Web应用

B/S：Browser Server，浏览器服务。
C/S：Client Server，客户端服务。

Web应用是通过浏览器访问的应用，应用一般发布在Web服务器上。常用的Web服务器有：Apache Tomcat、IIS（微软）、Nginx等。

Apache Tomcat官网：https://tomcat.apache.org

Servlet

一、什么是Servlet

Servlet（Server applet）是运行在服务器上的小程序。

`javax.servlet.Servlet`是一个接口，定义了Java类被浏览器访问（Tomcat识别）的规则。

自定义的业务Java类想要被Tomcat解析，就必须要继承Servlet接口。

二、Servlet入门配置

（一）Servlet 简单配置

1、创建一个类，实现Servlet接口，重写Servlet接口中的方法；

```
public class ServletDemo1 implements Servlet {
    // 重写的方法省略
}
```

2、配置XML文件：web.xml

```
<!-- 配置Servlet-->
<!-- 主要配置后台Java类的映射关系，Servlet名字和对应的Java类，类要写全路径（包名+类名） -->
<servlet>
    <servlet-name>demo1</servlet-name>
    <servlet-class>com.soft.servlet.ServletDemo1</servlet-class>
</servlet>

<!--
    主要配置浏览器和Servlet的映射关系，Servlet名字和浏览器访问地址
```

```
    url-pattern:
        1、/xxx
        2、/xxx/xxx
        3、xxx.do
        4、*（优先级最低）
-->
<servlet-mapping>
    <servlet-name>demo1</servlet-name>
    <url-pattern>/demo1</url-pattern>
</servlet-mapping>
```

3、通过浏览器请求路径：协议名://IP地址:Tomcat端口号/[虚拟路径]/请求路径

协议名：http
IP地址：localhost、127.0.0.1、真实IP
Tomcat端口号：8080（可以自行修改，但是不要占用其他端口号）
虚拟路劲：可以理解为工程名，可以写可以不写
请求路径：url-pattern对应的值

`http://localhost:8080/demo1`

（二）Servlet 原理（流程）

1、浏览器发送请求路径，去匹配web.xml中配置的url-pattern，找到对应的servlet-name。
2、通过servlet-name找到名字对应的Java类。
3、执行Java类中对应的方法。

（三）Servlet 生命周期

1、当浏览器第一请求服务器时，执行init初始方法，然后执行service方法。也可以设置在服务器启动时加载执行init方法。
2、以后再请求服务器直接执行service方法。
3、根据页面请求的方式（get/post）,去执行doGet/doPost方法。
4、当服务器正常关闭时执行destroy方法释放资源。

（四）Servlet 注解配置

Servlet3.0之后新增了注解的方式配置Servlet。步骤如下：

1、创建一个类，实现Servlet接口，重写Servlet接口中的方法；
2、在类的上面添加注解@WebServlet

```
// @WebServlet(value = "/demo2", loadOnStartup = 1)
@WebServlet("/demo2")
public class ServletDemo2 implements Servlet {
    // 方法省略
}
```

```
@WebServlet的属性：
    name：servlet的名字
    urlPatterns：访问Servlet的路径，可以是多个
    value：可以和重要的属性（urlPatterns）等价。value属性如果只有一个值，可以省略value不写，只写值
    loadOnStartup：启动的优先级，数字越小优先级越高
```

三、Servlet初级应用

（一）Request、Response

request：请求，包含页面请求的所有信息。
response：响应，把内容响应的浏览器上。

Request常用方法：

```
// 获取页面请求的同名的单个参数，返回值是字符（多用于获取输入框的值：账号、密码等）
String parameter = req.getParameter("");
// 获取页面请求的同名的多个参数，返回值是数组（多用于获取多选框的值：爱好等）
String[] parameterValues = req.getParameterValues("");

参数：
    1、form表单的name属性的值
        <input type="text" name="user">
    2、url中key的值
        http://xxx?key1=v1&key2=v2
        http://xxx?name=admin&pass=admin

// 获取虚拟路径（工程名）
String contextPath = req.getContextPath();
// 获取Servlet请求路径
String servletPath = req.getServletPath();
// 获取指定的请求头信息
String header = req.getHeader("");
// 获取请求浏览器所在机器的的IP地址
String remoteAddr = req.getRemoteAddr();
```

Response常用方法：

```
// 把内容响应到页面
resp.getWriter().writer("");
```

（二）页面跳转

转发

```
req.getRequestDispatcher("路径").forward(req, resp);
```

重定向

```
resp.sendRedirect("路径");
```

转发和重定向的区别：
1、转发浏览器的地址不变。
2、重定向浏览器的地址会变。
3、转发可以携带数据，重定向不能。
4、转发只能请求当前工程的路径，重定向可以请求当前和其他工程的路径。

