Java Web 复习要点

Author @CasterWx

Contact antzuhl1998@gmail.com 1325200471@qq.com

目录

第一章 JavaWeb开发快速入门

1.常见Web开发技术

表现层：前端开发技术，例如HTML 、CSS 、JavaScript 、DOM(Document Object Model) 、ActiveX、VBScript、Applet、JSP
控制层：Servlet 、Struts的Action等技术。
业务逻辑层：JavaBean和EJB((Enterprise JavaBean)等技术
持久层：JDBC、Hibernate、MyBatis

2.容器的作用（Tomcat是什么）

Web服务器是，只有将开发的Web项目放置到该容器运行及发布Web应用的容器中，才能使网络中的所有用户通过浏览器进行访问。安装Tomcat是一个JSP/Servlet容器。

3.Tomcat默认端口是8080，MySQL默认端口是3306。

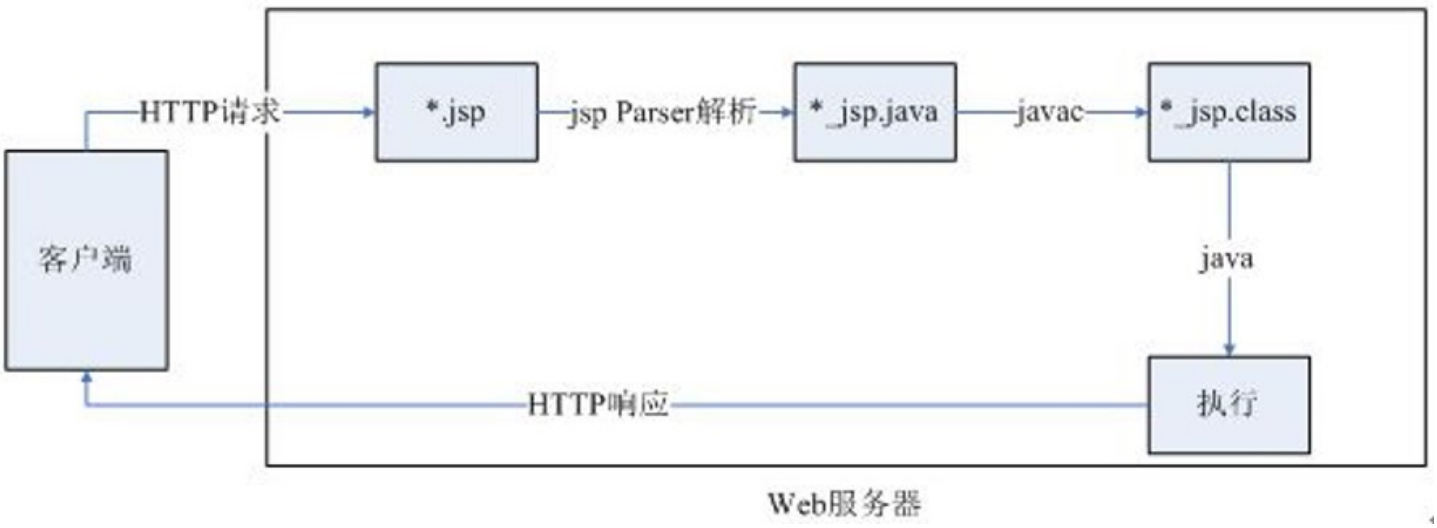
4.web应用的发布目录： webapps

5.web应用第三方库文件存访目录： WEB-INF下lib

6.JSP运行机制

当客户端浏览器向服务器发出请求访问一个JSP页面时，服务器根据该请求加载相应的JSP页面，并对该页面进行编译，然后执行。

- 当一个JSP页面第一次被请求时，容器首先会把JSP页面转换成Servlet。在转换时，所有HTML标签将被包含在println()语句中，所有JSP元素将会被转换成Java代码。
- 在转换的过程中，若JSP页面存在语法错误，转换会被终止，并向服务器和客户端输出错误信息。如果转换成功，转换后的Servlet会被编译成相应的class文件。 JSP的本质就是Servlet。
- 在调用Servlet时，首先执行_jspInit()方法，然后调用_jspService ()方法处理客户端的请求。对客户端发送的每一个请求，JSP容器都会创建一个新的线程来处理。如果有多个客户端同时请求该JSP文件，JSP容器会为每个客户端请求创建对应一个线程。
- 如果JSP文件被修改，服务器将根据设置决定是否对该文件进行重新编译。如果重新编译，内存中的Servlet会被新的编译结果取代。
- Servlet被处理完毕以后，调用_jspDestroy()方法结束它的生命周期，同时被JVM（Java虚拟机）的垃圾回收器回收。



第二章 Web前端技术

1.Html中标签的作用 [理解即可:填空可能会考]

标题标签<hn>
段落标签<p>
段内换行标签
水平线段标签<hr>
原样显示文字标签<pre>
特殊文字样式标签
    * b标记设置文字为粗体
    * i标记为斜体
    * sub标记和sup标记使文字成为下标或上标
表格：
    <table>：定义表格；
    <tr>：定义表格中的行；
    <td>：定义表格中某行中的单元格；
    <th>：定义表格中的表头单元格；
    <thead>：定义表格中的表头内容；
    <tfoot>：定义表格中的表注内容；
