Web代码复习

后端 Servlet

- 处理表单提交请求，如登录验证、信息发布、在线注册等；
- 为前端提供数据接口，如获取新闻列表、获取学术成果等；

过程

1. 接受参数
2. 判断参数合理性
3. 数据库参数, 连接
4. 循环取内容
    - json数组+单条JSON
    - 成功时, 将JSON数组转为字符串
    - 成功时, 将JSON对象转为字符串
5. 判断数据库取出内容合理性
    - 失败就设置响应json的success为false, msg为失败报文
    - 将JSON对象转为字符串

例子1: 数据库刷新取前五条

```
public class AcademicInfoServlet extends HttpServlet {

    // 数据库连接相关参数
    private final String jdbcDriver = "com.mysql.jdbc.Driver";
    private final String jdbcUrl = "jdbc:mysql://localhost:3306/test";
    private final String jdbcUsername = "root";
    private final String jdbcPassword = "password";

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            // 连接数据库
            Class.forName(jdbcDriver);
            conn = DriverManager.getConnection(jdbcUrl, jdbcUsername,
jdbcPassword);

            // 构造SQL语句
            String sql = "SELECT * FROM Tab_AcademicInfo ORDER BY infoDate DESC
LIMIT 5";
            stmt = conn.prepareStatement(sql);

            // 执行查询
```

```
            rs = stmt.executeQuery();

            // 处理查询结果，生成 academicInfoList 存放所有查询到的数据集
            List<JSONObject> academicInfoList = new ArrayList<JSONObject>();

            // rs是ResultSet对象，存储查询结果集. 使用rs.getxxx(colum_name)可以获取对应
列的值
            // rs.next() 方法返回一个布尔值，表示结果集中是否还有下一行数据。
            while (rs.next()) {
                JSONObject academicInfo = new JSONObject();
                academicInfo.put("id", rs.getInt("ID"));
                academicInfo.put("infoTitle", rs.getString("infoTitle"));
                academicInfo.put("infoDate", rs.getString("infoDate"));
                academicInfo.put("infoURL", rs.getString("infoURL"));
                academicInfoList.add(academicInfo);
            }

            // 将查询结果转换为JSON字符串并返回给前端
            // 使用 JSON.toJSONString 将List转为JSON字符串
            String academicInfoJsonResponse =
JSON.toJSONString(academicInfoList);

            // 响应头设置
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");
            // 把响应传给ajax
            response.getWriter().write(academicInfoJsonResponse);

        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

例子2: 登录处理/数据库/session

`@WebServlet("/login")` 注解中的 `/login` 表示将该 `Servlet` 映射到根路径下的 `login` 路径，即在浏览器中访问 `http://localhost:8080/login` 将会请求该 `Servlet`。如果需要将该 `Servlet` 映射到多个路径，可以在注解中使用花括号 `{}` 来指定多个路径，如 `@WebServlet({"/login", "/signin"})` 将该 `Servlet` 映射到根路径下的 `login` 和 `signin` 路径。

```
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    // 数据库连接相关参数
    private final String jdbcDriver = "com.mysql.jdbc.Driver";
    private final String jdbcUrl = "jdbc:mysql://localhost:3306/test";
    private final String jdbcUsername = "root";
    private final String jdbcPassword = "password";

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        JSONObject json = new JSONObject(); // 响应消息对象
```

```
        try (Connection conn = DriverManager.getConnection(jdbcUrl, jdbcUsername,
jdbcPassword);
             PreparedStatement stmt = conn.prepareStatement("SELECT * FROM user
WHERE username=? AND password=?")) {
            // 获取客户端提交的用户名和密码
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            stmt.setString(1, username);
            stmt.setString(2, password);

            // 执行查询
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    // 如果查询到了匹配的用户，设置Session信息，跳转到主页面
                    HttpSession session = request.getSession();
                    session.setAttribute("username", username);
                    session.setAttribute("password", password);
                    // 构造响应报文
                    json.put("success", true);
                    response.getWriter().write(json.toString());
                    response.sendRedirect("main.jsp");
                } else {
                    // 如果没有查询到匹配的用户，返回错误信息给前端
                    json.put("success", false);
                    json.put("message", "Invalid username or password.");
                    response.getWriter().write(json.toString());
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
"Server Error");
        }
    }
}
```

搜索响应