（三）JSP、EL、JSTL

1、JSP

指令元素：

```
语法结构：<%@ 指令名 属性1= "属性值1" 属性2= "属性值2" ... %>

page指令：主要设置页面的全局属性，编码格式。
    <%@ page contentType="text/html;charset=UTF-8" language="java"
pageEncoding="UTF-8" import="java.util.*" %>

include指令：包含，引入其他页面资源
    <%@ include file="top.jsp" %>

taglib指令：引入其他标签库
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

脚本代码：

```
<%  %>：普通的java代码
<%! %>：定义类/方法代码
<%= >：引用变量
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" import="java.util.*"
%>
```

```
<html>
<head>
    <title>Title</title>
</head>
<body>
    <%
        String str = "百度一下";
        System.out.println(str);
    %>
<a href="http://www.baidu.com"><%=str %></a>
<%!
    class Teacher {
        String name;
        String age;
    }
%>
<%
    Teacher t1 = new Teacher();
    t1.age = "18";

    List<String> list = new ArrayList<String>();
    list.add("tom");
    list.add("peter");
%>
<%=t1.age%>
<%=list.get(0)%>
<%=list.get(1)%>

<table border="1">
    <% for(int i = 0; i < 100; i++){ %>
        <tr>
            <td>Tom</td>
            <td>aaaa、</td>
            <td>bbb</td>
            <td>12313</td>
            <td>LKJS:DLFJS</td>
        </tr>
    <% } %>
</table>
</body>
</html>

```

注释

```
<!-- HTML注释 -->：只能注释HTML代码，同时在查看源码时可以看到
<%-- JSP注释 --%>：既可以注释HTML代码，也可以注释Java代码，在查看源码时看不到
```

内置对象

```
域对象：
    pageContext：当前页面
    request：一次请求
    session：浏览器
    application：服务器

其他对象：
    response：响应对象
    out：输出
    page：当前页面
    exception：异常，使用前提要表示页面是一个异常页面，使用page指令添加属性isErrorPage="true"
    config：配置
```

2、EL表达式

Expression Language（表达式语言），用于替换JSP页面的Java代码

语法结构：
${作用域.key}
${key}

用法：
1、计算返回表达式的值
2、获取作用域中的值

EL运算符：
1、算数运算符：+ - * /(div) %(mod)
2、关系运算符：> >= < <= !=
3、逻辑运算符：&&(and) ||(or) !(not)
4、空运算符：empty，判断对象是否为null或者长度是否为0

EL表达式从作用域中取值：
作用域：
pageScope
requestScope
sessioinScope
applicationScope

取值优先级：pageScope > requestScope > sessioinScope > applicationScope

当不指定具体作用域时，会按照优先级顺序查找，知道找到为止，找不到显示空字符。

EL表达式获取实体类中的属性的值：
1、EL表达式中写属性是实体类中get方法去掉get之后首字母小写。getName -> ${xxx.name}
2、去掉get之后，前两个字母都是大写，直接使用去掉get之后的值。getJSON - > ${xxx.JSON}

3、JSTL标签库

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<c:set var="speed" value="80"></c:set>

<c:if test="${speed <= 50}">
    低速行驶
</c:if>
<c:if test="${speed > 50 and speed <= 80}">
    中速行驶
</c:if>
<c:if test="${speed > 80}">
    高速行驶
</c:if>

<c:if test="${speed <= 50}"  var="result">
    低速行驶
</c:if>
<c:if test="${!result}">
    低速行驶
</c:if>

<%--<c:redirect url="http://www.baidu.com"/>--%>
<c:set var="sex" value="1231232"/>
<c:choose>
    <c:when test="${sex == 1}">
        男
    </c:when>
    <c:when test="${sex == 0}">
        女
    </c:when>
    <c:otherwise>
        未知
    </c:otherwise>
</c:choose>
<%--for(int i = 0; i <= 10; i++)--%>
<c:forEach var="i" begin="0" end="10" step="1">
    ${i}
</c:forEach>
<%
    List list = new ArrayList();
    list.add(12312313);
    list.add("aaa");
    list.add("bbbb");

```

```
    request.setAttribute("list", list);
%>
<%--for(Object obj : list)--%>
<c:forEach items="${list}" var="obj">
    ${obj}
</c:forEach>

<c:catch var="e">
    <% int i = 10 / 0;%>
</c:catch>
<c:out value="${e}"/>
```

（四）Servlet 实现CRUD

1、编写类继承HttpServlet
2、重写doGet、doPost方法
3、配置URL请求路径

1、分层

将模块和模块时间的联系（耦合度）降到最低，每个模块各司其职，又有一定的关联，同时使用接口，以便于后期扩展功能。主要思想是**MVC**（Model、View、Control）。
Model：模型，包含数据模型，业务模型；
View：视图，JSP、HTML等；
Control：控制，接收请求，转发请求，响应数据等；