    <tbody>：定义表格中的主体内容。
表单<form>
输入域标签<input>
下拉列表框<select>
下拉列表框中的选项<option>
多行文本框<textarea>

2.Css的作用 [理解即可]

- CSS是层叠样式表的简称，是一系列格式设置规则，它们控制Web页面内容的外观。使用CSS可以非常灵活并更好地控制具体的页面外观，从精确的布局定位到特定的字体和样式。

3.Javascript的作用

- JavaScript是一种基于浏览器运行的脚本语言，无须编译即可在浏览器中直接运行。JavaScript主要用于浏览器端的人机交互。JavaScript可以直接嵌入到网页中，也可以单独创建一个扩展名为“.js”文本文件编写JavaScript函数。JavaScript代码基于JavaScript的事件响应执行，即当一个JavaScript的函数响应动作发生时，浏览器就开始执行相应的JavaScript代码。

第三章 JSP语法基础

1.JSP注释的语法格式

```
<%-- 此处为注释 --%>
```

2.JSP声明 选择题可能会问[下面哪一种是正确的jsp声明?]

- 使用标记符： <%! %>

1.声明变量

```
<%!
    int i = 1 ;
    CloseableHttpClient hc =  HttpClients.createDefault() ;
%>
```

2.声明方法

```
<%!
    int add(int a,int b){
        return a+b ;
    }
%>
```

3.JSP中的Java表达式

- 使用标记符： `<%= %>` 后面不用加 分号;

4.JSP中的Java程序段

- 使用标记符： `<% %>`

5.JSP指令

- page指令的作用： 设置JSP页面的属性
  - language : jsp脚本语言名称，目前只支持Java
  - pageEncoding : jsp字符编码，默认ios-8859-1
  - import : 导入所需的Java Api
- include指令的作用： 页面包含指令

6.JSP动作 六个标记的作用

`<jsp:include>`、`<jsp:forward>`、`<jsp:param>`
`<jsp:useBean>`、`<jsp:setProperty>`、`<jsp:getProperty>`

后三个是和JavaBean相关的

- `<jsp:include>` 将一个指定的页面包含到使用此动作标记的JSP页面中
- `<jsp:param>` 用来传递参数不能单独使用，一般嵌套在 `<jsp:include>` 、 `<jsp:forward>` 等动作标记内，用于向这些动作标记传递参数。

```
<jsp:param name="参数名称" value="参数值" />
```

- `<jsp:forward>` 页面重定向，即跳转至page属性指定的页面
- `<jsp:useBean>`
- `<jsp:getProperty>` 获取JavaBean中指定属性的值
- `<jsp:setProperty>` 设置JavaBean属性的值

