JSP

一、JSP规范介绍

- 来自于JAVAEE规范中一种
- JSP规范制定了如何开发JSP文件代替响应对象将处理结果写入到响应体的开发流程
- JSP规范制定了Http服务器应该如何调用管理JSP文件

二、响应对象存在弊端

- 适合将数据量较少的处理结果写入到响应体
- 如果处理结果数量过多，使用响应对象增加开发难度

三、JSP文件优势

- JSP文件在互联网通信过程，是响应对象替代品.
- 降低将处理结果写入到响应体的开发工作量降低处理结果维护难度
- 在JSP文件开发时，可以直接将处理结果写入到JSP文件不需要手写out.print()命令，在Http服务器调用JSP文件时，根据JSP规范要求自动的将JSP文件书写的所有内容通过输出流写入到响应体

四、HTML文件与JSP文件区别

- 作为资源文件类型不同
  - HTML文件属于静态资源文件，其相关命令需要在浏览器编译并执行的.
  - JSP文件属于动态资源文件，其相关命令需要在服务端编译并执行的
- 调用形式不同
  - 如果浏览器访问HTML文件，此时Http服务器直接通过一个输出流将HTML文件中所有的内容写入到响应体
  - 如果浏览器访问JSP文件。此时Http服务器根据JSP规范来操作JSP文件编辑---->编译----->调用

五、JSP文件运行原理

```
一。Http服务器调用JSP文件步骤:
    1.Http服务器将JSP文件内容【编辑】为一个Servlet接口实现类（.java）
    2.Http服务器将Servlet接口实现类【编译】为class文件(.class)
    3.Http服务器负责创建这个class的实例对象，这个实例对象就是Servlet实例对象
    4.Http服务器通过Servlet实例对象调用_jspService方法，将jsp文件内容写入到响应体
二。Http服务器【编辑】与【编译】JSP文件位置:
    标准答案：我在【work】下看到这个证据
    C:\Users\[登录windows系统用户角色名]\.IntelliJIdea2018.3\system\tomcat\[网站工作空
间]\work\Catalina\localhost\【网站别名】\org\apache\jsp
```

- Tomcat根据JSP规范，将被访问的JSP文件[编辑]为一个java文件。这个Java文件是Servlet接口实现类
- Tomcat根据JSP规范，调用JVM（javac one_jsp.java）将这个java文件[编译]为class类型
- Tomcat根据JSP规范负责生成这个class文件的实例对象。这个实例对象是一个Servelt接口实例对象
- Tomcat根据JSP规范通过实例对象调用class文件中_jspService方法
- _jspService方法在运行时负责将JSP文件中书写内容写入到响应体中

六、在JSP文件中如何书写Java命令

1. 执行标记命令格式:

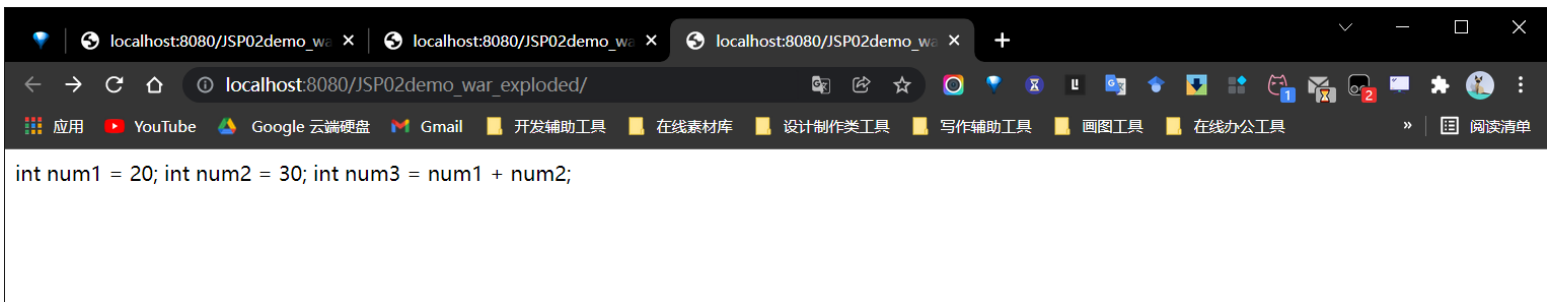
```
<% int a  =10;  %> <!--声明局部变量-->
<% boolean flag = 30 >= 40; %>  <!--Java中表达式(数学表达式，关系表达式，逻辑表达式)-->
<%
    if(判断条件){
    }else{
    }
    while(){
    }
%>  <!--书写控制语句-->
```

2. 执行标记命令作用：通知Http服务器将JSP文件中Java命令与其他普通执行结果进行区分
3. 输出标记命令格式

```
<%=java的变量名%>
<%=java的表达式%>
```

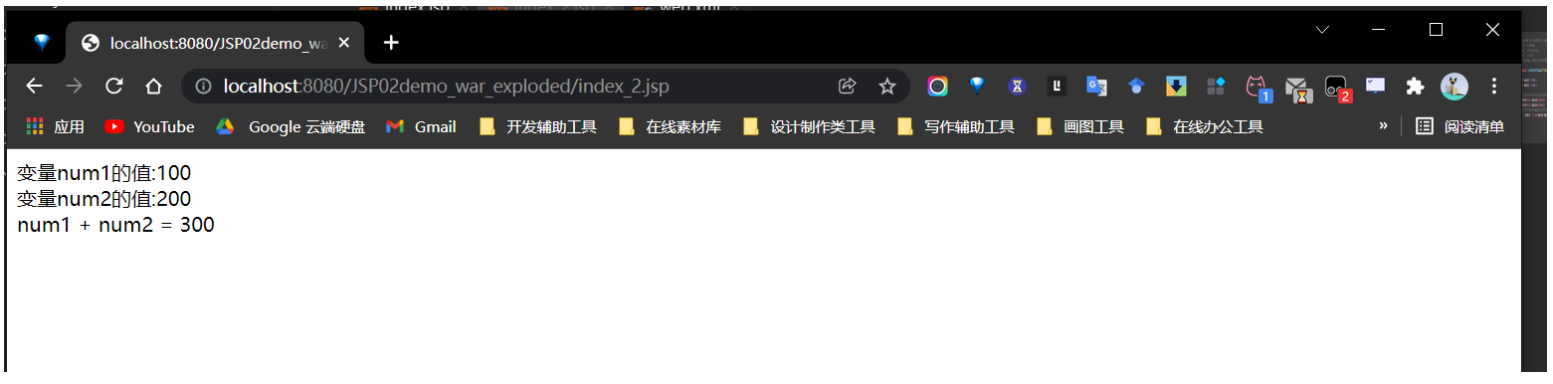
4. 输出标记命令作用：通知Tomcat将输出标记中【变量的值】或则输出标记中【表达式运算结果】写入到响应体
5. 代码:

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--在JSP文件中直接书写Java命令，是不能被JSP文件识别，此时只会被当做字符串写入到响应体-->
int num1 =20;
int num2 =30;
int num3 = num1 + num2;
```




```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
  //在jsp文件中，只有书写在执行标记中内容才会被当做Java命令
  //1.声明Java变量
  int num1 = 100;
  int num2 = 200;
  //2.声明运行表达式：数学运算，关系运算，逻辑运算
  int num3 = num1 + num2; //数学运算
  int num4 = num2>=num1?num2:num1;//关系运算
  boolean num5 = num2>=200 && num1>=100;//逻辑运算
  //3.声明控制语句
  if(num2>=num1){
  }else{
  }
  for(int i=1;i<=10;i++){
  }
%>
```

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
   int num1 =100;
   int num2 =200;
%>
<!--在JSP文件，通过输出标记，通知JSP将Java变量的值写入到响应体-->
变量num1的值:<%=num1%><br/>
变量num2的值:<%=num2%><br/>
<!--执行标记还可以通知Jsp将运算结果写入到响应体-->
num1 + num2 = <%=num1+num2%>
```




```jsp
<%@ page import="com.example.entity.Student" %><!--自动导包-->
<%@ page import="java.util.*" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
    //创建一个Student类型对象
    Student stu = new Student(10,"mike");//id name
    List list= new ArrayList();
%>
学员编号:<%=stu.getSid()%><br/>
学员姓名:<%=stu.getSname()%>
```


```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--JSP中所有的执行标记当作一个整体-->
<%
  int age =15;
%>

```

```jsp
<%
  if(age >=18){
%>

<font style="color:red;font-size:45px">热烈欢迎</font>

<%
  }else{
%>

<font style="color:red;font-size:45px">谢绝入内</font>

<%
  }
%>

<!--
{
    int age =15;
    if(age>=18){
        out.print(<font style="color:red;font-size:45px">热烈欢迎</font>)
    }else{
        out.print(<font style="color:red;font-size:45px">热烈欢迎</font>)
    }
}
-->
```

```jsp
<%@ page import="com.example.entity.Student" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.ArrayList" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--制造数据-->
<%
    Student stu1 = new Student(10,"mike");
    Student stu2 = new Student(20,"allen");
    Student stu3 = new Student(30,"smith");
    List<Student> list = new ArrayList();
    list.add(stu1);
    list.add(stu2);
    list.add(stu3);
%>

<!--数据输出-->
<table border="2" align="center">
    <tr>
        <td>学员编号</td>
        <td>学员姓名</td>
    </tr>
<%
    for(Student stu:list){
%>
    <tr>
        <td><%=stu.getSid()%></td>
        <td><%=stu.getSname()%></td>
    </tr>
<%
    }
%>
</table>
```

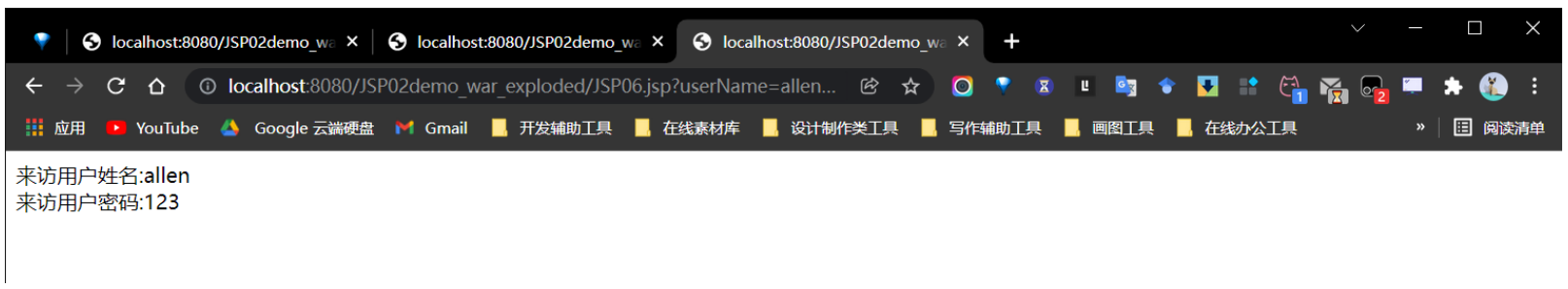
学员编号	学员姓名
10	mike
20	allen
30	smith

七、JSP文件内置对象

1. request

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--
    JSP文件内置对象: request
        类型：HttpServletRequest
        作用: 在JSP文件运行时读取请求包信息
            与Servlet在请求转发过程中实现数据共享
    浏览器:  http://localhost:8080/myWeb/request.jsp?userName=allen&password=123
-->
<%
    //在JSP文件执行时，借助于内置request对象读取请求包参数信息
    String userName = request.getParameter("userName");
```

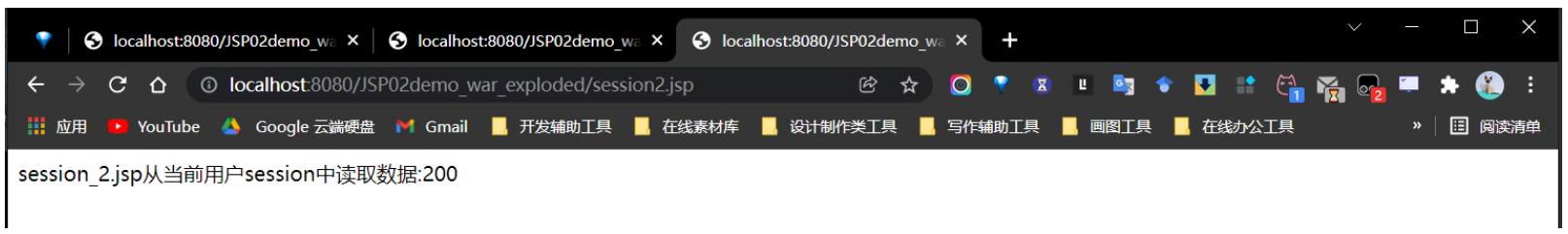
```
    String password =request.getParameter("password");
%>
来访用户姓名:<%=userName%><br/>
来访用户密码:<%=password%>
```



2. session

```
session.jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--
    JSP文件内置对象:session
      类型:HttpSession
      作用: JSP文件在运行时，可以session指向当前用户私人储物柜，添加
          共享数据，或则读取共享数据
-->
<!--将共享数据添加到当前用户私人储物柜-->
<%
  // HttpSession session = request.getSession();
  session.setAttribute("key1", 200);
%>
```

```
session2.jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--
    session.jsp 与  session2.jsp为同一个用户/浏览器提供服务。
    因此可以使用当前用户在服务端的私人储物柜进行数据共享
-->
<%
    Integer value=(Integer) session.getAttribute("key1");
%>
session_2.jsp从当前用户session中读取数据:<%=value%>
```



3. application

```
<!--
       ServletContext application;全局作用域对象
       同一个网站中Servlet与JSP，都可以通过当前网站的全局作用域对象实现数据共享
       JSP文件内置对象 ：  application
-->