工程架构：

```
servlet(包)
----| Xxx.java(类)

servcie(包)
----| XxxService.java(接口)
----| XxxServicesImpl(包)
--------| XxxServiceImpl.java(实现类)

dao(包)
----| XxxDao.java(接口)
----| xxxDaoImpl(包)
--------| XxxDaoImpl.java(实现类)

util(包)
----| xxx.java(类)

entity(包)
----| xxx.java(类)

Servlet：控制页面请求和响应已经核心的业务
```

```
Servcie：连接Servlet和Dao层，用于扩展功能
Dao：持久层，用于和数据库交互，把数据返回到上一层
Util：工具包，比如JDBC工具类，String工具类等；
Entity：实体类
```

2、查询(登录功能)

login.jsp

```
<form action="/login" method="post">
    <input type="text" name="user" placeholder="请输入账号">
    <input type="password" name="pass" placeholder="请输入密码">
    <input type="submit" value="提交"/>
</form>
```

LoginServlet.java

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 设置请求和响应的编码格式，防止中文乱码
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html;charset=UTF-8");

        /* 以下都是页面传过来的信息 */

        // 获取浏览器请求的账号和密码
        String user = req.getParameter("user");
        String pass = req.getParameter("pass");

//---------------------------------------------------------------------------
        /* 拿到页面传过来的信息之后，进行数据库或者其他业务的判断 */
        Connect conn = Connect.getInstance();

        String sql = "select count(1) from student where user = ? and pass = ?";
        List<Object> list = new ArrayList<Object>();
```

```
        list.add(user);
        list.add(pass);
        int count = conn.count(sql, list);

        // 获取打印对象
        PrintWriter writer = resp.getWriter();
//-------------------------------------------------------------------------
        /* 业务判断之后的数据/页面响应 */
        if(count > 0){
            writer.write("登录成功，欢迎" + user);
        } else {
            writer.write("登录失败");
        }
    }
}
```

3、查询(详情页面)

info.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <div class="main">
        <div class="menu">
            <a href="javascript:void(0)" class="export_but">导出</a>
            <a href="javascript:void(0)">批量删除</a>
        </div>
        <table>
            <thead>
                <tr>
                    <th><input type="checkbox" name="sno" class="check_rec"></th>
                    <th>学号</th>
                    <th>班级</th>
                    <th>姓名</th>
                    <th>性别</th>
                    <th>年龄</th>
                    <th>电话</th>
                    <th>籍贯</th>
                    <th>编辑</th>
                </tr>
            </thead>
            <tbody>
                <c:forEach items="${stuList}" var="stu">
```

```
                    <tr>
                        <td><input type="checkbox" name="sno" class="check_rec">
</td>
                        <td>${stu.sno}</td>
                        <td>${stu.clazz}</td>
                        <td>${stu.sname}</td>
                        <c:if test="${stu.sex == 1}">
                            <td>男</td>
                        </c:if>
                        <c:if test="${stu.sex == 0}">
                            <td>女</td>
                        </c:if>
                        <td>${stu.age}</td>
                        <td>${stu.tel}</td>
                        <td>${stu.home}</td>
                        <td>
                            <a href="javascript:void(0)" class="a_del">删除</a><a
href="javascript:void(0)" class="a_edit">修改</a>
                        </td>
                    </tr>
                </c:forEach>
                <%--<tr>
                    <td>1000</td>
                    <td>Java</td>
                    <td>张三</td>
                    <td>男</td>
                    <td>18</td>
                    <td>17337109782</td>
                    <td>河南</td>
                    <td>
                        <a href="javascript:void(0)" class="a_del">删除</a><a
href="javascript:void(0)" class="a_edit">修改</a>
                    </td>
                </tr>--%>
            </tbody>
        </table>
        <div class="page">
            <select name="toPage">
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
            </select>
            <a href="javascript:void(0)">尾页</a>
            <a href="javascript:void(0)">下一页</a>
            <a href="javascript:void(0)">1/1000</a>
            <a href="javascript:void(0)">上一页</a>
            <a href="javascript:void(0)">首页</a>
        </div>
```

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

InfoServlet.java

```
// 模拟数据库查询操作
List<Student> stuList = new ArrayList<Student>();
stuList.add(new Student("1000", "Tom", "1", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1001", "jack", "1", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1002", "Lili", "0", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1003", "Peter", "1", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1004", "Rose", "0", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1005", "Barry", "1", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1006", "Bob", "1", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1007", "Marry", "0", "18", "10086", "河南", "Java", null));
stuList.add(new Student("1008", "Mack", "1", "18", "17337109782", "河南", "Java",
null));

req.setAttribute("stuList", stuList);

req.getRequestDispatcher("/WEB-INF/page/info.jsp").forward(req, resp);
```

