainText）

8. 支持 Struts 的 JSTL 包有哪些（包括：struts-html、struts-bean、struts-logic），要熟悉、弄懂课堂演示例子和 PPT 里用到的 struts-html、struts-bean、struts-logic 的常用 tag 操作。

9. struts-html、struts-bean、struts-logic 等是第三方开发提供的扩展 Tag 库，如何使用自定义 tag 库。(不做要求)

10．用 JDBC-ODBC 桥在 J2SE、JSP 页面和 J2EE Web 程序中连接 Access、SQL Server、Oracle、mySQL 的端口是什么，如何操作，即主要步骤。(主要说明驱动程序选择、加载、会话工厂、会话、事务、语句等重点步骤如何操作，以及关闭操作)

11．在 JSP 页面如何和 Bean 进行数据传送，JSP 页面如何和 Action 类进行数据传送。

12. 配置文件是 J2EE Web 项目重要的一类文件，J2EE 项目支持 Struts、Hibernate 架构的配置文件是什么，J2EE Web 项目配置文件是什么，熟悉配置文件里的标记。

在 Struts 的配置文件中 action 的定义所用到的参数和属性。(熟悉课堂上讲到的标记、属性)

13．J2EE Web 项目需要通过部署服务器运行，常用的部署服务器有哪些。

14. Ajax 是一种用 JS 开发的支持页面局部、异步更新的软件包，该技术已应用在 J2EE 项目中的哪些方面。(不做要求)

15. jQuery 是一个 JavaScript 库，实现了常见任务的自动化和复杂任务的简单化，在 J2EE 项目中能实现那些常用的功能。(课件里有)

16．熟悉 jQuery 的变量语法、函数编写、控件操作等功能。

17．jQuery 的选择器有哪些基本类型。

18．jQuery 的子元素过滤选择器包括哪些。

19. JS 是近年来在 Web 项目中编写高深应用的语言，jQuery 是用 JS 语言开发的软件包，用 JS 操作常用控件如何操作。JS 的变量类型有哪些种类。

20. 正则表达式用于验证控件的输入有效性，熟悉用用正则表达式验证输入框内容的有效性。

21．J2EE 项目通过 Web 服务器部署，在 Netbeans IDE 中使用哪些 Web 服务器。

22. Servlet 是运行于服务器端的组件，由 ActionServlet 根据用户请求操作从 Servlet Container 中选取合适的 Servlet 运行，Servlet 常用的方法有哪些，doGet/doPost 的参数类型分别是什么，Servlet 能用于做什么方面的 Web 业务。(不做要求)

23. Hibernate 是通过持久层和实体类实现数据库的 CRUD 操作，叙述如何使用 Hibernate 进行数据库的 CRUD 操作。

24．JSF 是继 Struts 之后推出的又一重要框架，能够方便地实现 View 端的显示，使用 JSF 技术如何操作，JSF 的常用 Tag 有哪些。以 JSF 实现数据库的 CRUD 操作为例说明。(不做要求)

25．MVC、MVC2 是新一代 Web 项目使用的设计模式，该模式的主要内容、主要本质是什么。

26．以课堂上讲过的登陆窗口为例，用纯 JSP 语法、Struts 的 JSTL　tags-html。编写登陆窗口。用 Struts Action 如何实现登陆窗口。

27．用 Hibernate 实现数据库连接 CRUD 操作的主要步骤是什么。

28．在 jQuery 的前端表示页面如何调用其它 JSP 类型文件、Struts 的 Action。

29．jQuery 的过滤选择器有什么功能，过滤选择器有哪些类型。

30．Struts 的执行流程。(课件里有)

31．Struts 的 Action 搜索顺序。(课件里有)

32．Struts 的结果类型。(课件里有)

33．Struts 的输入校验流程。(课件里有)

34．叙述在 JSP 页面以表格形式显示 Stud={ID, Name, Address, Phone}表的主要代码。

35. 注入的依赖属性检查有哪些模式（simple、object、all 和 none 四种模式）。

36. Spring 自动装配可减少指定属性的设置。设置 Bean 元素的 autowire 属性指定 Bean 的自动装配模式，可设置哪些模式(byName、byType、constructor、autodetect 和 no 方式)。

37. Spring 切面可应用哪 5 种类型通知。(不做要求)

38. 切面的常用术语有通知(advice)、切点(pointcut)和连接(join point) (不做要求)