<%
    application.setAttribute("key1", "hello world");
%>
```

```
public class OneServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
          ServletContext application = request.getServletContext();
          String value = (String)application.getAttribute("key1");
        System.out.println("value = "+value);
    }
}
```

八、Servlet与JSP文件分工

```
一。Servlet与JSP分工:
    Servlet:  负责处理业务并得到【处理结果】--------------------大厨
    JSP:      不负责业务处理，主要任务将Servlet中【处理结果】写入到响应体----传菜员
二。Servlet与JSP之间调用关系
    Servlet工作完毕后，一般通过请求转发方式向Tomcat申请调用JSP
三。Servlet与JSP之间如何实现数据共享
    Servlet将处理结果添加到【请求作用域对象】
    JSP文件在运行时从【请求作用域对象】得到处理结果
```

```java
public class Student {
    private Integer sid;
    private String sname;
    public Integer getSid() {
        return sid;
    }
    public void setSid(Integer sid) {
        this.sid = sid;
    }
    public String getSname() {
        return sname;
    }
    public void setSname(String sname) {
        this.sname = sname;
    }
    public Student() {
    }
    public Student(Integer sid, String sname) {
        this.sid = sid;
        this.sname = sname;
    }
}
```

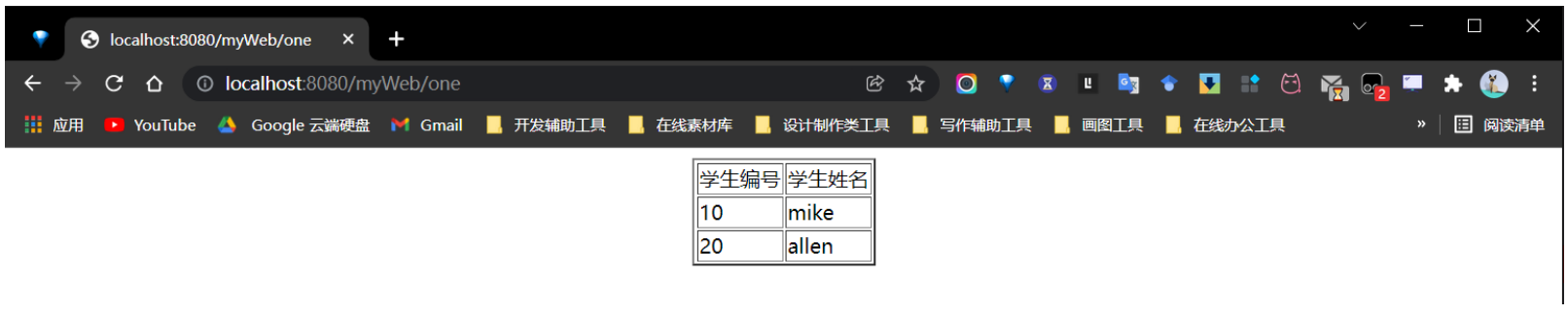
```java
public class OneServlet extends HttpServlet {
    //处理业务，得到处理结果-----查询学员信息
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        Student s1 = new Student(10,"mike");
        Student s2 = new Student(20,"allen");
        List<Student> stuList = new ArrayList();
        stuList.add(s1);
        stuList.add(s2);

        //将处理结果添加到请求作用域对象
        request.setAttribute("key", stuList);

        //通过请求转发方案，向Tomcat申请调用user_show.jsp
        //同时将request与response通过tomcat交给user_show.jsp使用
        request.getRequestDispatcher("/user_show.jsp").forward(request, response);
    }
}
```

```jsp
<%@ page import="java.util.List" %>
<%@ page import="com.example.entity.Student" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%
    //从请求作用域对象得到OneServlet添加进去的集合
    List<Student> stuList = (List<Student>) request.getAttribute("key");
%>
<!--将处理结果写入到响应体-->
<table border="2" align="center">
    <tr>
        <td>学生编号</td>
        <td>学生姓名</td>
    </tr>
    <%
        for(Student student : stuList){
    %>
    <tr>
        <td><%=student.getSid()%></td>
        <td><%=student.getSname()%></td>
    </tr>
    <%
        }
    %>
</table>
```



九、在线考试管理系统---试题功能

```
准备工作
任务：  在线考试管理系统----试题信息管理模块
子任务:
        添加试题
        查询试题
        删除试题
        更新试题
准备工作：
        1.准备试题信息表:单选题，每道题有4个选项  一个正确答案
          create table question(
            questionId int primary key auto_increment,
            title  varchar(50),# 10-8=?
            optionA varchar(20),#A: 9
            optionB varchar(20),#B: 1
            optionC varchar(20),#C: 2
            optionD varchar(20),#D: 0
            answer  char(1)     #正确答案：  C
          )
          2.com.bjpowernode.entity.Question
```

1 试题添加功能

```java
public class Question {
    private Integer questionId;
    private String title;
    private String optionA;
    private String optionB;
    private String optionC;
    private String optionD;
    private String answer;
    public Question() {
    }
    public Question(Integer questionId, String title, String optionA, String optionB, String optionC,
String optionD, String answer) {
        this.questionId = questionId;
        this.title = title;
        this.optionA = optionA;
        this.optionB = optionB;
        this.optionC = optionC;
        this.optionD = optionD;
        this.answer = answer;
    }

    public Integer getQuestionId() {
        return questionId;
    }
    public void setQuestionId(Integer questionId) {
        this.questionId = questionId;
    }

    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }

    public String getOptionA() {
        return optionA;
    }
    public void setOptionA(String optionA) {
        this.optionA = optionA;
    }

    public String getOptionB() {
        return optionB;
    }
    public void setOptionB(String optionB) {
        this.optionB = optionB;
    }

    public String getOptionC() {
        return optionC;
    }
    public void setOptionC(String optionC) {
        this.optionC = optionC;
    }

    public String getOptionD() {
        return optionD;
```

```
    }
    public void setOptionD(String optionD) {
        this.optionD = optionD;
    }

    public String getAnswer() {
        return answer;
    }
    public void setAnswer(String answer) {
        this.answer = answer;
    }
}
```

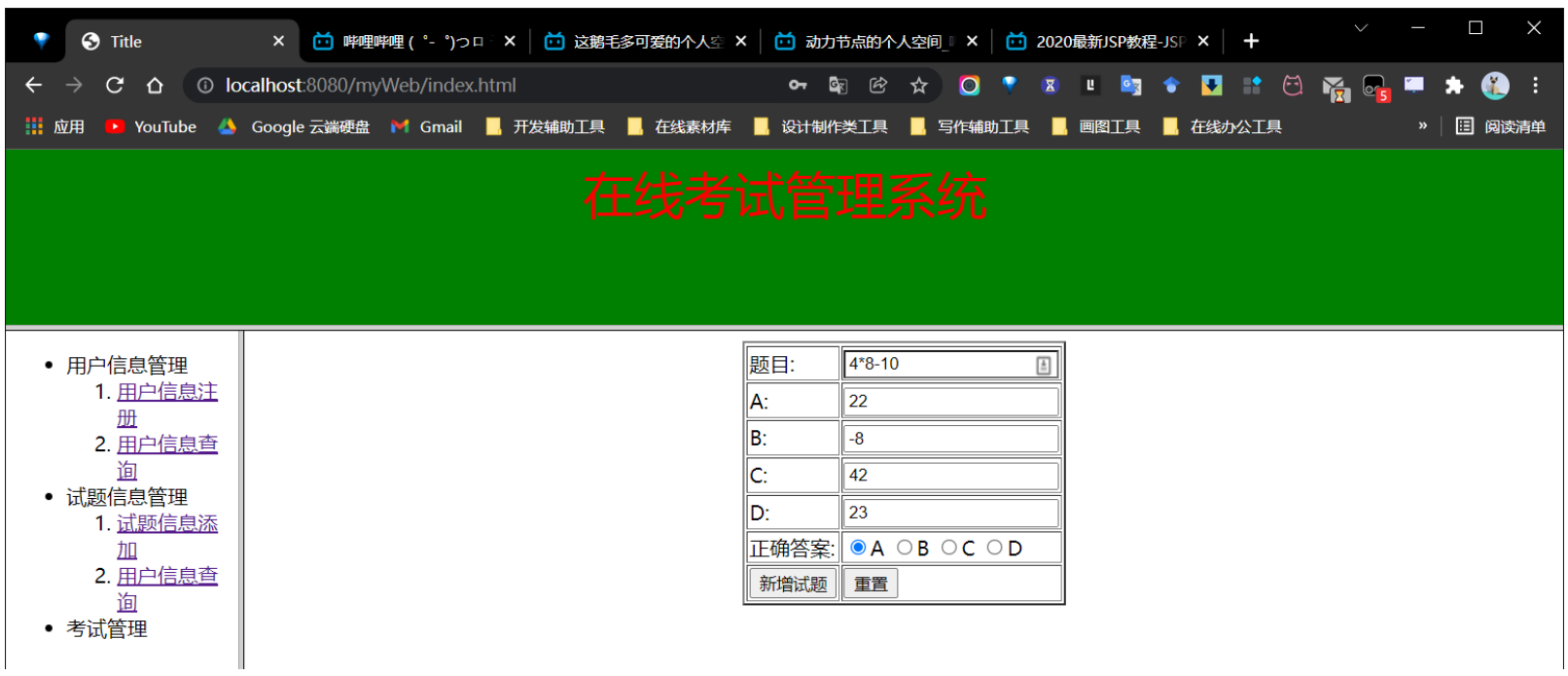
```
public class QuestionDao {
    public static int add(Question question){
        Connection conn = null;
        PreparedStatement ps = null;
        int res = 0;
        try {
            String sql = "insert into question(title, optionA, optionB, optionC, optionD, answer)
values(?,?,?,?,?,?)";
            conn = JdbcUtil.getConnection();
            ps = conn.prepareStatement(sql);
            ps.setString(1, question.getTitle());
            ps.setString(2, question.getOptionA());
            ps.setString(3, question.getOptionB());
            ps.setString(4, question.getOptionC());
            ps.setString(5, question.getOptionD());
            ps.setString(6, question.getAnswer());
            res = ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, null);
        }
        return res;
    }
}
```

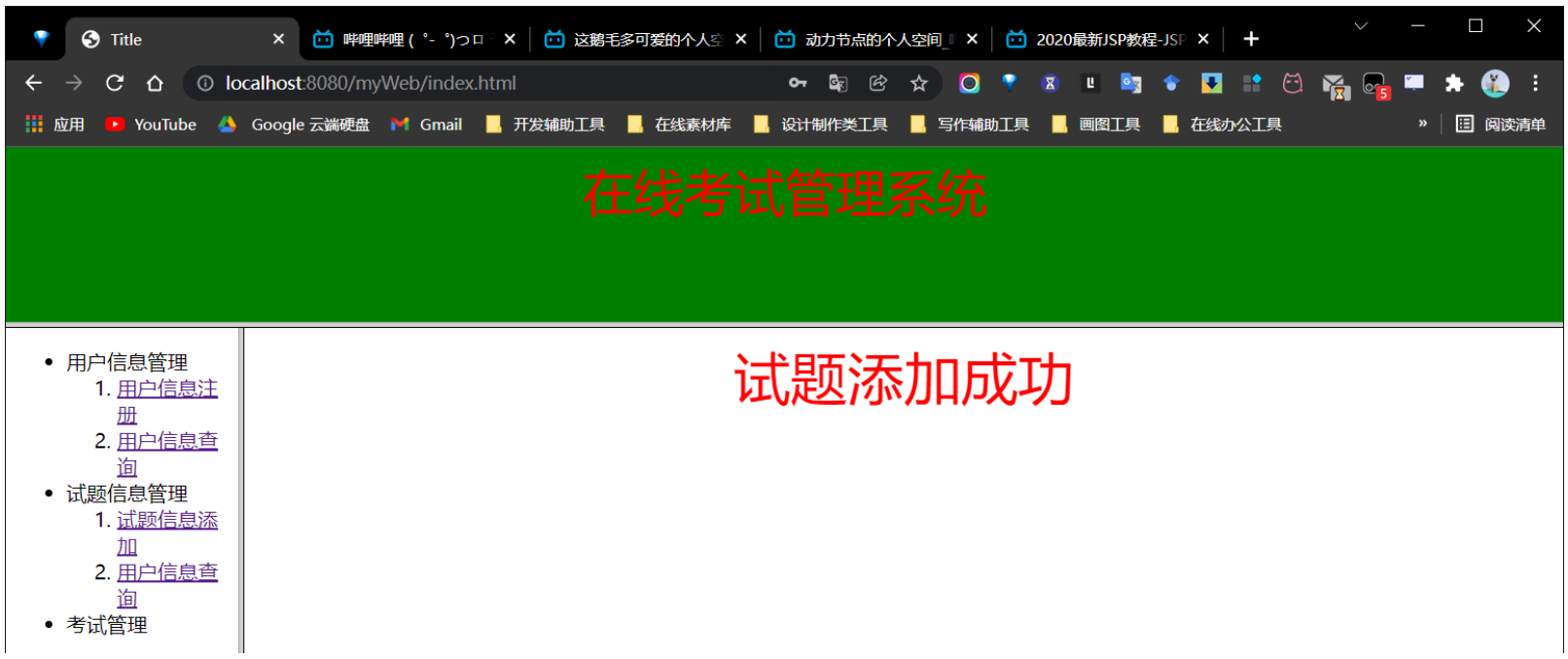
```
public class QuestiomAddServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        String title, optionA, optionB, optionC, optionD, answer;
        //1.调用请求对象读取请求信息，得到新增信息内容
        title = request.getParameter("title");
        optionA = request.getParameter("optionA");
        optionB = request.getParameter("optionB");
        optionC = request.getParameter("optionC");
        optionD = request.getParameter("optionD");
        answer = request.getParameter("answer");
        //2.调用Dao对象将Insert命令推送到数据库，并得到处理结果
        Question question = new Question(null, title, optionA, optionB, optionC, optionD, answer);
        int res = QuestionDao.add(question);
        //3.通过请求转发，向Tomcat索要info.jsp将处理结果写入到响应体
        if(res == 1){
            request.setAttribute("info", "试题添加成功");
        } else {
            request.setAttribute("info", "试题添加失败");
        }
        //4.请求转发向Tomcat申请info.jsp将结果写入到响应体
        request.getRequestDispatcher("/info.jsp").forward(request, response);
    }
}
```