4、添加(注册功能)

regist.jsp

```
<form action="regist" method="get">
    <label>姓名：</label><input type="text" name="sname" class="inp" placeholder="请输
入姓名！"/><br>
    <label>性别：</label><input type="radio" name="sex" value="1" class="check"
checked/>男
    <input type="radio" name="sex" value="0"class="check"/>女<br>
    <label>年龄：</label><input type="number" name="age" class="inp" placeholder="请输
入年龄！"/><br>
    <label>电话：</label><input type="text" name="tel" class="inp" placeholder="请输入
电话/座机！"/><br>
    <label>籍贯：</label><select name="home" class="inp">
                            <option value="0">请选择籍贯</option>
                            <option value="1">河南</option>
                            <option value="2">河北</option>
                            <option value="3">湖南</option>
                            <option value="4">湖北</option>
                        </select><br>
    <label>班级：</label><select name="class" class="inp">
                            <option value="0">请选择班级</option>
```

```
                            <option value="1">Java</option>
                            <option value="2">Net</option>
                            <option value="3">Test</option>
                            <option value="4">BigData</option>
                        </select><br>
    <label>密码：</label><input type="password" name="pass" class="inp"
placeholder="请输入密码！"/><br>
    <label>确认密码：</label><input type="password" class="inp" placeholder="请确认密
码！"/><br>
    <label>爱好：</label><input type="checkbox" name="hobby"
value="game"class="check"/>游戏
                        <input type="checkbox" name="hobby"
value="tv"class="check"/>追剧
                        <input type="checkbox" name="hobby"
value="sing"class="check"/>唱歌<br>
    <input type="submit" value="注册" class="but sub"/>
    <input type="button" value="登录" class="but"
onclick="location.href='index.jsp'"/>
</form>
```

RegistServlet.java

```
@WebServlet("/regist")
public class RegistServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html;charset=UTF-8");

        String sname = req.getParameter("sname");
        String sex = req.getParameter("sex");
        String age = req.getParameter("age");
        String tel = req.getParameter("tel");
        String home = req.getParameter("home");
        String clas = req.getParameter("class");
        String pass = req.getParameter("pass");

        String[] hobbies = req.getParameterValues("hobby");

        System.out.println(sname);
```

```
25          System.out.println(sex);
26          System.out.println(age);
27          System.out.println(tel);
28          System.out.println(home);
29          System.out.println(clas);
30          System.out.println(pass);
31          System.out.println(hobbies);
32
33          RegistService registService = new RegistServiceImpl();
34          int count = registService.regist(new Student("", sname,  sex,  age,  tel,
     home,  clas, pass));
35          if(count > 0){
36              resp.getWriter().write("注册成功");
37          } else {
38              resp.getWriter().write("注册失败");
39          }
40      }
41  }
```

RegistService.java

```
1   public interface RegistService {
2       public int regist(Student stu);
3   }
```

RegistServiceImpl.java

```
1   public class RegistServiceImpl implements RegistService {
2       @Override
3       public int regist(Student stu) {
4           RegistDao registDao = new RegistDaoImpl();
5           return registDao.regist(stu);
6       }
7   }
```

RegistDao.java

```
1   public interface RegistDao {
2       public int regist(Student stu);
3   }
```

RegistDaoImpl.java

```java
public class RegistDaoImpl implements RegistDao {
    @Override
    public int regist(Student stu) {
        Connect instance = Connect.getInstance();
        String sql = "insert into student(sno, sname, sex, age, tel, native_place,
class, pass) "
                    + "values(seq_student.nextval, ?,?,?,?,?,?,?)";
        List<Object> param = new ArrayList<Object>();
        param.add(stu.getSname());
        param.add(stu.getSex());
        param.add(stu.getAge());
        param.add(stu.getTel());
        param.add(stu.getHome());
        param.add(stu.getClazz());
        param.add(stu.getPass());
        int count = instance.execute(sql, param);
        return count;
    }
}
```

5、删除/修改

删除和修改功能相对简单，通过页面把数据的主键传输到后台（servlet -> service -> dao -> db），使用request对象获取主键；然后把通过主键进心数据操作，操作成功之后，返回到详情页面即可；

四、Servlet高级应用

（一）文件上传下载

文件上传：把文件以流的形式通过浏览器上传到服务器；

要求：

1、页面form表单的提交方式必须为post；
2、form表单添加属性【enctype="multipart/form-data"】；
3、控制层（Servlet）添加注解@MultipartConfig；
4、通过头信息可以获取文件的部分信息；

步骤：

1、获取要上传的路径
2、获取文件流
3、把流写入硬盘

代码分析：

```
// 获取服务器上传的路径，参数是服务器上存放文件的路径
String serverPath = req.getServletContext().getRealPath("headImg");

<input type="file" name="headImg"/>
// 获取上传文件的对象，参数是form表单文件表单name属性的值
Part headImg = req.getPart("headImg");

// 获取文件头信息，可以获取文件名和后缀名，参数固定写法
String headImgHeader = headImg.getHeader("content-disposition");
// 获取输入流对象
InputStream inputStream = headImg.getInputStream();
```