39. 一下配置文件或代码要熟悉

(1)文件名 Employee.hbm.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!--
    Mapping file autogenerated by MyEclipse Persistence Tools
-->
<hibernate-mapping>
    <class name="org.j2ee.ssh.Employee" table="employee"    catalog="test" >
        <id name="id" type="long">
            <column name="id" />
            <generator class="increment" />
        </id>
        <property name="name" type="string">
            <column name="name" length="20" />
        </property>
        <property name="address" type="string">
            <column name="address" length="30" />
        </property>
        <property name="phone" type="string">
            <column name="phone" length="10" />
        </property>
  </class>

</hibernate-mapping>
```

(2) hibernate.cfg.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
            "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
            "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```

```
<!-- Generated by MyEclipse Hibernate Tools.                   -->
<hibernate-configuration>

<session-factory>
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/test</property>
        <property name="connection.username">root</property>
        <property name="connection.password">niitniit</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="myeclipse.connection.profile">mySQL</property>
        <mapping resource="org/j2ee/ssh/Employee.hbm.xml" />
</session-factory>

</hibernate-configuration>
```

(3)课堂例子里用到的 struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
"http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
    <package name="default" extends="struts-default" namespace="/">
        <action name="hql" class="org.j2ee.ssh.HQLOprAction">
            <result name="success">/dspdata.jsp </result>
        </action>
        <action name="recordselect" class="org.j2ee.ssh.HQLOprAction"
            method="GetRecordSelect">
            <result name="success">/DspRecordData.jsp </result>
        </action>
        <action name="saverecord" class="org.j2ee.ssh.HQLOprAction"
            method="SaveRecord">
```

```
        <result name="success">/DspRecordData.jsp </result>
    </action>
    <action name="deleterecord" class="org.j2ee.ssh.HQLOprAction"
        method="DeleteRecord">
        <result name="success">/DspRecordData.jsp </result>
    </action>
    <action name="updaterecord" class="org.j2ee.ssh.HQLOprAction"
        method="UpdateRecord">
        <result name="success">/DspRecordData.jsp </result>
    </action>
    <action name="hqlfieldselect" class="org.j2ee.ssh.HQLOprAction"
        method="HQLFieldSelect">
        <result name="success">/dspfieldata.jsp </result>
    </action>
    <action name="hqlparaselect" class="org.j2ee.ssh.HQLOprAction"
        method="HQLParaSelect">
        <result name="success">/dspdata.jsp </result>
    </action>
    <action name="sqlselect" class="org.j2ee.ssh.HQLOprAction"
        method="SQLSelect">
        <result name="success">/dspdata.jsp </result>
    </action>
    <action name="sqlinsert" class="org.j2ee.ssh.HQLOprAction"
        method="SQLInsertData">
        <result name="success">/dspindata.jsp </result>
    </action>
    <action name="sqldelete" class="org.j2ee.ssh.HQLOprAction"
        method="SQLDeleteData">
        <result name="success">/dspdata.jsp</result>
    </action>
```

```
            <action name="sqlupdate" class="org.j2ee.ssh.HQLOprAction"
                  method="SQLUpdateData">
                  <result name="success">/dspdata.jsp</result>
            </action>
            <action name="QBEAddCriteria" class="org.j2ee.ssh.HQLOprAction"
                  method="QBCAddCritedia">
                  <result name="success">/dspdata.jsp</result>
            </action>
      </package>
</struts>
```