```
<!--info.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
  <center>
    <%
      String info = (String) request.getAttribute("info");
    %>
    <font style="color: red; font-size: 45px">
      <%=info%>
    </font>
  </center>
</body>
```

```
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
  <Center>
    <form action="/myWeb/question/add" method="get">
      <table border="2">
        <tr>
          <td>题目:</td>
          <td><input type="text" name="title"></td>
        </tr>
        <tr>
          <td>A:</td>
          <td><input type="text" name="optionA"></td>
        </tr>
        <tr>
          <td>B:</td>
          <td><input type="text" name="optionB"></td>
        </tr>
        <tr>
          <td>C:</td>
          <td><input type="text" name="optionC"></td>
        </tr>
        <tr>
          <td>D:</td>
          <td><input type="text" name="optionD"></td>
        </tr>
        <tr>
          <td>正确答案:</td>
          <td>
            <input type="radio" name="answer" value="A">A
            <input type="radio" name="answer" value="B">B
            <input type="radio" name="answer" value="C">C
            <input type="radio" name="answer" value="D">D
          </td>
        </tr>
        <tr>
          <td><input type="submit" value="新增试题"></td>
          <td><input type="reset" ></td>
        </tr>
      </table>
    </form>
  </Center>
</body>
</html>
```





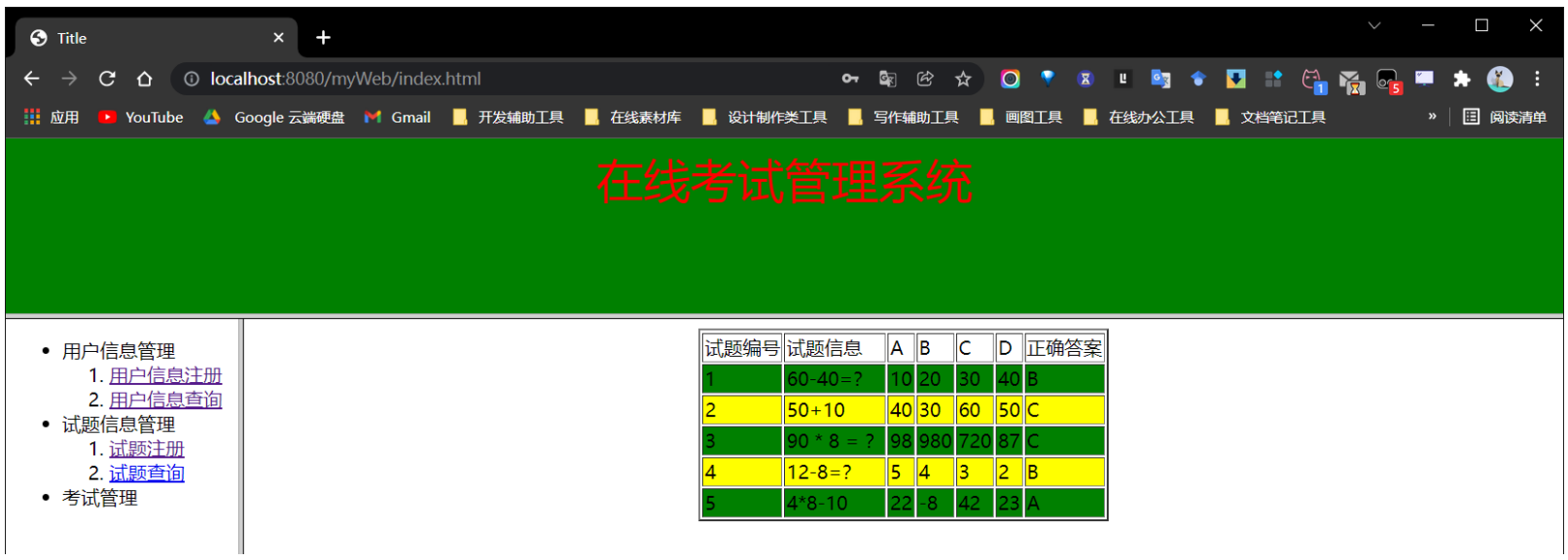
2 所有试题查询功能

```
public class QuestionFindServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.调用DAO对象将查询命令推送到数据库中得到试题信息【List】
        List<Question> questionList = QuestionDao.findAll();
        //2.通过请求转发，向Tomcat索要question.jsp，将试题信息写入到响应体
        request.setAttribute("key", questionList);
        request.getRequestDispatcher("/question.jsp").forward(request, response);
    }
}
```

```
public class QuestionDao {
    public static List<Question> findAll(){
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        List<Question> questionList = new ArrayList<>();
        try {
            String sql = "select * from question";
            conn = JdbcUtil.getConnection();
            ps = conn.prepareStatement(sql);
            rs = ps.executeQuery();
            while(rs.next()){
                Integer questionId = rs.getInt("questionId");
                String title = rs.getString("title");
                String optionA = rs.getString("optionA");
                String optionB = rs.getString("optionB");
                String optionC = rs.getString("optionC");
                String optionD = rs.getString("optionD");
                String answer = rs.getString("answer");
                Question question = new Question(questionId, title, optionA, optionB, optionC, optionD,
answer);
                questionList.add(question);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, rs);
        }
        return questionList;
    }
}
```

```
<%@ page import="java.util.List" %>
<%@ page import="com.example.entity.Question" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
  <table border="2" align="center">
      <tr>
          <td>试题编号</td>
          <td>试题信息</td>
          <td>A</td>
          <td>B</td>
          <td>C</td>
```

```
            <td>D</td>
            <td>正确答案</td>
        </tr>
        <%
            List<Question> list = (List<Question>) request.getAttribute("key");
            for(int i = 0; i < list.size(); i++){
                Question question = list.get(i);
         %>
        <%
          if(i % 2 == 0){
         %>
        <tr style="background-color: green">
        <%
          }else{
        %>
        <tr style="background-color: yellow">
        <%
          }
        %>
            <td><%=question.getQuestionId()%></td>
            <td><%=question.getTitle()%></td>
            <td><%=question.getOptionA()%></td>
            <td><%=question.getOptionB()%></td>
            <td><%=question.getOptionC()%></td>
            <td><%=question.getOptionD()%></td>
            <td><%=question.getAnswer()%></td>
        </tr>
        <%
            }
        %>
    </table>
</body>
</html>
```



3 试题删除功能

```
public class QuestionDeleteServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.调用请求对象读取请求头参数信息，得到试题编号
        String questionId = request.getParameter("questionId");
        //2.调用Dao对象将删除命令推送到数据库服务器
        int result = QuestionDao.delete(questionId);
        //3.调用info.jsp将处理结果写入到响应体
        if(result ==1){
            request.setAttribute("info", "试题删除成功");
        }else{
            request.setAttribute("info", "试题删除失败");
        }
        request.getRequestDispatcher("/info.jsp").forward(request, response);
    }
}
```

```
public class QuestionDao {
    public static int delete(String questionId){
        Connection conn = null;
        PreparedStatement ps = null;
        int res = 0;
        try {
            conn = JdbcUtil.getConnection();
            String sql="delete from question where questionId=?";
            ps = conn.prepareStatement(sql);
            ps.setInt(1, Integer.valueOf(questionId));
            res = ps.executeUpdate();
        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, null);
        }
        return res;
    }
}

```

```
<%@ page import="java.util.List" %>
<%@ page import="com.example.entity.Question" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
  <table border="2" align="center">
      <tr>
          <td>试题编号</td>
          <td>试题信息</td>
          <td>A</td>
          <td>B</td>
          <td>C</td>
          <td>D</td>
          <td>正确答案</td>
      </tr>
      <%
          List<Question> list = (List<Question>) request.getAttribute("key");
          for(int i = 0; i < list.size(); i++){
              Question question = list.get(i);
       %>
      <%
        if(i % 2 == 0){
       %>
      <tr style="background-color: green">
      <%
        }else{
      %>
      <tr style="background-color: yellow">
      <%
        }
      %>
          <td><%=question.getQuestionId()%></td>
          <td><%=question.getTitle()%></td>
          <td><%=question.getOptionA()%></td>
          <td><%=question.getOptionB()%></td>
          <td><%=question.getOptionC()%></td>
          <td><%=question.getOptionD()%></td>
          <td><%=question.getAnswer()%></td>
          <td style="background-color: white">
              <a href="/myWeb/question/delete?questionId=<%=question.getQuestionId()%>">删除试题</a>
          </td>
          <td style="background-color: white">
              <a href="/myWeb/question/findById?questionId=<%=question.getQuestionId()%>">详细信息</a>
          </td>
      </tr>
      <%
          }
      %>
  </table>
</body>
</html>
```

4 试题编号查询功能

```
public class QuestionFindByIdServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        String questionId;
        Question question = null;
        //1.调用请求对象读取请求头中参数信息，得到试题编号
        questionId = request.getParameter("questionId");
        //2.调用Dao推送查询命令得到这个试题编号对应的试题信息
        question = QuestionDao.findById(questionId);
        //3.调用question_update.jsp将试题信息写入到响应体
        request.setAttribute("key", question);
        request.getRequestDispatcher("/question_update.jsp").forward(request, response);
    }
}
```

```
public class QuestionDao {
    public static Question findById(String questionId) {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Question question = null;
        try {
            String sql = "select * from question where questionId=?";
            conn = JdbcUtil.getConnection();
            ps = conn.prepareStatement(sql);
            ps.setInt(1, Integer.valueOf(questionId));
            rs = ps.executeQuery();
            while(rs.next()){
                Integer quesId = rs.getInt("questionId");
                String title = rs.getString("title");
                String optionA = rs.getString("optionA");
                String optionB = rs.getString("optionB");
                String optionC = rs.getString("optionC");
                String optionD = rs.getString("optionD");
                String answer = rs.getString("answer");
                question = new Question(quesId, title, optionA, optionB, optionC, optionD, answer);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, rs);
        }
        return question;
    }
}
```

```
<%@ page import="com.example.entity.Question" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
  <title>Title</title>
</head>
<body>
<%
  Question q =(Question) request.getAttribute("key");
%>
<center>
  <form action="/myWeb/question/update" method="get">
    <table border="2">
      <tr>
        <td>题目编号:</td>
        <td><input type="text" name="questionId" value="<%=q.getQuestionId()%>" readOnly></td>
      </tr>
      <tr>
        <td>题目:</td>
        <td><input type="text" name="title" value="<%=q.getTitle()%>"></td>
      </tr>
      <tr>
        <td>A:</td>
        <td><input type="text" name="optionA" value="<%=q.getOptionA()%>"></td>
      </tr>
      <tr>
        <td>B:</td>
        <td><input type="text" name="optionB" value="<%=q.getOptionB()%>"></td>
      </tr>
      <tr>
        <td>C:</td>
        <td><input type="text" name="optionC" value="<%=q.getOptionC()%>"></td>
      </tr>
```

```
      <tr>
        <td>D:</td>
        <td><input type="text" name="optionD" value="<%=q.getOptionD()%>"></td>
      </tr>
      <tr>
        <td>正确答案:</td>
        <td>
          <input type="radio" name="answer" value="A" <%="A".equals(q.getAnswer())?"checked":""%> >A
          <input type="radio" name="answer" value="B" <%="B".equals(q.getAnswer())?"checked":""%> >B
          <input type="radio" name="answer" value="C" <%="C".equals(q.getAnswer())?"checked":""%> >C
          <input type="radio" name="answer" value="D" <%="D".equals(q.getAnswer())?"checked":""%> >D
        </td>
      </tr>
      <tr>
        <td><input type="submit" value="更新试题"/></td>
        <td><input type="reset" ></td>
      </tr>
    </table>
  </form>
</center>
</body>
</html>
```

5 试题更新功能

```
public class QuestionUpdateServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        String title,optionA,optionB,optionC,optionD,answer,questionId;
        Question question = null;
        int result = 0;
        //1.调用请求对象读取请求头参数信息
        title= request.getParameter("title");
        optionA= request.getParameter("optionA");
        optionB= request.getParameter("optionB");
        optionC = request.getParameter("optionC");
        optionD = request.getParameter("optionD");
        answer  = request.getParameter("answer");
        questionId = request.getParameter("questionId");
        //2.调用Dao实现更新操作
        question = new Question(Integer.valueOf(questionId), title, optionA, optionB, optionC, optionD,
answer);
        result = QuestionDao.update(question);
        //3.调用info.jsp将操作结果写入到响应体
        if(result == 1){
            request.setAttribute("info", "试题更新成功");
        }else{
            request.setAttribute("info", "试题更新失败");
```

```
        }
        request.getRequestDispatcher("/info.jsp").forward(request, response);
    }
}
```

```
public class QuestionDao {
    public static int update(Question question) {
        Connection conn = null;
        PreparedStatement ps = null;
        int res = 0;
        try {
            String sql = "update question set title=?, optionA=?, optionB=?, optionC=?, optionD=?,
answer=? where questionId=?";
            conn = JdbcUtil.getConnection();
            ps = conn.prepareStatement(sql);
            ps.setString(1, question.getTitle());
            ps.setString(2, question.getOptionA());
            ps.setString(3, question.getOptionB());
            ps.setString(4, question.getOptionC());
            ps.setString(5, question.getOptionD());
            ps.setString(6, question.getAnswer());
            ps.setInt(7, question.getQuestionId());
            res = ps.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, null);
        }
        return res;
    }
}
```

EL表达式

一、EL 工具包介绍

- 由Java技术开发一个jar包
- 作用降低JSP文件开发时Java命令开发强度
- Tomcat服务器本身自带了EL工具包（Tomcat安装地址/lib/el-api.jar）

二、JSP文件开发步骤

- JSP文件作用：代替 响应对象 将 Servlet 中 doGet/doPost 的执行结果写入到响应体
- JSP开发步骤