文件下载：把文件以流的形式通过浏览器下载到磁盘；

要求：设置响应的头信息【"content-disposition"】【"attachment;filename=xxx"】

```
// 获取文件路径
String path = req.getParameter("path");
// 文件的绝对路径
String filePath = req.getServletContext().getRealPath(path);

// 根据文件绝对路径获取输入流
FileInputStream fis = new FileInputStream(filePath);

// 创建字节数组
byte[] bytes = new byte[fis.available()];
// 把输入流的内容写入到数组中
fis.read(bytes);

// 设置响应的编码格式
resp.setCharacterEncoding("UTF-8");
// 设置响应头信息，可以弹出下载框，不是在页面直接下载
resp.setHeader("content-disposition","attachment;filename=headImg.jpg");

// 把字节数组通过输出流输出到浏览器
resp.getOutputStream().write(bytes);
```

（二）会话跟踪

会话：一次会话中包含多次请求和相应；当浏览器给服务器发送请求时建立会话，直到一方断开链接，会话结束。会话的作用是在一次会话中多次请求之间共享数据。

实现方式：

1、客户端会话技术：Cookie
2、服务器会话技术：Session

1、Cookie

使用步骤：

```
// 1、创建Cookie对象，绑定数据
Cookie c = new Cookie(key, value);
// 2、发送Cookie对象
resp.addCookie();
// 3、获取Cookie对象，拿到数据
Cookie[] cs = req.getCookies();
```

Cookie的生命周期：

```
// 创建并发送Cookie
Cookie c = new Cookie(key, value);
resp.addCookie();

// 使用Cookie
Cookie[] cs = req.getCookies();
cs.getName();
cs.getValue();

// 销毁Cookie
1、浏览器关闭Cookie销毁
2、设置Cookie的存活时间
    setMaxAge(int seconds);
    seconds：
        正数：将Cookie写入到硬盘中。表示cookie的存活时间；
        负数：将Cookie写入到浏览器内存中。也是默认的方式
        零：删除Cookie信息
```

Cookie共享数据：

```
// Cookie在同一个服务器中跨工程共享资源
// 设置Cookie的获取范围，默认是当前工程，
// 如果需要在同一个服务器中跨工程共享资源，需要把路径设置为服务器的跟路径
setPath(String path); // setPath("/");
```

```
// Cookie在多个服务器中跨工程共享资源：
// 设置一级域名相同可以在多个服务器之间共享资源
setDomain(String path); // setDomain(".baidu.com");
```

Cookie的特点：

1、Cookie存在浏览器客户端（内存中）
2、单个Cookie的大小不能超过4KB
3、同一个域名下最多能存储20个Cookie

Cookie的作用：

1、Cookie用于存储少量、不重要的数据
2、在不登录的情况下，完成服务器对客户端的身份识别，一旦登录同步Cookie中的信息到服务器

2、Session

使用步骤：

```
// 1、获取Session对象
HttpSession session = req.getSession();
// 2、使用Session
session.setAttribute(key, val);
session.getAttribute(key);
session.removeAttribute(key);
```

Session的生命周期：

```
// 创建/获取session
HttpSession session = req.getSession();
```

```
// 使用session
// 给session赋值
session.setAttribute(key, val);
// 获取session的值
session.getAttribute(key);
```

```
// 销毁session
1、服务器关闭session销毁
2、session对象调用invalidate()方法
3、session默认的存活时间是30分钟，可以设置存活时间
    (1) session.setMaxInactiveInterval(seconds)
        "-1"表示永不失效
    (2) 修改tomcat的web.xml文件
        <session-config>
            <session-timeout>30</session-timeout>
        </session-config>
    (3) 修改工程的web.xml文件
        <session-config>
            <session-timeout>30</session-timeout>
```

```
14          </session-config>
```

Session共享数据：

```
1   // 如何保证在浏览器关闭之后还能再次获取同一个Session
2   HttpSession session = req.getSession();
3
4   Cookie cookie = new Cookie("JSESSIONID", session.getId());
5   cookie.setMaxAge(seconds);
6   resp.addCookie(cookie);
```

```
1   // 如何保证服务器关闭之后还能再获获取session中存储的数据
2   session钝化
3       服务器关闭时将session中的数据序列化服务器中
4   session活化
5       服务器加载时加载已经序列化的文件还原数据
```

Session的特点：

1、session数据存储在服务器端
2、session的数据任意类型，任意大小

Session和Cookie的区别：

1、Session存储数据在服务端、Cookie存储数据在客户端
2、Session存储数据没有大小限制
3、Session存储的数据更安全

思考？购物车的信息存放在Cookie还是Session?