(4)HibernateSessionFactory.java

```
package org.j2ee.ssh;

//import org.hibernate.HibernateException;
//import org.hibernate.Session;
//import org.hibernate.cfg.Configuration;
//import org.hibernate.service.ServiceRegistry;
////import org.hibernate.service.ServiceRegistryBuilder;
//
///**
// * Configures and provides access to Hibernate sessions, tied to the
// * current thread of execution.    Follows the Thread Local Session
// * pattern, see {@link http://hibernate.org/42.html }.
// */
//public class HibernateSessionFactory {
//
//       /**
//         * Location of hibernate.cfg.xml file.
//         * Location should be on the classpath as Hibernate uses
```

```
//         * #resourceAsStream style lookup for its configuration file.
//         * The default classpath location of the hibernate config file is
//         * in the default package. Use #setConfigFile() to update
//         * the location of the configuration file for the current session.
//         */
//    private static final ThreadLocal<Session> threadLocal = new ThreadLocal<Session>();
//      private static org.hibernate.SessionFactory sessionFactory;
//
//      private static Configuration configuration = new Configuration();
//     // private static ServiceRegistry serviceRegistry;
//
//    static {
//          try {
//                configuration.configure();
//                //serviceRegistry                                =                                new
ServiceRegistryBuilder().applySettings(configuration.getProperties()).buildServiceRegistry();
//                sessionFactory = new Configuration().configure().buildSessionFactory();
//          } catch (Exception e) {
//                System.err.println("%%%% Error Creating SessionFactory %%%%");
//                e.printStackTrace();
//          }
//      }
//      HibernateSessionFactory() {
//      }
//
//    /**
//         * Returns the ThreadLocal Session instance.    Lazy initialize
//         * the <code>SessionFactory</code> if needed.
//         *
//         *    @return Session
```

```
//         *    @throws HibernateException
//         */
//        public static Session getSession() throws HibernateException {
//            Session session = (Session) threadLocal.get();
//
//            if (session == null || !session.isOpen()) {
//                if (sessionFactory == null) {
//                    rebuildSessionFactory();
//                }
//                session = (sessionFactory != null) ? sessionFactory.openSession()
//                        : null;
//                threadLocal.set(session);
//            }
//
//            return session;
//        }
//
//    /**
//         *    Rebuild hibernate session factory
//         *
//         */
//    public static void rebuildSessionFactory() {
//            try {
//                configuration.configure();
//                //serviceRegistry                                =                                new
ServiceRegistryBuilder().applySettings(configuration.getProperties()).buildServiceRegistry();
//                sessionFactory = new Configuration().configure().buildSessionFactory();
//            } catch (Exception e) {
//                System.err.println("%%%% Error Creating SessionFactory %%%%");
//                e.printStackTrace();
```

```
//          }
//    }
//
//    /**
//        *    Close the single hibernate session instance.
//        *
//        *    @throws HibernateException
//        */
//       public static void closeSession() throws HibernateException {
//            Session session = (Session) threadLocal.get();
//            threadLocal.set(null);
//
//            if (session != null) {
//                  session.close();
//            }
//      }
//
//    /**
//        *    return session factory
//        *
//        */
//    public static org.hibernate.SessionFactory getSessionFactory() {
//          return sessionFactory;
//    }
//    /**
//        *    return hibernate configuration
//        *
//        */
//    public static Configuration getConfiguration() {
//          return configuration;
```

```
//    }
//
//}


import java.io.Serializable;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;



public class HibernateSessionFactory {



        private final Configuration CONFIG;

        private final SessionFactory FACTORY;
        private static HibernateSessionFactory instance;



        public HibernateSessionFactory(){

                //加载 XML 配置文件

                CONFIG = new Configuration().configure();
```

```
        //构建工厂

        FACTORY = CONFIG.buildSessionFactory();

    }

    public static HibernateSessionFactory getInstance(){
        if(instance==null)
            instance = new HibernateSessionFactory();
        return instance;
    }

    //从工厂中获得 session 对象

    public Session getCurrentSession(){

        return FACTORY.openSession();

    }

}
```