```
<%
    String value = (String) request.getAttribute("key");
%>
<%=value%>
```

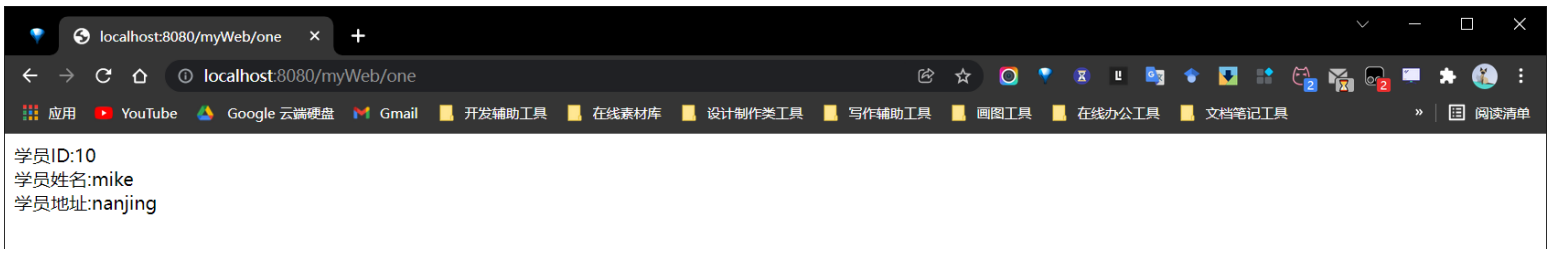
1. 从指定的作用域对象读取处理结果
2. 将得到数据进行类型强转
3. 将转换后的数据写入到响应体

```
public class OneServlet extends HttpServlet {//  /myWeb/one
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.分别将共享数据添加到作用域对象
        ServletContext application = request.getServletContext();
        HttpSession session = request.getSession();

        application.setAttribute("sid", 10);
        session.setAttribute("sname", "mike");
        request.setAttribute("home", "nanjing");

        //2.通过请求转发方式，向Tomcat申请调用index_1.jsp，由index_1.jsp负责将
        // 作用域对象共享数据读取并写入到响应体，交给浏览器
        request.getRequestDispatcher("/index_1.jsp").forward(request, response);
    }
}
```

```
<!--index_1.jsp-->
<%--
  Created by IntelliJ IDEA.
  User: Amadeus
  Date: 2022/1/8
  Time: 16:50
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
    Integer sid = (Integer) application.getAttribute("sid");
    String sname = (String) session.getAttribute("sname");
    String home = (String) request.getAttribute("home");
%>
学员ID:<%=sid%><br>
学员姓名:<%=sname%><br>
学员地址:<%=home%>
```

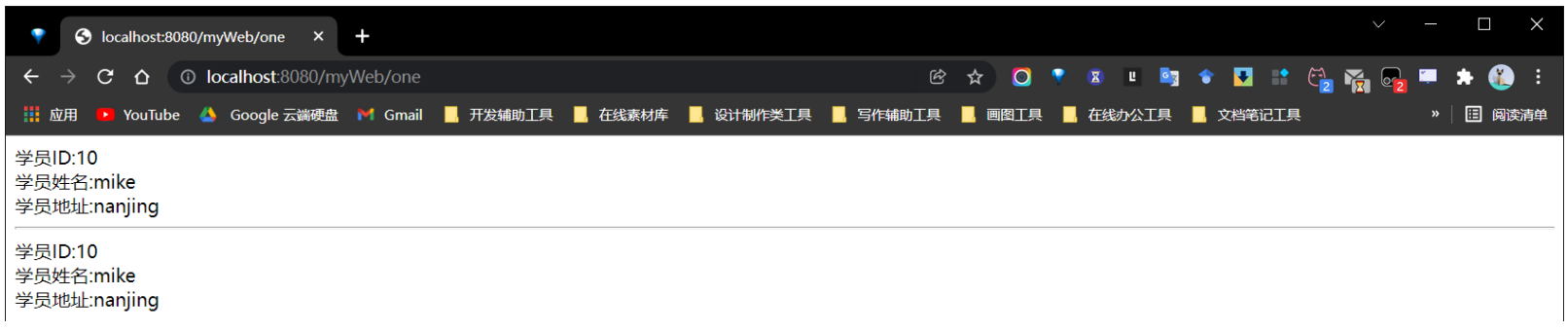


三、EL表达式

```
<%--
  Created by IntelliJ IDEA.
  User: Amadeus
  Date: 2022/1/8
  Time: 16:50
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
    Integer sid = (Integer) application.getAttribute("sid");
```

```
    String sname = (String) session.getAttribute("sname");
    String home = (String) request.getAttribute("home");
%>
学员ID:<%=sid%><br>
学员姓名:<%=sname%><br>
学员地址:<%=home%>

<hr/>
学员ID:${applicationScope.sid}<br>
学员姓名:${sessionScope.sname}<br>
学员地址:${requestScope.home}<br>
```



1. 命令格式：${作用域对象别名.共享数据名}
2. 命令作用：
    - EL表达式是EL工具包提供一种特殊命令格式【表达式命令格式】
    - EL表达式在JSP文件上使用
    - 负责在JSP文件上从作用域对象读取指定的共享数据并输出到响应体

四、EL表达式 --- 作用域对象别名

1. JSP文件可以使用的作用域对象
    - ServletContext　　application: 全局作用域对象
    - HttpSession　　session:　　会话作用域对象
    - HttpServletRequest　　request:　　请求作用域对象
    - PageContext　　pageContext: 当前页作用域对象，这是JSP文件独有的作用域对象；Servlet中不存在。在当前页作用域对象存放的共享数据仅能在当前JSP文件中使用，不能共享给其他Servlet或则其他JSP文件。

      真实开发过程，主要用于JSTL标签与JSP文件之间数据共享数据。

      JSTL------->pageContext---->JSP
2. EL表达式提供作用域对象别名

JSP	EL表达式
application	${applicationScope.共享数据名}
session	${sessionScope.共享数据名}
request	${requestScope.共享数据名}
pageContext	${pageScope.共享数据名}

五、EL表达式将引用对象属性写入到响应体

1. 命令格式：${作用域对象别名.共享数据名.属性名}
2. 命令作用：从作用域对象读取指定共享数据关联的引用对象的属性值；并自动将属性的结果写入到响应体
3. 属性名：一定要去引用类型属性名完全一致（区分大小写）
4. EL表达式没有提供遍历集合方法，因此无法从作用域对象读取集合内容输出

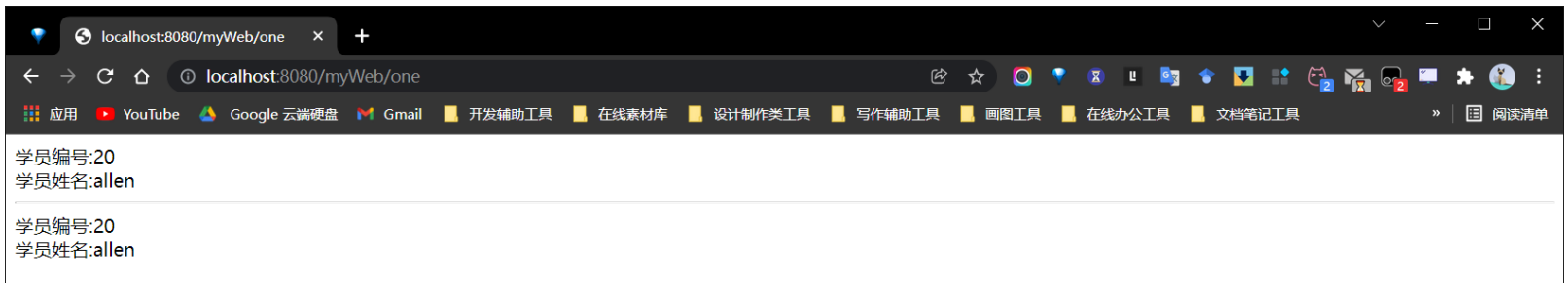
```
public class Student {
    private Integer sid;
    private String sname;
    public Student() {
    }
    public Student(Integer sid, String sanem) {
        this.sid = sid;
        this.sname = sanem;
    }
    public Integer getSid() {
        return sid;
    }
    public void setSid(Integer sid) {
        this.sid = sid;
    }
    public String getSname() {
        return sname;
    }
    public void setSname(String sname) {
        this.sname = sname;
    }
```

```
}
```

```
public class OneServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.创建一个引用类型实例对象
        Student stu = new Student(20, "allen");
        //2.将引用类型对象存入到请求作用域对象作为共享数据
        request.setAttribute("key", stu);
        //3.请求转发，向Tomcat申请调用index_1.jsp
        request.getRequestDispatcher("/index_1.jsp").forward(request, response);
    }
}
```

```
<%@ page import="com.example.entity.Student" %><%--
  Created by IntelliJ IDEA.
  User: Amadeus
  Date: 2022/1/8
  Time: 17:32
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--传统写法-->
<%
  Student stu = (Student) request.getAttribute("key");
%>
学员编号:<%=stu.getSid()%><br>
学员姓名:<%=stu.getSname()%>

<hr>
<!--EL表达式-->
学员编号:${requestScope.key.sid}<br>
学员姓名:${requestScope.key.sname}
```



六、EL表达式简化版

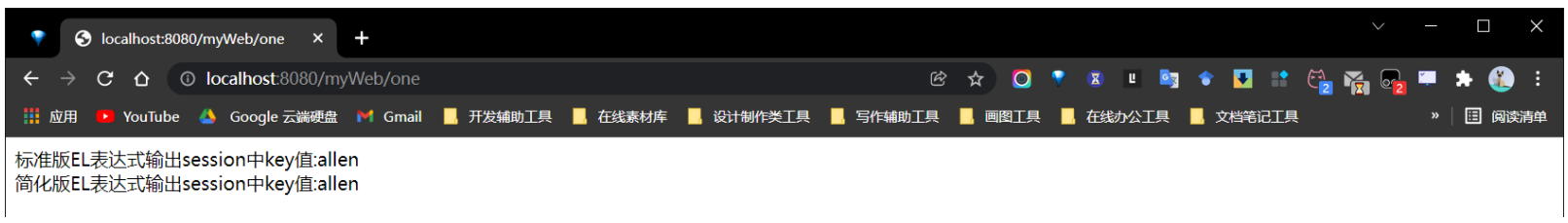
1. 命令格式： ${共享数据名}
2. 命令作用： EL表达式允许开发人员开发时省略作用域对象别名
3. 工作原理：EL表达式简化版由于没有指定作用域对象，所以在执行时采用【猜】算法
    - 首先到【pageContext】定位共享数据，如果存在直接读取输出并结束执行
    - 如果在【pageContext】没有定位成功，到【request】定位共享数据，如果存在直接读取输出并结束执行
    - 如果在【request】没有定位成功，到【session】定位共享数据，如果存在直接读取输出并结束执行
    - 如果在【session】没有定位成功，到【application】定位共享数据，如果存在直接读取输出并结束执行
    - 如果在【application】没有定位成功，返回null

      当前页作用域对象--->请求--->会话--->全局
4. 存在隐患：
    - 容易降低程序执行速度【南辕北辙】
    - 容易导致数据定位错误
5. 应用场景：设计目的，就是简化从pageContext读取共享数据并输出难度
6. EL表达式简化版尽管存在很多隐患，但是在实际开发过程中，开发人员为了节省时间，一般都使用简化版，拒绝使用标准版

```
public class OneServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.向当前用户session添加一个共享数据
        HttpSession session = request.getSession();
        session.setAttribute("key", "allen");

        //2.请求转发，申请调用index_1.jsp
        request.getRequestDispatcher("/index_1.jsp").forward(request, response);
    }
}
```

```jsp
<!--index_1.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
标准版EL表达式输出session中key值:${sessionScope.key}<br>
简化版EL表达式输出session中key值:${key}
```

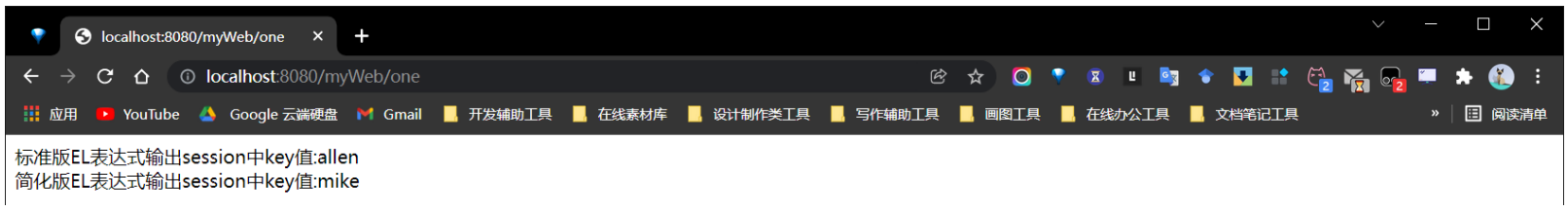



```java
public class OneServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.向当前用户session添加一个共享数据
        HttpSession session = request.getSession();
        session.setAttribute("key", "allen");

        //2.向当前请求作用域对象添加一个共享数据
        request.setAttribute("key", "mike");

        //3.请求转发，申请调用index_1.jsp
        request.getRequestDispatcher("/index_1.jsp").forward(request, response);
    }
}
```

```jsp
<!--index_1.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
标准版EL表达式输出session中key值:${sessionScope.key}<br>
简化版EL表达式输出session中key值:${key}
```



七、EL表达式-----支持运算表达式

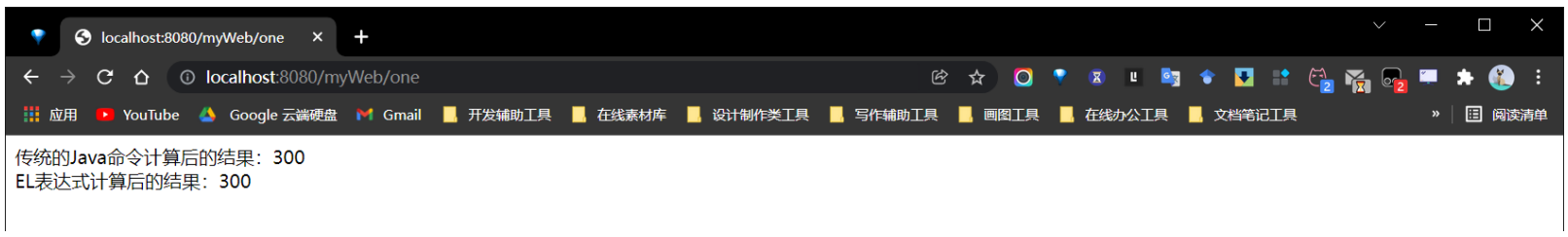
1. 前提：在JSP文件有时需要将读取共享数据进行一番运算之后，将运算结果写入到响应体
2. 运算表达式:
    - 数学运算
    - 关系运算

>	>=	==	<	<=	!=
gt	ge	eq	lt	le	!=

    - 逻辑运算：&&　||　!