（三）过滤器 （Filter）

1、过滤器概念

访问服务器资源时，过滤器可以将某些请求拦截下来，完成一些特殊的功能。一般用于完成一些公共的操作，比如：编码格式、登录验证、过滤敏感字符……

2、过滤器使用步骤

```
1   1.编写一个类，实现接口Filter，重写方法
2   2.配置Filter拦截路径
3       2.1 xml配置
4           <filter>
5               <filter-name>filterName</filter-name>
```

```
            <filter-class>filterName</filter-class>
        </filter>
        <filter-mapping>
            <filter-name>filterName</filter-name>
            <url-pattern>/*</url-pattern>
            <dispatcher></dispatcher>
        </filter-mapping>

    2.2 注解配置
        @WebFilter(value="/*", dispatcherTypes=DispatcherType.REQUEST)
            dispatcherTypes共有5个值可选：
                FORWARD：转发请求时执行过滤器。
                REQUEST：直接通过浏览器请求时执行过滤器。
                INCLUDE
                ASYNC
                ERROR
```

3、过滤器的生命周期

服务器启动时执行init方法，每一次请求都会执行对应的doFilter方法。当服务器正常关闭时，会执行destroy方法。

4、过滤器路径配置

```
1、具体的资源路径，/index.jsp
2、拦截指定请求路径资源，/login/*
3、拦截指定后缀路径资源，*.do
4、拦截所有资源：/*
```

5、配置多个过滤器

执行顺序：

1、xml配置中按照 `filter-mapping` 标签的顺序执行。

2、注解配置中按照 `Filter` 类的字典顺序执行。

（四）监听器（Listener）

1、监听器的概念

监听器就是监听某个对象的的状态变化的组件。

事件源：被监听的对象，web项目中监听的对象有三个作用域（request、response、servletContext）

监听器：监听 `事件源` 的对象

注册监听：将监听器和事件源绑定

响应行为： 监听器监听到事件源的状态变化时 所涉及的功能代码

2、监听器种类

	ServletContext域	HttpSession域	ServletRequest域
域对象的创建与销毁	ServletContextListener	HttpSessionListener	ServlerRequestListener
域对象内的属性的变化	ServletContextAttributeListener	HttpSessionAttributeListener	ServlerRequestAttributeListener

3、监听器的使用步骤

```
1、创建类，继承监听器，重写方法
2、配置监听
    注解配置：@WebListener
    xml配置：
        <listener>
            <listener-class>包名+类名</listener-class>
        </listener>
```

五、问题分析

404：路径错误，请求路径错/响应路径错；

解决思路：
1、检查服务器启动是否报错；
2、按照浏览器报错路径，检查配置是否出错，判断请求路径错误还是响应路径错误；
3、清除浏览器（ctrl+shift+delete）和项目缓存（删除项目缓存重新编译）；

500：后台代码报错，页面解析报错；

解决思路：
1、分析页面/后台报错信息，具体报错信息；是空指针异常还是其他异常；
2、找到具体报错具体地点（找自己的类，看包名是工程的）；
3、debug调试；

请求路径没有响应：

查看请求路径是否被过滤器过滤掉。

JS事件没有触发：

1、F12查看浏览器是否编译报错；尤其注意使用EL表达式时会造成表达式不完整，导致编译报错；
2、方法名/变量名不要使用关键字；
3、检查是否存在非英文的字符，尤其是空格；

页面CSS样式不更新：

1、检查CSS样式编写是否正确，是否存在样式覆盖的问题；
2、清除浏览器（ctrl+shift+delete）和项目缓存（删除项目缓存重新编译）；

六、JSON

导包

```
commons-beanutils-1.9.3.jar
commons-collections-3.2.2.jar
commons-lang-2.6.jar
commons-logging-1.2.jar
ezmorph-1.0.6.jar
json-lib-2.4-jdk15.jar
```

使用：

```
// JSON字符串转成JSON对象
JSONObject jsonObj = JSONObject.fromObject("JSON字符串");
// 通过key获取对象
JSONObject obj = jsonObj.getJSONObject("key");
// 获取对象中的属性
obj.get("key")

// 通过key获取数组对象
JSONArray array = jsonObject.getJSONArray("key");
// 获取数组的长度
array.size();
// 通过下标获取对象
JSONObject subObj = array.getJSONObject(index);
```

七、Servlet杂项配置

web.xml中配置欢迎页/首页

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

web.xml中配置404、500页面

```
<error-page>
    <error-code>404</error-code>
    <location>/404.html</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/500.html</location>
</error-page>
<error-page>
    <exception-type>java.lang.Throwable</exception-type>
    <location>/500.html</location>
</error-page>
```

在JSP页面路径前添加工程路径

```
${pageContext.request.contextPath}/url/...
```

URL编码

```
// URL地址中存在的特殊字符可能不能很好的解析到浏览器中请求，此时需要将URL进行转码。
String url = "URL地址";

// 转码
URLEncoder.encode(url, "UTF-8");
// 解码
URLDecoder.decode(url, "UTF-8");
```