7.九个内置对象及其作用 要会背，名字与作用

- request：表示HTTP协议的请求，提供对请求数据的访问，JSP页面可在请求范围内共享数据。
- response：表示HTTP协议的响应，提供了访问响应报文的相关方法。
- page：代表JSP页面对应的Servlet实例。
- pageContext：表示JSP页面本身的上下文，它提供了一组方法用于管理具有不同作用域的属性。
- session：表示HTTP协议的会话，可以共享服务器与浏览器之间会话数据，一旦关闭了浏览器，会话数据将自动消毁。
- application：代表应用程序上下文，允许JSP页面与同一应用程序中的Web组件共享数据。
- out：提供对输出流的访问。
- config：提供了一组方法可访问Web应用程序的配置文件web.xml。
- exception：表示异常对象，该对象含有特定JSP异常处理页面访问的异常信息。

8.request对象中的方法 程序填空

- getAttribute() 获得名称为name的属性值，若不存在则返回null
- setAttribute(String name,Object obj) 设置一个名称为name的参数，并且其值为obj
- getCharacterEncoding() 返回request请求体的字符编码
- getParameter(String name) 获得指定参数name的参数值

9.response

- void sendRedirect(String path) 重定向客户的请求到指定页面

10.session

- getAttribute()
- setAttribute(String name,Object obj)

11.application

12.web.xml的作用：配置属性信息

考试会考servlet在web.xml中怎么配置？

```
<servlet>
    <servlet-name>myslt</servlet-name>
    <servlet-class>该servlet具体的实现类</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>myslt</servlet-name>
    <url-pattern>/myslt</url-pattern>
</servlet-mapping>
```

第四章 JDBC技术

1. JDBC访问数据库的五个步骤

（1）加载JDBC驱动程序

```
Class.forName("com.mysql.jdbc.Driver");
```

（2）建立数据库连接；

```
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/userdb", root, passwd);
```

（3）创建操作数据库SQL的Statement/PreparedStatement对象；

```
Statement statement = connection. createStatement ();

PreparedStatement ps=con.prepareStatement("insert into wy_table values (?,?)");
ps.setInt(1,6);
ps.setInt(2,"demo");
```

（4）执行语句并分析执行结果；

```
execute(String sql) 执行指定的SQL语句。如果SQL语句返回结果，此方法返回true，否则返回false
executeQuery(String sql) 执行查询类型(select)的SQL语句，此方法返回查询所获取的结果集ResultSet对象
executeUpdate(String sql) 执行SQL语句中DML类型(insert、update、delete)的SQL语句，返回更新所影响的行数；执行
```

（5）释放资源。

2. 操作数据库SQL的对象 代码填空会考

- Statement
- PreparedStatement

两者的区别:Statement执行静态SQL语句,PreparedStatement执行动态SQL语句

第五章 JavaBean

1. JavaBean相关的三个标记以及作用

```
jsp:useBean指令
jsp:getProperty
jsp:setProperty
```

第六章 Servlet

1.Servlet生命周期中三个关键的方法及其作用

init()、service()和destroy()

- Servlet容器完成加载Servlet类和实例化一个Servlet对象后，init()方法完成初始化工作，该方法由Web容器调用完成；service()方法处理客户端请求，并返回响应结果；destroy()方法在Web服务器卸载Servlet之前被调用，释放一些资源。

2.创建一个Servlet

- 继承HttpServlet
- 继承GenericServlet
- web.xml

3.创建一个Filter

- 继承一个Filter接口
- 实现doFilter方法

第七章 MVC与DAO模式

1.MVC每一部分的含义，作用与用什么技术实现

M表示模型 业务逻辑层 数据业务处理功能

V代表视图 视图层 数据显示

C即控制器 控制层 流程控制

Model模型：用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。模型有对数据直接访问的权力。

View视图：视图层能够实现数据有目的的显示。在视图中一般没有程序上的逻辑，它从模型那里获得数据并指定这些数据如何表现。当模型变化时，视图负责维持数据表现的一致性，视图同时负责将用户需求通知给控制器。