```java
public class OneServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        request.setAttribute("key1", "100");
        request.setAttribute("key2", 200);
        request.getRequestDispatcher("index_1.jsp").forward(request, response);
    }
}
```

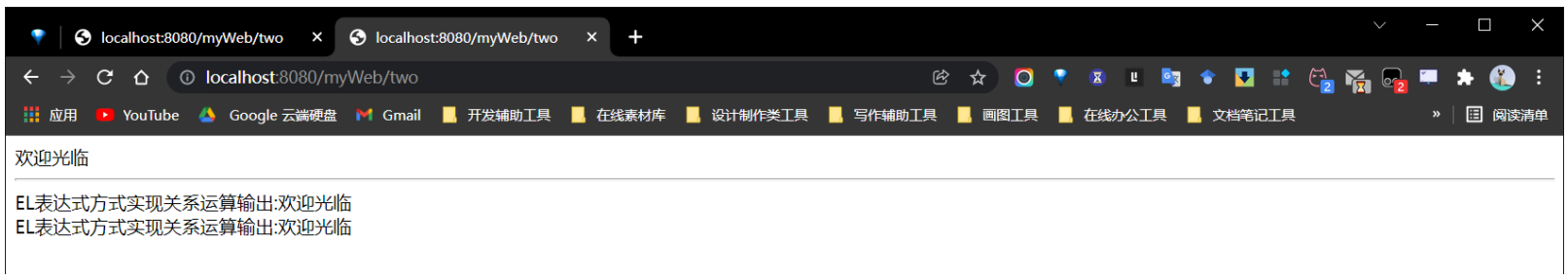
```jsp
<!--index_1.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--将作用域对象中共享数据读取出来相加，将相加结果写入到响应体-->
<%
    String num1 = (String) request.getAttribute("key1");
    Integer num2 = (Integer) request.getAttribute("key2");
    int sum = Integer.valueOf(num1) + num2;
%>
传统的Java命令计算后的结果：<%=sum%><br>
EL表达式计算后的结果：${key1+key2}
```




```java
public class TwoServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        request.setAttribute("age", "25");
        request.getRequestDispatcher("index_2.jsp").forward(request, response);
    }
}
```

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--传统Java命令方式实现关系运算输出-->
<%
  String age = (String) request.getAttribute("age");
  if(Integer.valueOf(age) >= 18){
%>
      欢迎光临<br>
<%
  } else {
%>
      谢绝入内<br>
<%
  }
%>

<hr>
EL表达式方式实现关系运算输出:${age >= 18 ? "欢迎光临" : "谢绝入内"}<br>
EL表达式方式实现关系运算输出:${age ge 18 ? "欢迎光临" : "谢绝入内"}
```



八、EL表达式提供内置对象

1 param

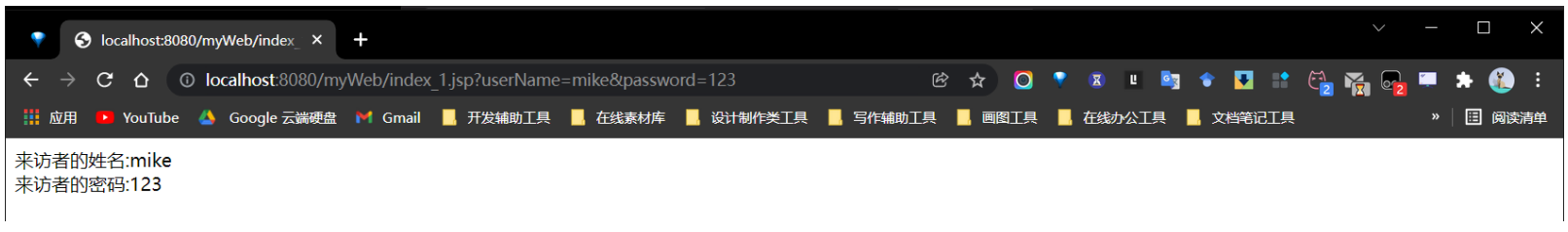
1. 命令格式: ${param.请求参数名}
2. 命令作用:
   - 通过请求对象读取当前请求包中请求参数内容
   - 并将请求参数内容写入到响应体
3. 代替命令

   index_1.jsp

   发送请求：Http://localhost:8080/myWeb/index_1.jsp?userName=mike&password=123

```jsp
<%
    String userName =   request.getParameter("userName");
    String password =   request.getParameter("password");
%>
<%=userName%>
<%=password%>
```

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--
    Http://localhost:8080/myWeb/index_1.jsp?userName=mike&password=123
-->
来访者的姓名:${param.userName}<br>
来访者的密码:${param.password}
```



2 paramValues

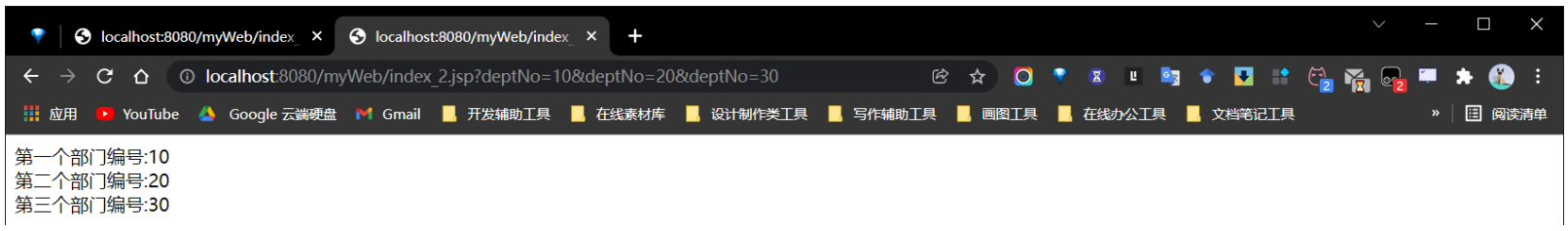
1. 命令格式：${paramValues.请求参数名[下标]}
2. 命令作用:
   - 如果浏览器发送的请求参数是[一个请求参数关联多个值]，此时可以通过paramVaues读取请求参数下指定位置的值，并写入到响应体
3. 代替命令：

   http://localhost:8080/myWeb/index_2.jsp?pageNo=1&pageNo=2&pageNo=3

   此时pageNo请求参数在请求包以数组形式存在

```
pageNo:[1,2,3]
<%
    String array[]= request.getParameterValues("pageNo");
%>
第一个值:<%=array[0]%>
第二个值:<%=array[1]%>
```

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!--
http://localhost:8080/myWeb/index_2.jsp?deptNo=10&deptNo=20&deptNo=30
-->
第一个部门编号:${paramValues.deptNo[0]}<br>
第二个部门编号:${paramValues.deptNo[1]}<br>
第三个部门编号:${paramValues.deptNo[2]}<br>
```



九、应用

在线考试管理系统---通过questionId显示试题详细信息EL版

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
  <title>EL_Question_Update</title>
</head>
<body>
<center>
  <form action="/myWeb/question/update" method="get">
    <table border="2">
      <tr>
        <td>题目编号:</td>
        <td><input type="text" name="questionId" value="${requestScope.key.questionId}" readOnly></td>
      </tr>
      <tr>
        <td>题目:</td>
        <td><input type="text" name="title" value="${requestScope.key.title}"></td>
      </tr>
      <tr>
        <td>A:</td>
        <td><input type="text" name="optionA" value="${requestScope.key.optionA}"></td>
      </tr>
      <tr>
        <td>B:</td>
        <td><input type="text" name="optionB" value="${requestScope.key.optionB}"></td>
      </tr>
      <tr>
        <td>C:</td>
        <td><input type="text" name="optionC" value="${requestScope.key.optionC}"></td>
      </tr>
      <tr>
        <td>D:</td>
        <td><input type="text" name="optionD" value="${requestScope.key.optionD}"></td>
      </tr>
      <tr>
        <td>正确答案:</td>
        <td>
          <input type="radio" name="answer" value="A" ${"A" eq requestScope.key.answer ? "checked" : ""}
>A
          <input type="radio" name="answer" value="B" ${"B" eq requestScope.key.answer ? "checked" : ""}
>B
          <input type="radio" name="answer" value="C" ${"C" eq requestScope.key.answer ? "checked" : ""}
>C
          <input type="radio" name="answer" value="D" ${"D" eq requestScope.key.answer ? "checked" : ""}
>D
```

```
        </td>
      </tr>
      <tr>
        <td><input type="submit" value="更新试题"/></td>
        <td><input type="reset" ></td>
      </tr>
    </table>
  </form>
</center>
</body>
</html>
```

十、EL表达式常见异常

```
javax.el.PropertyNotFoundException;//在对象中没有找到指定属性
```

十一、在线考试管理系统

```
开发任务：随机出题
任务描述：用户点击【参加考试】，系统【随机】提取四道考试题，交给用户
开发任务：在线阅卷
        1.根据用户提供答案与正确答案比较得到用户分数
        2.将用户分数交给scor.jsp做输出
```

1 自动出题

```
#查询职员信息，按照部门编号排序，截取前两个
select * from emp order by deptno desc limit 0,2;
#查询职员信息，按照职员编号倒叙，截取前两个
select * from emp order by empno asc limit 0,2;
#查询职员信息，按照工资倒叙，截取前两个
select * from emp order by sal asc limit 0,2;