(5) HQLOprAction.java

只摘选了部分代码。熟悉操作对象是对象型的操作方法(比如，Employee，比如List<Employee>类型)，数据解析方法；熟悉数据是 List<Object[]>类型。

```
    // HQL statement select 1
    public String execute() throws Exception
    {
```

```
//            sessionFactory =    (HibernateSessionFactory)HibernateSessionFactory.getS

//            if(sessionFactory == null)
//                  sessionFactory = new HibernateSessionFactory();

            session = HibernateSessionFactory.getInstance().getCurrentSession();

            String hql = "from Employee";

            System.out.println(session==null?"111":"222");

            Query query = session.createQuery(hql);

            this.dataObject = (List<Employee>)query.list();

            for(Employee edo : this.dataObject)
            {
                  System.out.println(Long.toString(edo.getId()) + "       " + edo.getName() +"     " +
edo.getAddress() +"    " + edo.getPhone());
            }

            session.close();

            return SUCCESS;
      }
      // get 查询
      public String GetRecordSelect() throws Exception
      {
```

```
            Session session =    HibernateSessionFactory.getInstance().getCurrentSession();

            Transaction tx = session.beginTransaction();

            inObject = (Employee) session.get(Employee.class, new Long(1));

            tx.commit();

            System.out.println(Long.toString(inObject.getId()) + "      " + inObject.getName() +"    "
+ inObject.getAddress() +"    " + inObject.getPhone());

            session.close();

            return SUCCESS;
        }

        // 记录保存
        public String SaveRecord() throws Exception
        {

            Session session =    HibernateSessionFactory.getInstance().getCurrentSession();

            Transaction tx = session.beginTransaction();

            //this.inObject = new Employee(10, "张三", "南工大", "1386667777");
            this.inObject = new Employee();
            this.inObject.setId(10);
            this.inObject.setName("张三");
            this.inObject.setAddress("南工大");
            this.inObject.setPhone("13866677");
```

```
            session.save(this.inObject);

            tx.commit();

            System.out.println(Long.toString(inObject.getId()) + "       " + inObject.getName() +"    "
+ inObject.getAddress() +"    " + inObject.getPhone());

            return SUCCESS;
        }
        //Update
        public String UpdateRecord() throws Exception
        {

            Session session =    HibernateSessionFactory.getInstance().getCurrentSession();

            Transaction tx = session.beginTransaction();

            this.inObject = (Employee) session.get(Employee.class, new Long(2));

            this.inObject.setName("张飞");

            session.update(this.inObject);

            tx.commit();

            System.out.println(Long.toString(inObject.getId()) + "       " + inObject.getName() +"    "
+ inObject.getAddress() +"    " + inObject.getPhone());

            return SUCCESS;
```

```
	}

	// delete
	//Update
	public String DeleteRecord() throws Exception
	{
		Session session =    HibernateSessionFactory.getInstance().getCurrentSession();

		Transaction tx = session.beginTransaction();

		this.inObject = (Employee) session.get(Employee.class, new Integer(2));

		session.delete(this.inObject);

		tx.commit();

		System.out.println(Long.toString(inObject.getId()) + "       " + inObject.getName() +"     "
+ inObject.getAddress() +"    " + inObject.getPhone());

		return SUCCESS;
	}

	// HQL statement select 2
	public String HQLFieldSelect() throws Exception
	{
		Session session =    HibernateSessionFactory.getInstance().getCurrentSession();

		String hql = "select id, name, address, phone from Employee";

		Query query = session.createQuery(hql);
```

```
            //List<Employee> datalist = (List<Employee>)query.list();

            this.fieldObject = (List<Object[]>)query.list();

            for(Object[] edo : this.fieldObject)
            {
                System.out.println(edo[0].toString() + "      " + (String)edo[1] +"    " + (String)edo[2]
+"    " + (String)edo[3]);
            }

            session.close();

            return SUCCESS;
        }
```

（6）登陆窗口文件

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>
```

```
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->
</head>

<body>
    <form action = "doLogin.jsp">
        <h2 align="center">用户登录</h2>
        <table align="center">
            <tr>
                <td align="right">用户名:</td>
                <td align="left"><input type="text" name="userName"></td>
            </tr>
            <tr>
                <td align="right">密  码:</td>
                <td align="left"><input type="password" name="userPass">
                </td>
            </tr>
            <tr>
                <td align="center"><input type="submit" value="登录"></td>
                <td align="center"><input type="reset" value="重置"></td>
            </tr>
        </table>
    </form>
```

```
    </body>
</html>
```