八、技术扩展

验证码

页面加载时，访问后台，生成图片流和验证码，把验证码存储到session中，把图片流信息输出，放到img标签中展示。提交表单获取页面输入的验证码和session中的验证码比对。

```
public class CodeUtil {
    /**
     * 生成验证码图片
     * @param width 图片宽度，默认：130
     * @param height 图片高度，默认：40
     * @param count 验证码字符个数，默认：4
     * @return Map<String, BufferedImage> 键：生成的验证码；值：图片缓冲对象
     * @throws Exception
```

```
     */
    public static Map<String, BufferedImage> getCode(int width, int height, int count) {
        // 定义图片的宽高
        width = width == 0 ? 130 : width;
        height = height == 0 ? 40 : height;
        // 验证码的个数
        count = count == 0 ? 4 : count;

        // 字符库
        char[] codes = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','1','2','3','4','5','6','7','8','9','0'};

        // 图片缓冲区，定义宽高和颜色的编码
        BufferedImage bi = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

        // 获取画板
        Graphics graphics = bi.getGraphics();

        // 设置颜色
        graphics.setColor(Color.WHITE);
        graphics.fillRect(0, 0, width, height);

        // 从字符库中获取内容
        Random random = new Random();

        // 存储验证码
        StringBuffer sb = new StringBuffer();

        // 循环获取随机的字符
        for(int i = 1; i <= count; i++){
            // 随机数
            int temp = random.nextInt(codes.length);
            // 从字符库中获取字符
            String code = codes[temp] + "";
            sb.append(code);

            // 设置画笔颜色
            graphics.setColor(new Color(getRandomColor(), getRandomColor(), getRandomColor()));

            // 设置字体样式
            Font font = new Font("微软雅黑", Font.ITALIC, height / 2 + 5);
            graphics.setFont(font);

            // 把字符画到画布上
```

```
            graphics.drawString(code, width / (count + 2) * i, height / 2 + 5);
        }

        // 线条干扰项
        // 设置画笔颜色
        graphics.setColor(new Color(getRandomColor(), getRandomColor(),
getRandomColor()));
        // 画线
        graphics.drawLine(10, 10, 120, 30);
        graphics.drawLine(10, 30, 120, 10);

        // 随机麻点干扰项
        for(int j = 0; j < 150; j++){
            // 设置画笔颜色
            graphics.setColor(Color.RED);

            int x = random.nextInt(width);
            int y = random.nextInt(height);
            // 画线
            graphics.drawLine(x, y, x, y);
        }

        Map<String, BufferedImage> map = new HashMap<String, BufferedImage>();
        map.put(sb.toString(), bi);

        return map;
    }

    /**
     * 随机生成颜色代码，范围是0-255的rgb颜色
     * @return
     */
    public static int getRandomColor(){
        return new Random().nextInt(255);
    }
}
```

```
@WebServlet("/getCode")
public class GetCodeServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        doPost(req, resp);
    }

    @Override
```

```
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 通过工具类获取验证码map，图片的宽高可以从前台页面获取
        Map<String, BufferedImage> code = CodeUtil.getCode(130, 40, 4);

        // 遍历map
        Iterator<String> iterator = code.keySet().iterator();

        if(iterator.hasNext()){
            // 获取key
            String key = iterator.next();
            // 把key存储到Session中便于登录时获取校验
            req.getSession().setAttribute("yzmCode", key);
            // 通过key获取字符缓冲对象，写入磁盘
            ImageIO.write(code.get(key), "JPEG", new FileOutputStream("D://yzmCode.jpg"));
            // 通过key获取字符缓冲对象，写入浏览器输出对象
            ImageIO.write(code.get(key), "JPEG", resp.getOutputStream());
        }
    }
}
```

密码加密

项目中数据不能存储明文密码，所以要对密码进行加密，常见的加密方式有：MD5、Base64、加盐、钥匙串、对称加密算法、不对称加密算法等。

注册时把密码加密后存储到数据库。登陆时，把密码加密后，与数据库数据对比。

```
package com.soft.util;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5Util {
    public static String encode(String pass) throws NoSuchAlgorithmException {
        // 创建MessageDigest对象指定加密方式
        String algorithm = "MD5";
        MessageDigest digest = MessageDigest.getInstance(algorithm);

        // 将密码转化为byte数组，并通过updae方法封装数据
        digest.update(pass.getBytes());

        // 返回封装后的数据
        byte[] bytes = digest.digest();

        String result = new BigInteger(bytes).toString(16);
```

```
        System.out.println(result);
        return result;
    }
}
```

邮件

```
public class MailUtil {
    public static void sendMail() throws Exception {
        // 1、配置发件人信息：账号，密码，邮箱服务器，端口号
        String from = "邮箱地址";
        String pass = "授权码，并非登录的真实密码";
        String host = "smtp.163.com";
        String port = "25";

        Properties properties = System.getProperties();
        // 发邮件时验证密码
        properties.setProperty("mail.smtp.auth", "true");
        // 发邮件的方式
        properties.setProperty("mail.transoprt", "smtp");
        // 邮箱服务器地址，端口
        properties.setProperty("mail.smtp.host", host);
        properties.setProperty("mail.smtp.port", port);

        // 2、构建会话
        Session session = Session.getDefaultInstance(properties);

        // 3、构建邮件对象
        MimeMessage mm = new MimeMessage(session);

        // 4、设置邮件的发件人
        mm.setFrom(from);
        // 5、设置收件人
        mm.addRecipients(Message.RecipientType.TO, "邮箱地址");
        /*
        mm.addRecipients(Message.RecipientType.TO, new Address[]{
                new InternetAddress("邮箱地址"),
                new InternetAddress("邮箱地址"),
                new InternetAddress("邮箱地址")
        });
        */

        // 6、设置邮件主题
        mm.setSubject("我在通过Java发送邮件", "UTF-8");

        // 7、设置邮件内容
        mm.setText("你好：Hello World!");
```

```
        mm.setContent("<h1>你好：Hello World!</h1>","text/html;charset=utf-8");

        // 8、获取邮件传输对象
        Transport transport = session.getTransport("smtp");
        // 9、通过账号密码连接
        transport.connect(from, pass);
        // 10、发送信息
        transport.sendMessage(mm, mm.getAllRecipients());
        // 11、关闭连接
        transport.close();
    }
}
```