#如果每次查询时，排序字段不同，然后在截取得到数据行内容往往不一样
select rang();#随机返回0---1之间小数
#rand() 0.5--->5  order by 5
# 0.9--->order by 1
select * from question order by rand() limit 0,4;
```

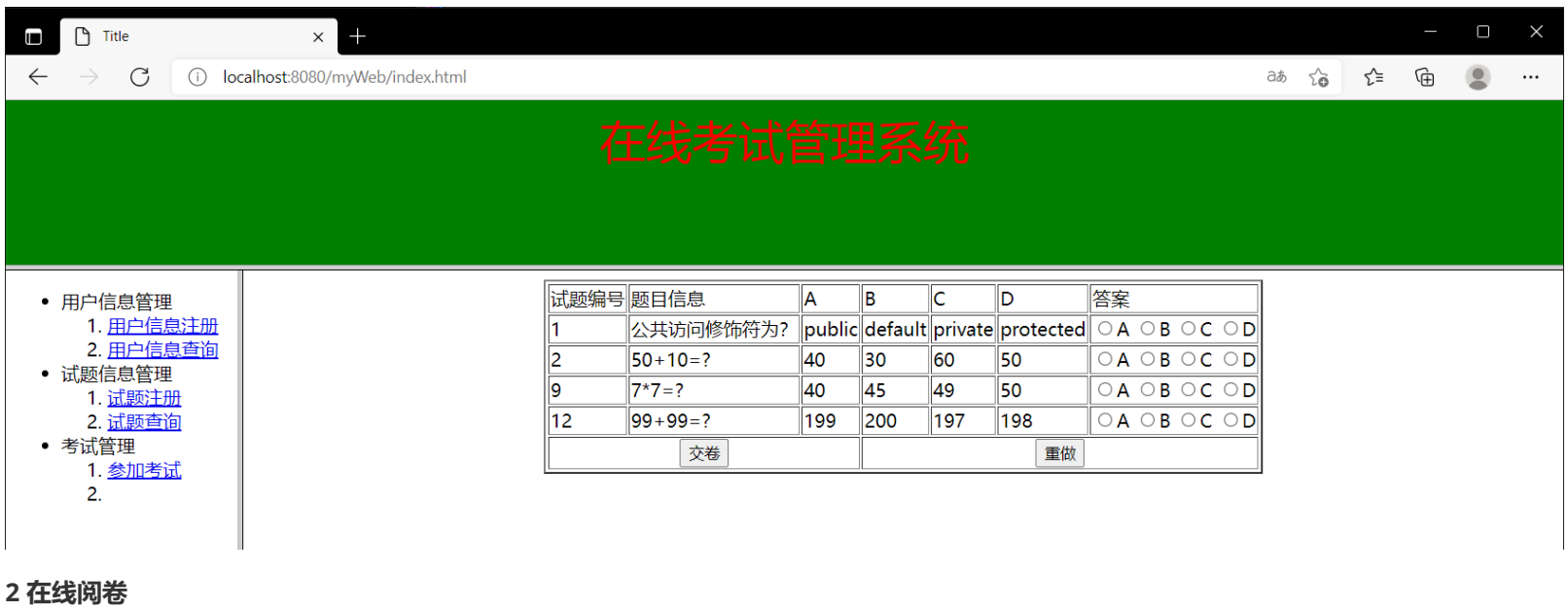
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
  <ul>
    <li>用户信息管理
      <ol>
        <li><a href="/myWeb/user_Add.html" target="right">用户信息注册</a> </li>
        <li><a href="/myWeb/user/find" target="right">用户信息查询</a></li>
      </ol>
    </li>
    <li>试题信息管理
      <ol>
        <li><a href="/myWeb/question_Add.html" target="right">试题注册</a> </li>
        <li><a href="/myWeb/question/find" target="right">试题查询</a></li>
      </ol>
    </li>
    <li>考试管理
      <ol>
        <li><a href="/myWeb/question/rand" target="right">参加考试</a> </li>
        <li><a href="/myWeb/question/find" target="right"></a></li>
      </ol>
    </li>
  </ul>
</body>
</html>
```

```java
public class QuestionRandServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //1.调用Dao对象随机从question表拿出4道题目
        List<Question> list = QuestionDao.findRand();
        //2.将4道题目添加到request作为共享数据
        request.setAttribute("key", list);
        //3.请求转发，申请exam.jsp将4道题目写入到响应体
        request.getRequestDispatcher("/exam.jsp").forward(request, response);
    }
}
```

```java
public class QuestionDao {
    public static List<Question> findRand(){
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        List<Question> questionList = new ArrayList<>();
        try {
            String sql = "select * from question order by rand() limit 0,4";
            conn = JdbcUtil.getConnection();
            ps = conn.prepareStatement(sql);
            rs = ps.executeQuery();
            while(rs.next()){
                Integer questionId = rs.getInt("questionId");
                String title = rs.getString("title");
                String optionA = rs.getString("optionA");
                String optionB = rs.getString("optionB");
                String optionC = rs.getString("optionC");
                String optionD = rs.getString("optionD");
                String answer = rs.getString("answer");
                Question question = new Question(questionId, title, optionA, optionB, optionC, optionD, answer);
                questionList.add(question);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, rs);
        }
        return questionList;
    }
}
```

```jsp
<!--exam.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <center>
        <form action="/myWeb/exam">
            <table border="2">
                <tr>
                    <td>试题编号</td>
                    <td>题目信息</td>
                    <td>A</td>
                    <td>B</td>
                    <td>C</td>
                    <td>D</td>
                    <td>答案</td>
                </tr>
                <%
                   List<Question> questionList = (List<Question>) request.getAttribute("key");
                   for(Question question : questionList){
                %>
                    <tr>
                        <td><%=question.getQuestionId()%></td>
                        <td><%=question.getTitle()%></td>
                        <td><%=question.getOptionA()%></td>
                        <td><%=question.getOptionB()%></td>
                        <td><%=question.getOptionC()%></td>
                        <td><%=question.getOptionD()%></td>
                        <td>
                            <input type="radio" name="answer_<%=question.getQuestionId()%>" value="A">A
                            <input type="radio" name="answer_<%=question.getQuestionId()%>" value="B">B
                            <input type="radio" name="answer_<%=question.getQuestionId()%>" value="C">C
                            <input type="radio" name="answer_<%=question.getQuestionId()%>" value="D">D
                        </td>
                    </tr>
```

```
                <%
                    }
                %>
                <tr align="center">
                    <td colspan="3"><input type="submit" value="交卷"></td>
                    <td colspan="4"><input type="reset" value="重做"></td>
                </tr>
            </table>
        </form>
    </center>
</body>
</html>
```



2 在线阅卷

```
public class ExamServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //分数
        int score = 0;
        //1.从当前用户私人储物柜得到系统提供四道题目信息
        HttpSession session = request.getSession(false);
        List<Question> questionList = (List<Question>) session.getAttribute("key");
        //2.从请求包读取用户对于四道题目给出答案
        for (Question question : questionList){
            String answer = question.getAnswer();
            Integer questionId = question.getQuestionId();
            String userAnswer = (String) request.getParameter("answer_" + questionId);
            //3.判分
            if(userAnswer.equals(answer)){
                score += 25;
            }
        }
        //4.将分数写入到request中，作为共享数据
        request.setAttribute("info", "本次考试成绩: " + score);
        //5.请求转发调用jsp将用户本次考试分数写入到响应体
        request.getRequestDispatcher("info.jsp").forward(request, response);
    }
}
```


Title
localhost:8080/myWeb/index.html
在线考试管理系统
用户信息管理
1. 用户信息注册
2. 用户信息查询
试题信息管理
1. 试题注册
2. 试题查询
考试管理
1. 参加考试
2.
本次考试成绩: 100

MVC开发规则

一、介绍

1. MVC开发规则制定了互联网通信开发过程中必须出现的角色有哪些
2. MVC开发规则制定了互联网通信开发过程中必须出现的角色所要担负的职责
3. MVC开发规则制定了互联网通信开发过程中必须出现角色的出场顺序

二、角色

1. DAO对象：DAO对象提供某张表文件的操作细节，降低对表文件操作难度；避免反复开发表文件操作的代码，提高代码复用性
2. Service对象：服务对象，提供【业务】的具体解决方案；service对象一个方法指定一个业务的解决方案；避免业务开发重复性开发行为，提供复用性；网站每一个业务都有一个独立标准解决方案

三、业务

浏览器向Http服务器发送请求

用户向网站发送请求

举个例子：张三用户发送请求：要求在服务端实现将张三账户3000元钱转给李四账户

业务处理方案:

1. 判断"张三"是否是当前系统中用户；
2. 判断"李四"是否是当前系统中用户；
3. 读取"张三账户余额"，判断余额是否充足；
4. 读取"李四账户余额"，背账；
5. 更新"张三账户余额 -3000"；
6. 更新 "李四账户余额 +3000 。

四、业务特征

1. 真实业务场景中，一个业务往往包含多个分支任务；因此解决业务开发工作量往往比较巨大
2. 真实业务场景中，只有所有分支任务都能顺利成功解决，才可以认为当前业务处理成功

五、解决业务开发困恼

1. 一个业务可能在网站的多个地方重复出现，如果不做【封装】，增加开发难度，进行业务解决代码重复性开发
2. 【百人有百味】，不同程序员面对同一个业务时，给出解决方案往往有偏差，导致最终解决数据会有偏差

六、MVC开发规则----互联网通信开发过程中必须出现角色有哪些

一次互联网开发过程，必须出现角色有三个:

1. C contorller object； 控制层对象 （servlet对象）
2. M model object； 业务模型对象 （Service对象）
3. V, view object； 视图层对象 (jsp or HttpServletResponse)

七、MVC开发规则------互联网通信开发过程中必须出现角色担负职责

1. C（servlet对象）：
    - 【可以】调用【请求对象】读取【请求包】参数信息
    - 【必须】调用【Service对象】处理业务
    - 【必须】调用【 视图层对象】将结果写入到响应体
2. M（service对象）：
    - 处理业务中所有分支任务
    - 根据分支任务执行情况判断业务是否处理成功
    - 必须通过return将处理结果返回给【控制层对象】
3. V（jsp/HttpServletResponse）：
    - 【禁止参与业务处理】
    - 唯一任务将处理结果写入到响应体

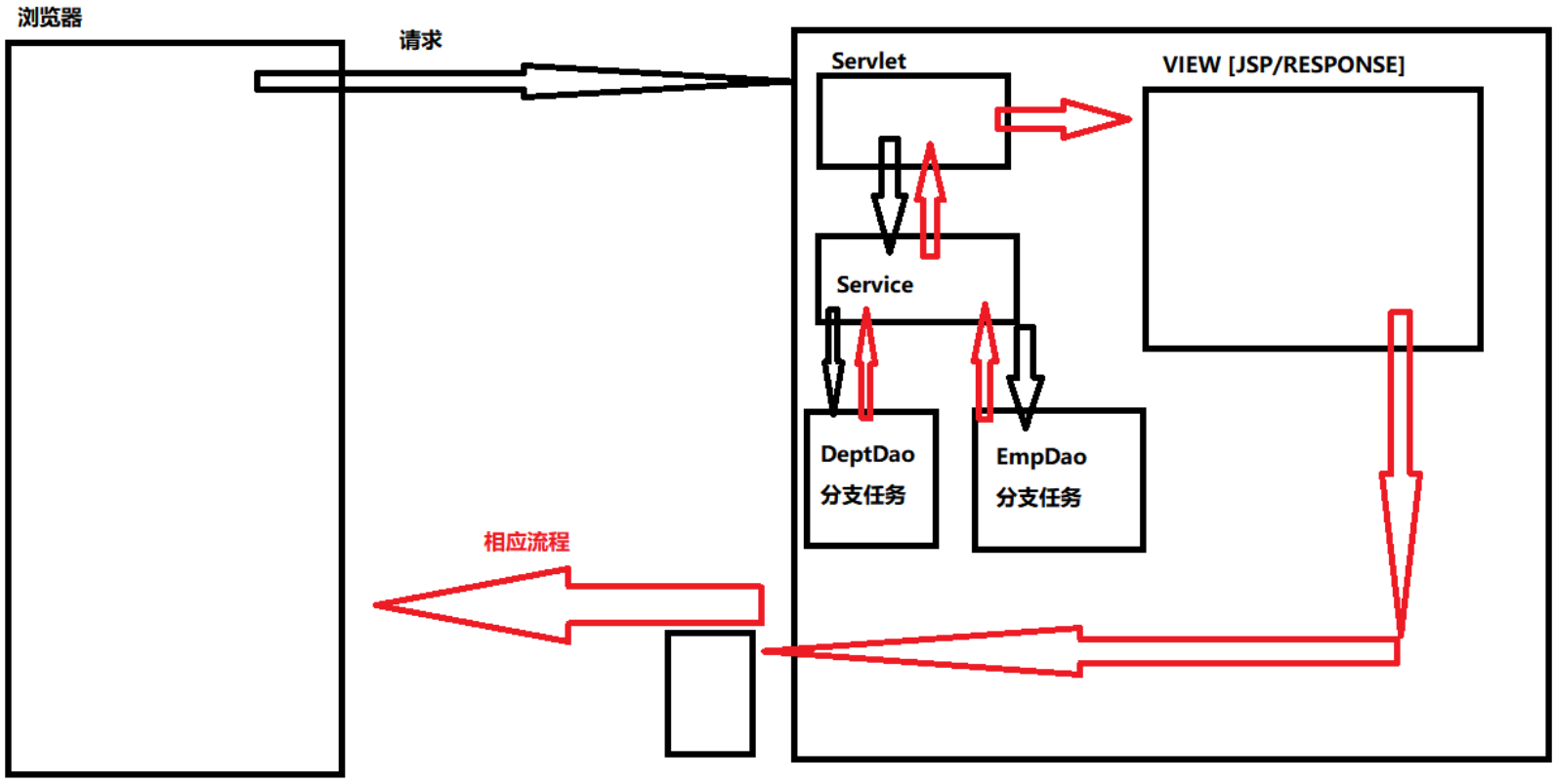
八、互联网通信开发过程中必须出现角色的出场顺序

请求调用顺序:

浏览器---发送请求--->Servlet------>Service------>DeptDao/EmpDao

响应顺序:

DeptDao/EmpDao---分支任务结果--->Service------>Servlet------>View------>响应体---(Tomcat负责推送)--->浏览器


浏览器
请求
Servlet
VIEW [JSP/RESPONSE]
Service
DeptDao
分支任务
EmpDao
分支任务
相应流程

AJAX

一、全局刷新和局部刷新

全局刷新：整个浏览器被新的数据覆盖；在网络中传输大量的数据；浏览器需要加载，渲染页面。

全局刷新原理：

1. 必须由浏览器亲自向服务端发送请求协议包
2. 这个行为导致服务端直接将【响应包】发送到浏览器内存中
3. 这个行为导致浏览器内存中原有内容被覆盖掉
4. 这个行为导致浏览器在展示数据时候，只有响应数据可以展示

局部刷新：在浏览器器的内部，发起请求，获取数据，改变页面中的部分内容；其余的页面无需加载和渲染。 网络中数据传输量少， 给用户的感受好。

局部刷新原理：

1. 不能由浏览器发送请求给服务端
2. 浏览器委托浏览器内存中一个脚本对象代替浏览器发送请求
3. 这个行为导致导致服务端直接将【响应包】发送脚本对象内存中
4. 这个行为导致脚本对象内容被覆盖掉，但是此时浏览器内存中绝大部分内容没有受到任何影响
5. 这个行为导致浏览器在展示数据时候,同时展示原有数据和响应数据

ajax是用来做局部刷新的。局部刷新使用的核心对象是 异步对象（XMLHttpRequest）
这个异步对象是存在浏览器内存中的， 使用javascript语法创建和使用XMLHttpRequest对象。

二、AJAX

1 什么是AJAX

- AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。
- AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分页面内容的新方法
- AJAX 不是新的编程语言，而是使用现有技术混合使用的一种新方法。
- AJAX 中使用的技术有 JavaScript、html、dom、xml、css 等。主要是 JavaScript、XML
- JavaScript：使用脚本对象 XMLHttpRequest 发送请求，接收响应数据
- XML：发送和接收的数据格式，现在使用 JSON
- AJAX 不单需要前端的技术，同时需要后端（服务器）的配合。服务器需要提供数据，数据是 AJAX 请求的响应结果。

2 AJAX 异步实现步骤

1. 创建异步对象方式

```
var xmlHttp = new XMLHttpRequest();
```

2. 给异步对象绑定事件
   - onreadystatechange：当异步对象发起请求，获取了数据都会触发这个事件。这个事件需要指定一个函数， 在函数中处理状态的变化。

```
btn.onclick = fun1()
    function fun1(){
     alert("按钮单击");
    }

    //例如：
    xmlHttp.onreadystatechange= function(){
          //处理请求的状态变化。
         if(xmlHttp.readyState == 4 && xmlHttp.status== 200 ){
               //可以处理服务器端的数据，更新当前页面
              var data = xmlHttp.responseText;
              document.getElementById("name").value= data;
         }
    }
```