Controller控制器：控制器相当于调度者，用于控制应用程序的流程。它处理事件并做出响应。

MVC所需技术参考第一章

2.请求转发与页面重定向

页面跳转总结：

1）<jsp:forward page="xxx.jsp"/>

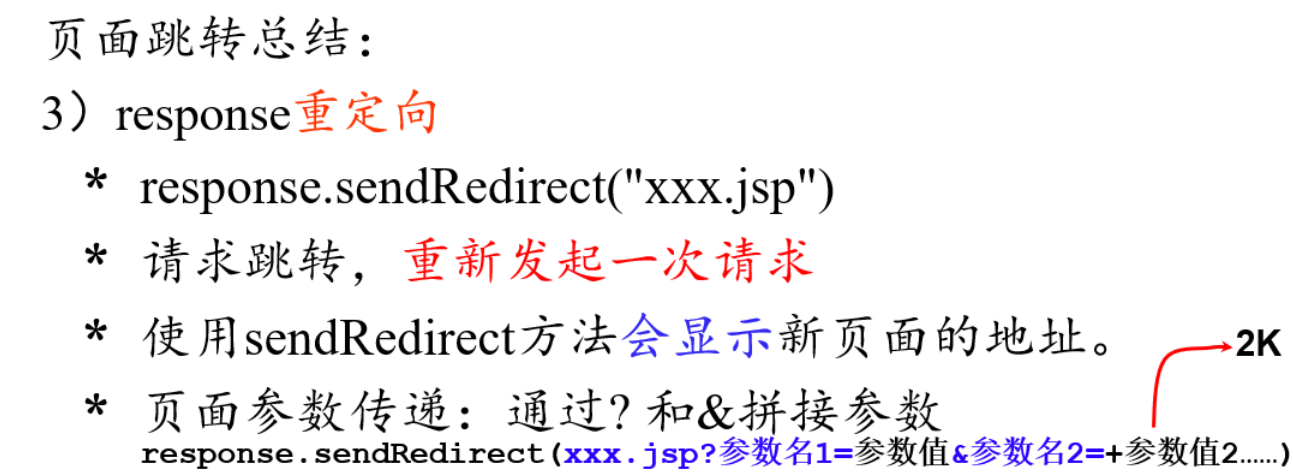
* 请求转发，存储在request中的信息被保留并带至目标页面
* 转到新的页面后，地址栏中不会显示新页面的地址。
* 参数传递：<jsp:param name="" value=""/>
* 主要使用在JSP页面中

- 页面跳转总结

2）RequestDispatcher对象实现跳转

* RequestDispatcher rd = request.getRequestDispatcher("目的页面url");
  rd.forward(request,response);
* 请求转发，存储在request中的信息被保留并带至目标页面
* 转到新的页面后，地址栏中不会显示新页面的地址。
* 页面参数传递：setAttribute(String name, Object object)
  Object getAttribute(String name)

注意：以上两种方法跳转前后属于同一次请求！



例子:`response.sendRedirect("loginResult.jsp?name=zhangsan&password=123");`

参数传递拼接变量的例子：

```
String name = request.getParameter("name");
String password = request.getParameter("password");
response.sendRedirect("loginResult.jsp?name="+name+"&password="+password);
```

第十章 Struct2技术框架

Struts2核心配置文件： `struts.xml` 放在src目录下

1.struts配置方法

必考点:给定命名空间和Action的名字，写出访问这个Action的url是什么。也可能会反过来考。

```
<struts>
<package name="com" namespace="/user" extends="struts-default">
        <action name="hello" class="com.struts2.hello.HelloAction">
                <result name="success">/hello.jsp</result>
        </action>
</package>
</struts>
```

action中class不指定默认是ActionSupport类。

method不指定默认是execute方法。

根据method指定的方法返回一个字符串，用来指定响应视图，Result就是用来处理逻辑视图与物理视图的关系。

2.Action实现方式

- 普通Java类

- Action接口
- 继承ActionSupport类

3.动态方法调用 使用!