支付宝支付

导包

alipay-sdk-java-4.8.10.ALL.jar

调用接口

```
package com.soft.servlet;

import com.alipay.api.AlipayApiException;
import com.alipay.api.AlipayClient;
import com.alipay.api.DefaultAlipayClient;
import com.alipay.api.request.AlipayTradePagePayRequest;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.UUID;

@WebServlet("/pay")
public class PayServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String APP_ID = "应用ID";
        String APP_PRIVATE_KEY = "私钥";
```

```
        String FORMAT = "JSON";
        String CHARSET = "UTF-8";
        String ALIPAY_PUBLIC_KEY = "公钥";
        String SIGN_TYPE = "RSA2";

        AlipayClient alipayClient = new DefaultAlipayClient("https://openapi.alipaydev.com/gateway.do", APP_ID, APP_PRIVATE_KEY, FORMAT, CHARSET, ALIPAY_PUBLIC_KEY, SIGN_TYPE); //获得初始化的AlipayClient
        AlipayTradePagePayRequest alipayRequest = new AlipayTradePagePayRequest();//创建API对应的request

        // 设置支付成功之后要跳转的页面 -> 给用户反馈
        alipayRequest.setReturnUrl("回调地址");
        // 设置支付成功之后要提醒的路径 -> 给商家反馈
        //alipayRequest.setNotifyUrl("http://domain.com/CallBack/notify_url.jsp");//在公共参数中设置回跳和通知地址

        String order = UUID.randomUUID().toString();
        String totalAmount = "0.1";
        String subject = "Iphone 11";
        String body = "暗夜绿 256G";

        alipayRequest.setBizContent("{" +
                "    \"out_trade_no\":\"" + order + "\"," +
                "    \"product_code\":\"FAST_INSTANT_TRADE_PAY\"," +
                "    \"total_amount\":"+ totalAmount + "," +
                "    \"subject\":\"" +subject + "\"," +
                "    \"body\":\"" + body + "\"," +
                "    \"passback_params\":\"merchantBizType%3d3C%26merchantBizNo%3d2016010101111\"," +
                "    \"extend_params\":{" +
                "    \"sys_service_provider_id\":\"2088511833207846\"" +
                "    }"+
                "  }");//填充业务参数

        String form="";
        try {
            form = alipayClient.pageExecute(alipayRequest).getBody(); //调用SDK生成表单
        } catch (AlipayApiException e) {
            e.printStackTrace();
        }

        resp.setContentType("text/html;charset=" + CHARSET);
        resp.getWriter().write(form);//直接将完整的表单html输出到页面
        resp.getWriter().flush();
        resp.getWriter().close();
```

```
    }
}
```

百度智能云-图片识别

导包

java-sdk-4.12.0.jar
slf4j-api-1.6.1.jar
slf4j-log4j12-2.0.0-alpha1.jar

调用接口

```
package com.soft.servlet;

import com.baidu.aip.ocr.AipOcr;
import org.json.JSONObject;
import java.util.HashMap;

public class ImgShiBie {
    //设置APPID/AK/SK
    public static final String APP_ID = "APP_ID";
    public static final String API_KEY = "API_KEY";
    public static final String SECRET_KEY = "SECRET_KEY";

    public static void main(String[] args) {
        // 初始化一个AipOcr
        AipOcr client = new AipOcr(APP_ID, API_KEY, SECRET_KEY);

        // 可选：设置网络连接参数
        client.setConnectionTimeoutInMillis(2000);
        client.setSocketTimeoutInMillis(60000);

        // 可选：设置代理服务器地址, http和socket二选一，或者均不设置
        /*client.setHttpProxy("proxy_host", proxy_port);  // 设置http代理
        client.setSocketProxy("proxy_host", proxy_port);  // 设置socket代理*/

        // 可选：设置log4j日志输出格式，若不设置，则使用默认配置
        // 也可以直接通过jvm启动参数设置此环境变量
        System.setProperty("aip.log4j.conf", "path/to/your/log4j.properties");

        // 调用接口
        String path = "C:\\Users\\admin\\Desktop\\QQ浏览器截图20191114171501.png";
        JSONObject res = client.basicGeneral(path, new HashMap<String, String>());
        System.out.println(res.toString(2));
    }
}
```