   - 异步对象的属性 readyState 表示异步对象请求的状态变化
     - 0：创建异步对象时； new XMLHttpRequest();
     - 1：初始化异步请求对象； xmlHttp.open(请求方式，请求地址，true)
     - 2：异步对象发送请求；xmlHttp.send()
     - 3：异步对象接收应答数据；从服务端返回数据。XMLHttpRequest 内部处理。
     - 4：异步请求对象已经将数据解析完毕；此时才可以读取数据。（在4的时候，开发人员做什么？更新当前页面。）
   - status属性：表示网络请求的状况的
     - 200："OK" | "通信成功"
     - 404：未找到页面
     - 500：服务器端代码出错
3. 初始化请求参数
   - 异步的方法open()
   - open(method, url, async)： 初始化异步请求对象

- method：请求的类型；GET 或 POST
- url：服务器端的访问地址（例如servlet地址）
- async：true（异步）或 false（同步）

4. 发送请求

```
xmlHttp.send();
```

5. 获取服务器端返回的数据
   - 如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。
   - responseText：获得字符串形式的响应数据
   - responseXML：获得 XML 形式的响应数据
   - 回调：当请求的状态变化时，异步对象会自动调用onreadystatechange事件对应的函数

三、AJAX实例一

1 全局刷新计算BMI

- 需求：计算某个用户的 BMI，用户在JSP输入自己的身高，体重；servlet中计算 BMI，并显示 BMI 的计算结果和建议。
- BMI 指数（即身体质量指数，英文为 BodyMassIndex，简称 BMI），是用 体重公斤数 / 身高米数平方 得出的数字，是目前国际上常用的衡量人体胖瘦程度以及是否健康的一个标准
- 成人的 BMI 数值：
  - 过轻：低于18.5
  - 正常：18.5 - 23.9
  - 过重：24 - 27
  - 肥胖：28 - 32
  - 非常肥胖：高于32

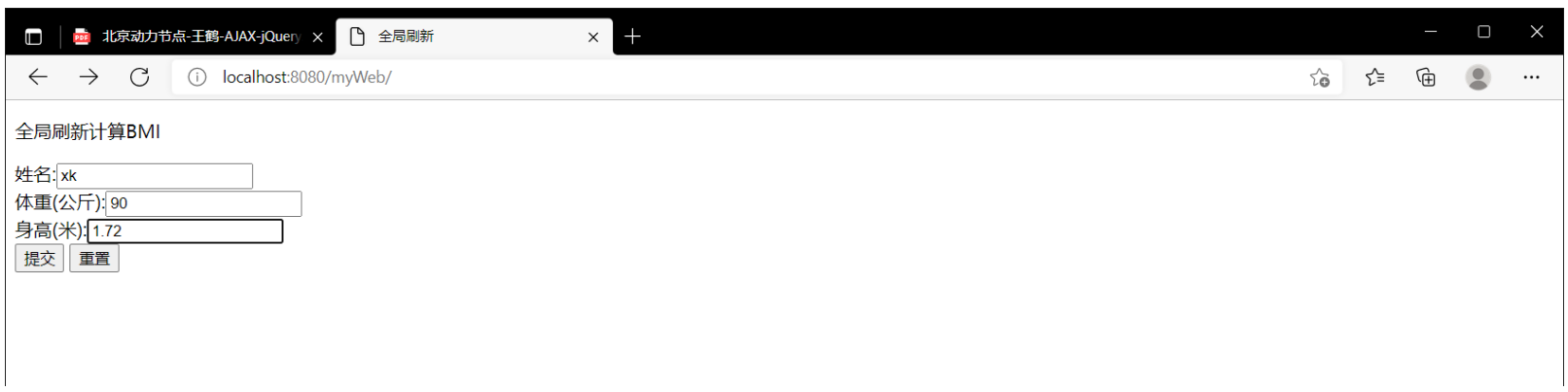
```
<!--index.jsp-->
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>全局刷新</title>
</head>
<body>
    <p>全局刷新计算BMI</p>
    <form action="/myWeb/bmi" method="get">
        姓名:<input type="text" name="name"><br>
        体重(公斤):<input type="text" name="weight"><br>
        身高(米):<input type="text" name="height"><br>
        <input type="submit" value="提交">
        <input type="reset" value="重置">
    </form>
</body>
</html>
```

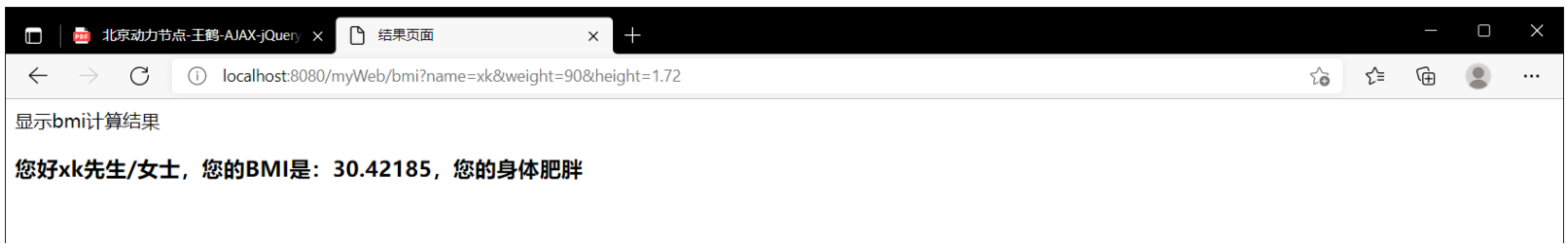
```
//servlet
public class BMIServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.接收请求参数
        String name = (String) request.getParameter("name");
        String weight = (String) request.getParameter("weight");
        String height = (String) request.getParameter("height");
        //2.计算BMI: bmi = weight / (height * height);
        float w = Float.valueOf(weight);
        float h = Float.valueOf(height);
        float bmi = w / (h * h);
        //3.判断BMI的范围
        String msg = "";
        if(bmi <= 18.5){
            msg = "您比较瘦";
        } else if(bmi <= 23.9){
            msg = "您的BMI是正常的";
        } else if(bmi <= 27){
            msg = "您的身体过重";
        } else if (bmi <= 32){
            msg = "您的身体肥胖";
        } else {
            msg = "您的身体过于肥胖";
        }
        System.out.println("msg = " + msg);
        msg = "您好" + name + "先生/女士，您的BMI是: " + bmi + "，" + msg;
        System.out.println(msg);
        //4.把数据存入到request
        request.setAttribute("msg", msg);
        //5.转发到新的页面
        request.getRequestDispatcher("/result.jsp").forward(request, response);
```

```
    }
}
```

```
<!--result.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>结果页面</title>
</head>
<body>
  <p>显示bmi计算结果</p>
  <h3>${requestScope.msg}</h3>
</body>
</html>
```

```
<!--web.xml-->
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">
    <servlet>
        <servlet-name>BMIServlet</servlet-name>
        <servlet-class>com.example.BMIServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>BMIServlet</servlet-name>
        <url-pattern>/bmi</url-pattern>
    </servlet-mapping>
</web-app>
```



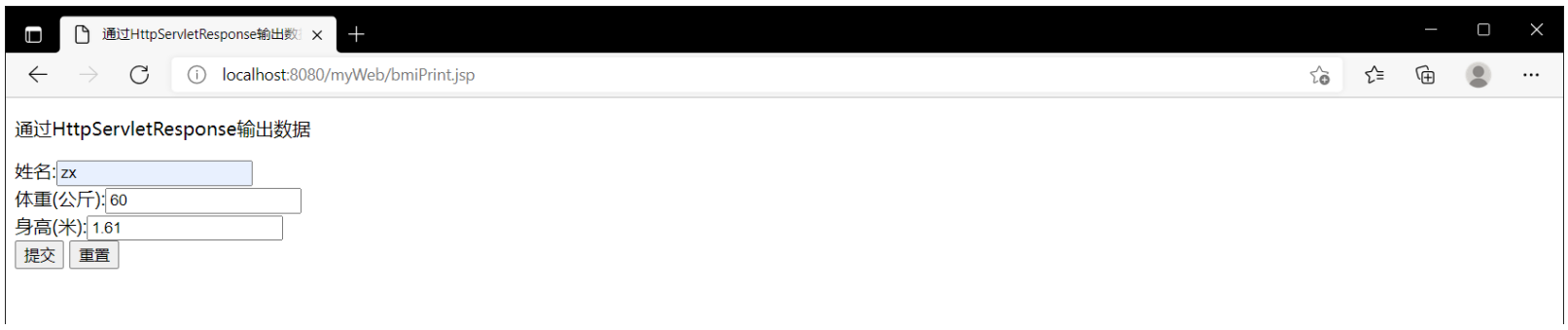


2 通过HttpServletResponse输出数据

```
<!--bmiPrint.jsp-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
  <title>通过HttpServletResponse输出数据</title>
</head>
<body>
<p>通过HttpServletResponse输出数据</p>
<form action="/myWeb/bmiprint" method="get">
  姓名:<input type="text" name="name"><br>
  体重(公斤):<input type="text" name="weight"><br>
  身高(米):<input type="text" name="height"><br>
  <input type="submit" value="提交">
  <input type="reset" value="重置">
</form>
</body>
</html>
```

```
//通过HttpServletResponse输出数据
public class BMIPrintServlet extends HttpServlet {// /bmiprint
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //1.接收请求参数
        String name = (String) request.getParameter("name");
        String weight = (String) request.getParameter("weight");
        String height = (String) request.getParameter("height");
```

```
        //2.计算BMI: bmi = weight / (height * height);
        float w = Float.valueOf(weight);
        float h = Float.valueOf(height);
        float bmi = w / (h * h);
        //3.判断BMI的范围
        String msg = "";
        if(bmi <= 18.5){
            msg = "您比较瘦";
        } else if(bmi <= 23.9){
            msg = "您的BMI是正常的";
        } else if(bmi <= 27){
            msg = "您的身体过重";
        } else if (bmi <= 32){
            msg = "您的身体肥胖";
        } else {
            msg = "您的身体过于肥胖";
        }
        System.out.println("msg = " + msg);
        msg = "您好" + name + "先生/女士，您的BMI是：" + bmi + "，" + msg;
        System.out.println(msg);
        //4.使用HttpServletResponse输出数据
        response.setContentType("text/html;charset=utf-8");
        PrintWriter writer = response.getWriter();//获取PrintWriter
        writer.println(msg);//输出数据
        writer.flush();//清空缓存
        writer.close();//关闭close
    }
```



3 使用AJAX请求，计算BMI

```
使用AJAX的局部刷新
1.新建JSP，使用XMLHttpResquest异步对象
    使用异步对象有四个步骤：
        1.创建
        2.绑定事件
        3.初始请求
        4.发送请求
2.创建服务器的Servlet，接收并处理数据，把数据输出给异步对象。
```

```
<!--index.jsp-->
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
  <title>局部刷新-AJAX</title>
  <script type="text/javascript">
    //使用内存中的异步对象，代替浏览器发起请求。异步对象使用js创建和管理的
    function doAjax(){
      //1.创建异步对象
      var xmlHttp = new XMLHttpRequest();
      //2.绑定事件
      xmlHttp.onreadystatechange = function (){
        //处理服务端返回的数据，更新当前页面
        // alert("readyState属性值=====" + xmlHttp.readyState + "| status = " + xmlHttp.status)
        if(xmlHttp.readyState == 4 && xmlHttp.status == 200){
          // alert(xmlHttp.responseText)
          var data = xmlHttp.responseText;
          document.getElementById("myData").innerText = data;
        }
      }
      //3.初始请求数据
      //获取dom对象的value属性值
      var name = document.getElementById("name").value;
      var weight = document.getElementById("weight").value;
```

```
      var height = document.getElementById("height").value;
      //bmiPrint?name=李四&weight=82&height=1.8
      var param = "name=" + name + "&weight=" + weight + "&height=" + height;
      // alert(param);
      xmlHttp.open("get", "bmiAjax?"+param, true);
      //4.发起请求
      xmlHttp.send();
    }
  </script>
</head>
<body>
  <p>局部刷新AJAX-计算bmi</p>
  <div>
    <!--没有使用form表单-->
    姓名:<input type="text" id="name"><br>
    体重(公斤):<input type="text" id="weight"><br>
    身高(米):<input type="text" id="height"><br>
    <input type="button" value="计算BMI" onclick="doAjax()">
    <br><br>
    <div id="myData">等待加载数据.....</div>
  </div>
</body>
</html>
```

```
public class BMIAJAXServlet extends HttpServlet {// /bmiAjax
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.接收请求参数
        String name = (String) request.getParameter("name");
        String weight = (String) request.getParameter("weight");
        String height = (String) request.getParameter("height");
        //2.计算BMI: bmi = weight / (height * height);
        float w = Float.valueOf(weight);
        float h = Float.valueOf(height);
        float bmi = w / (h * h);
        //3.判断BMI的范围
        String msg = "";
        if(bmi <= 18.5){
            msg = "您比较瘦";
        } else if(bmi <= 23.9){
            msg = "您的BMI是正常的";
        } else if(bmi <= 27){
            msg = "您的身体过重";
        } else if (bmi <= 32){
            msg = "您的身体肥胖";
        } else {
            msg = "您的身体过于肥胖";
        }
        System.out.println("msg = " + msg);
        msg = "您好" + name + "先生/女士，您的BMI是: " + bmi + ", " + msg;
        System.out.println(msg);
        //4.响应ajax需要的数据，使用HttpServletResponse输出数据
        response.setContentType("text/html;charset=utf-8");
        PrintWriter pw = response.getWriter();
        pw.println(msg);
        pw.flush();
        pw.close();
    }
}
```