4.在Action中获取ServletApi

- loc方式
  - 实现 `ServletRequestAware`，`ServletResponseAware`，`SessionAware` 接口
- 非loc方式

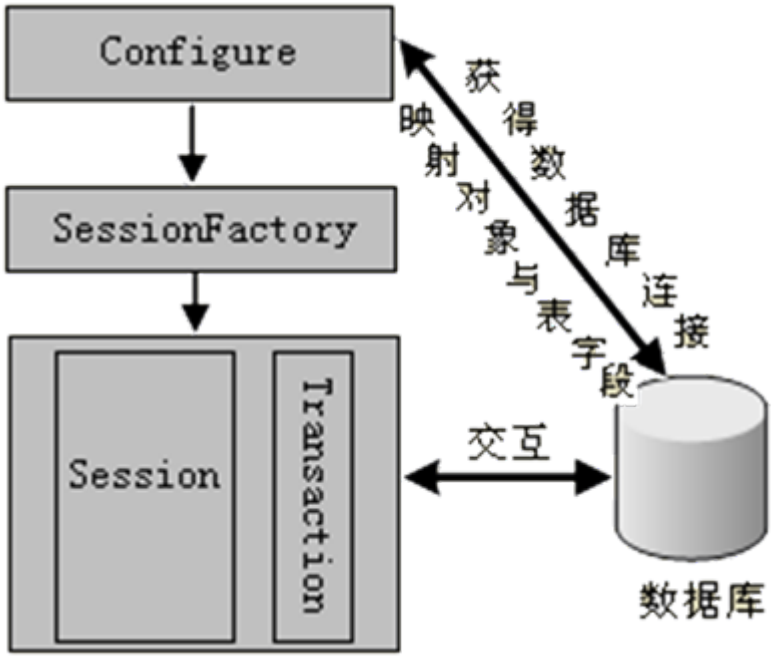
```
HttpServletRequest request=ServletActionContext.getRequest();
HttpServletResponse response=ServletActionContext.getResponse();
```

第十一章 Hibernate技术框架

1.什么是ORM?

- ORM是解决在面向对象的编程中，程序设计语言与关系型数据库发展不匹配而提出来的一种中间层解决方案。通过使用描述对象和数据库之间映射的元数据，将应用程序中的对象自动持久化到关系型数据库中，本质上是将数据从一种形式转换成另外一种形式。帮助开发人员完成面向对象的程序设计语言到关系型数据库的映射，从而实现在项目中既保持以一种完全地面向对象的思想设计与开发应用程序和持久化数据库，又能利用关系型数据库的技术优势。

2.Hibernate的工作过程



（1）读取并解析配置文件：这是使用Hibernate框架的入口，由Configure类来创建。

（2）读取并解析映射信息：调用Configure中的buildSessionFactory()方法来实现，同时创建SessionFactory。

（3）开启Sesssion：调用SessionFactory的openSession()方法来实现。

（4）创建事务管理对象Transaction：调用Session对象的beginTransaction()来实现。

（5）数据交互操作：调用Session对象的各种操纵数据库的方法来处理数据。例如增、删、改、查。

（6）提交事务：完成了对数据库的操作后应该提交事务，完成一次事务处理。

（7）关闭Session：结束了对数据库的访问以后，应该立即关闭Session对象，释放其占用的内存。

（8）关闭SessionFactory：完成了全部的数据库操作后关闭SessionFacory对象。

```
Configuration configuration = new Configuration() ;
configuration.configure("hibernate.cfg.xml") ; //"hibernate.cfg.xml"
SessionFactory sessionFactory = configuration.buildSessionFactory() ;
Session session = sessionFactory.openSession() ;
Transaction transaction = session.getTransaction() ;
transaction.begin(); //开启事务
session.save(Object);
transaction.commit(); //提交事务
session.close() ;
hibernateSession.closeSession();
```

3.hibernate默认配置文件 `hibernate.cfg.xml`

4.映射文件 `*.hbm.xml`

5.查询结果

```
Query query = session.createQuery(hql) ;
List<Comment> list = query.list() ;
```

6.HQL中不使用表名，使用持久化的类名，原生态的SQL使用createSQLQuery执行。