四、AJAX实例二

- 需求：用户在文本框架输入省份的编号 id，在其他文本框显示省份名称
- 项目环境准备：
  - 数据库：springdb
  - 数据表：

```sql
#省份信息表：
CREATE TABLE `province` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(255) DEFAULT NULL COMMENT '省份名称',
 `jiancheng` varchar(255) DEFAULT NULL COMMENT '简称',
 `shenghui` varchar(255) DEFAULT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;

#城市信息表：
CREATE TABLE `city` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(255) DEFAULT NULL,
 `provinceid` int(11) DEFAULT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=utf8;
```

province @springdb (XK_MySQL) - 表

文件 编辑 查看 窗口 帮助

导入向导 导出向导 筛选向导 网格查看 表单查看 备注 十六进制 图像 升序排序 降序排序 移除排序

id	name	jiancheng	shenghui
1	河北	冀	石家庄
2	山西	晋	太原市
3	内蒙古	蒙	呼和浩特市
4	辽宁	辽	沈阳
5	江苏	苏	南京
6	浙江	浙	杭州
7	安徽	皖	合肥
8	福建	闽	福州
9	江西	赣	南昌

1

`* FROM `province` LIMIT 0, 1000 第 1 条记录 (共 9 条) 于 1 页

```html
<!--index.jsp-->
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>ajax根据省份Id获取名称</title>
    <script type="text/javascript">
        function search(){
            //发起AJAX请求，传递参数给服务器，服务器返回数据
            //1.创建异步对象
            var xmlHttp = new XMLHttpRequest();
            //2.绑定事件
            xmlHttp.onreadystatechange = function (){
                if(xmlHttp.readyState == 4 && xmlHttp.status == 200){
                    document.getElementById("proname").value = xmlHttp.responseText;
                }
            }
            //3.初始异步对象
            //获取proid文本框的值
            var proid = document.getElementById("proid").value;
            xmlHttp.open("get", "queryProvice?proid="+proid, true);
            //4.发送请求
            xmlHttp.send();
        }
    </script>
</head>
<body>
    <p align="center">ajax根据省份Id获取名称</p>
    <table align="center">
        <tr>
            <td>省份编号：</td>
            <td>
                <input type="text" id="proid">
                <input type="button" value="搜索" onclick="search()">
```

```
            </td>
        </tr>
        <tr>
            <td>省份名称：</td>
            <td><input type="text" id="proname"></td>
        </tr>
    </table>
</body>
</html>
```

```
public class QueryProvinceServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println("响应了AJAX请求");
        String name = "";
        //处理get请求
        String proidStr = request.getParameter("proid");
        System.out.println("proid==="+proidStr);
        if(proidStr != null && !"".equals(proidStr.trim())){
            //访问Dao，查询数据库
            name = ProvinceDao.queryProvinceNameById(Integer.valueOf(proidStr));
        }
        //使用HttpServletResponse输出数据
        //响应ajax需要的数据，使用HttpServletResponse输出数据
        response.setContentType("text/html;charset=utf-8");
        PrintWriter pw = response.getWriter();
        pw.println(name);
        pw.flush();
        pw.close();
    }
}
```

```
//使用JDBC访问数据库
public class ProvinceDao {
    //根据Id获取名称
    public static String queryProvinceNameById(Integer provinceId){
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String name = "";
        try {
            conn = JdbcUtil.getConnection();
            String sql = "select name from province where id=?";
            ps = conn.prepareStatement(sql);
            ps.setInt(1, provinceId);
            rs = ps.executeQuery();
            if(rs.next()){
                name = rs.getString("name");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, rs);
        }
        return name;
    }
}
```

五、JSON

- ajax发起请求---- Servlet（返回的一个json格式的字符串 { name:"河北", jiancheng:"冀","shenghui":"石家庄"}）
- JSON分类：
  - json对象，JSONObject，这种对象的格式：“名称:值”，也可以看做是 "key:value" 格式。
  - json数组，JSONArray，基本格式 [{ name:"河北", jiancheng:"冀","shenghui":"石家庄"} , { name:"山西", jiancheng:"晋","shenghui":"太原"}]
- 为什么使用JSON（优点）：
  - json格式好理解
  - json格式数据在多种语言中，比较容易处理；使用java，javascript读写json格式的数据比较容易。
  - json格式数占用的空间小，在网络中传输快，用户的体验好。
- 处理json的工具库：
  - gson（google）
  - fastjson（阿里）：速度快，不是符合JSON处理规范的
  - jackson：性能好，规范好
  - json-lib：性能差，依赖多
- 在js中，可以把json格式的字符串，转为json对象， json中的key，就是json对象的属性名。

1. jackson使用

```
public class TestJson {
    public static void main(String[] args) throws JsonProcessingException {
        //使用 jackson 把 java对象转为json格式
        Province province = new Province();
        province.setId(1);
        province.setName("江苏");
        province.setJiancheng("苏");
        province.setShenghui("南京");

        //使用jackson 把 province对象 转为 json
        ObjectMapper om = new ObjectMapper();
        //writeValueAsString:把参数的java对象转为json格式的字符串
        String json = om.writeValueAsString(province);
        System.out.println("json===" + json);
        //json==={"id":1,"name":"江苏","jiancheng":"苏","shenghui":"南京"}
    }
}
```

2. ajax请求使用json格式的数据

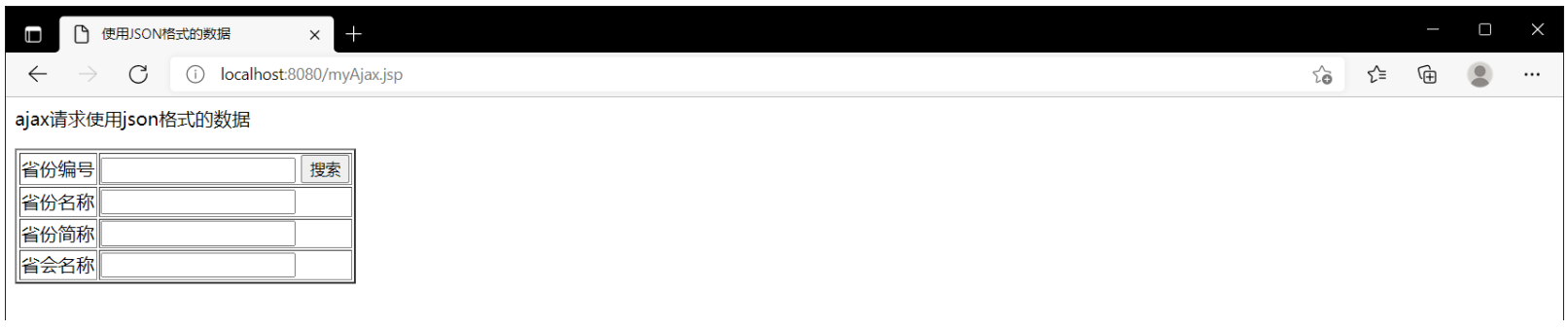
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>使用JSON格式的数据</title>
    <script type="text/javascript">
        function doSearch(){
            //1.创建异步对象
            var xmlHttp = new XMLHttpRequest();
            //2.绑定事件
            xmlHttp.onreadystatechange = function (){
                if(xmlHttp.readyState == 4 && xmlHttp.status == 200){
                    var data = xmlHttp.responseText;
                    //eval是执行括号中的代码，把json字符串转为json对象
                    var jsonObj = eval("(" + data + ")")
                    //更新Dom对象
                    document.getElementById("proname").value = jsonObj.name;
                    document.getElementById("projc").value = jsonObj.jiancheng;
                    document.getElementById("prosh").value = jsonObj.shenghui;
                }
            }
            //3.初始异步对象
            //获取proid文本框的值
            var proid = document.getElementById("proid").value;
            xmlHttp.open("get", "queryJson?proid="+proid, true);
            //4.发送请求
            xmlHttp.send();
        }
    </script>
</head>
<body>
    <p>ajax请求使用json格式的数据</p>
    <table border="2">
        <tr>
            <td>省份编号</td>
            <td>
                <input type="text" id="proid">
                <input type="button" value="搜索" onclick="doSearch()">
            </td>
        </tr>
```

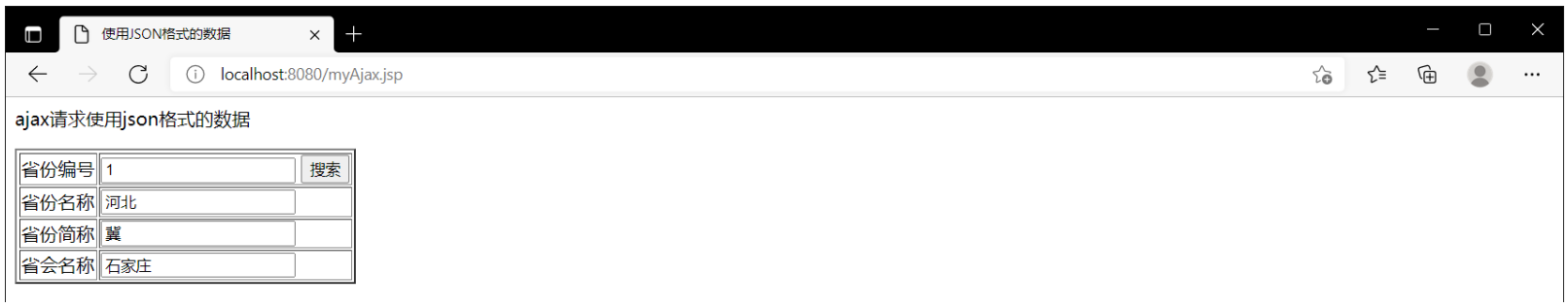
```
        <tr>
            <td>省份名称</td>
            <td><input type="text" id="proname"></td>
        </tr>
        <tr>
            <td>省份简称</td>
            <td><input type="text" id="projc"></td>
        </tr>
        <tr>
            <td>省会名称</td>
            <td><input type="text" id="prosh"></td>
        </tr>
    </table>
</body>
</html>
```

```
public class QueryJsonServlet extends HttpServlet { // queryJson
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //默认值,{}:表示json格式的数据
        String json = "{}";
        //获取请求参数，省会的Id
        String strProid = request.getParameter("proid");
        //判断proid有值时，调用dao查询参数
        if(strProid != null && strProid.trim().length() > 0){
            Province province = ProvinceDao.queryProvinceById(Integer.valueOf(strProid));
            //需要使用jackson 把 province对象 转为 json
            ObjectMapper om = new ObjectMapper();
            json = om.writeValueAsString(province);
        }
        //把获取的数据，通过网络传给ajax中的异步对象，响应结果数据
        //指定服务器端(servlet)返回给浏览器的是JSON格式的数据
        response.setContentType("application/json;charset=utf-8");
        PrintWriter pw = response.getWriter();
        pw.println(json);//输出数据，会交付给ajax中responseText属性
        pw.flush();
        pw.close();
    }
}

```

```
//使用JDBC访问数据库
public class ProvinceDao {
    //根据Id获取完整的Province对象
    public static Province queryProvinceById(Integer provinceId){
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Province province = null;
        try {
            conn = JdbcUtil.getConnection();
            String sql = "select * from province where id=?";
            ps = conn.prepareStatement(sql);
            ps.setInt(1, provinceId);
            rs = ps.executeQuery();
            while (rs.next()){
                province = new Province();
                province.setId(rs.getInt("id"));
                province.setName(rs.getString("name"));
                province.setJiancheng(rs.getString("jiancheng"));
                province.setShenghui(rs.getString("shenghui"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            JdbcUtil.close(conn, ps, rs);
        }
        return province;
    }
}
```






```
<!--通过回调函数处理返回的数据-->
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>使用JSON格式的数据</title>
    <script type="text/javascript">
        function doSearch(){
            //1.创建异步对象
            var xmlHttp = new XMLHttpRequest();
            //2.绑定事件
            xmlHttp.onreadystatechange = function (){
                if(xmlHttp.readyState == 4 && xmlHttp.status == 200){
                    var data = xmlHttp.responseText;
                    //eval是执行括号中的代码，把json字符串转为json对象
                    var jsonObj = eval("(" + data + ")")
                    callback(jsonObj);
                }
            }
            //3.初始异步对象
            //获取proid文本框的值
            var proid = document.getElementById("proid").value;
            xmlHttp.open("get", "queryJson?proid="+proid, true);
            //4.发送请求
            xmlHttp.send();
        }

        //定义函数，处理服务器端返回的数据
        function callback(json){
            //更新Dom对象
            document.getElementById("proname").value = json.name;
            document.getElementById("projc").value = json.jiancheng;
            document.getElementById("prosh").value = json.shenghui;
        }
    </script>
</head>
<body>
    <p>ajax请求使用json格式的数据</p>
    <table border="2">
        <tr>
            <td>省份编号</td>
            <td>
                <input type="text" id="proid">
                <input type="button" value="搜索" onclick="doSearch()">
            </td>
        </tr>
        <tr>
            <td>省份名称</td>
            <td><input type="text" id="proname"></td>
        </tr>
        <tr>
            <td>省份简称</td>
            <td><input type="text" id="projc"></td>
        </tr>
        <tr>
            <td>省会名称</td>
            <td><input type="text" id="prosh"></td>
        </tr>
    </table>
</body>
</html>
```

六、同步和异步

- open(method, url, async)：初始化异步请求对象
- method：请求的类型；GET 或 POST
- url：服务器端的访问地址（例如servlet地址）
- async：true（异步）或 false（同步）
- true：异步处理请求；使用异步对象发起请求后，不用等待数据处理完毕，就可以执行其他的操作。
- false：同步处理请求；异步对象必须处理请求，从服务端获取数据后，才能执行send之后的代码；任意时刻只能执行一个异步请求。