```
public class SearchServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    // 数据库连接相关参数
    private final String jdbcDriver = "com.mysql.jdbc.Driver";
    private final String jdbcUrl = "jdbc:mysql://localhost:3306/test";
    private final String jdbcUsername = "root";
    private final String jdbcPassword = "password";

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String keyword = request.getParameter("keyword");
        if (keyword == null || keyword.trim().isEmpty()) {
            // 如果关键词为空，返回错误信息
            sendErrorResponse(response, "Invalid keyword");
```

```
            return;
        }

        // 连接数据库并查询
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            Class.forName(jdbcDriver);
            conn = DriverManager.getConnection(jdbcUrl, jdbcUsername,
jdbcPassword);

            String sql = "SELECT ID, Title, Content, Ptime FROM Tab_news WHERE
Title LIKE ? OR Content LIKE ?";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, "%" + keyword + "%");
            stmt.setString(2, "%" + keyword + "%");
            rs = stmt.executeQuery();

            if (rs.next()) {
                // 如果查询到结果，构造JSON响应
                JSONArray resultArray = new JSONArray();
                do {
                    JSONObject itemObject = new JSONObject();
                    itemObject.put("id", rs.getInt("ID"));
                    itemObject.put("title", rs.getString("Title"));
                    itemObject.put("content", rs.getString("Content"));
                    itemObject.put("ptime", rs.getString("Ptime"));
                    resultArray.add(itemObject);
                } while (rs.next());
                JSONObject responseJson = new JSONObject();
                responseJson.put("success", true);
                responseJson.put("data", resultArray);
                sendJsonResponse(response, responseJson);
            } else {
                // 如果查询结果为空，返回错误信息
                sendErrorResponse(response, "No results found");
            }
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
            sendErrorResponse(response, "Server error");
        } finally {
            try { if (rs != null) rs.close(); } catch (SQLException e) {
e.printStackTrace(); }
            try { if (stmt != null) stmt.close(); } catch (SQLException e) {
e.printStackTrace(); }
            try { if (conn != null) conn.close(); } catch (SQLException e) {
e.printStackTrace(); }
        }
    }

    // 发送JSON响应
    private void sendJsonResponse(HttpServletResponse response, JSONObject json)
throws IOException {
        response.setContentType("application/json");
```

```
        response.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        out.print(json.toJSONString());
        out.flush();
    }

    // 发送错误信息
    private void sendErrorResponse(HttpServletResponse response, String message)
throws IOException {
        JSONObject errorJson = new JSONObject();
        errorJson.put("success", false);
        errorJson.put("message", message);
        sendJsonResponse(response, errorJson);
    }
}
```

前后端交互 Ajax

Ajax提交是通过js来提交请求，请求与响应均由js引擎来处理，页面不会刷新，用户感觉不到实际上浏览器发出了请求。比如说我们希望网页总是显示最新的新闻，而又不想老是去点刷新按钮，我们就可以用Ajax机制来实现。网上的客服软件也是ajax请求的一个比较好的案例。传统的请求页面将实现刷新，因此局限性很大。

1.为什么用AJAX?
使用AJAX，用户对Web的体验会更“敏捷”：数据提交页面不会闪屏；页面局部更新速度快；网络带宽占用低。

2.AJAX开发相较传统模式的简单之处：
传统模式下，表单提交则整个页面重绘，为了维持页面用户对表单的状态改变，要多些不少代码。要在控制器和模板之间传递更多参数以保持页面状态。而AJAX不然，因为页面只是局部更新， 不关心也不会影响页面其他部分的内容。

3.AJAX开发相较传统模式的难度：
需要了解、精通JavaScript，而JavaScript存在调试麻烦、浏览器兼容性等很多障碍。

有如下几种区别：

1. A在提交时，是在后台新建一个请求；F却是放弃本页面，而后再请求；
2. A必须要使用JS来实现，不启用JS的浏览器，无法完成该操作；F却是浏览器的本能，无论是否开启JS，都可以提交表单；
3. A在提交、请求、接收时，整个过程都需要使用程序来对其数据进行处理；F提交时，却是根据你的表单结构自动完成，不需要代码干预。

4. form是整体刷新，请求成功会进行页面的跳转，而ajax是局部刷新，在请求时不会跳转页面，页面加载效率也更高；
5. form请求用户体验不好，涉及到了页面的来回跳转；而ajax用户体验好，因为它请求时页面是不变的；

6. form更适合传统的前后端不分离项目；而ajax更适合前后端分离的项目；
7. 支持的请求方式不一样；form 仅支持 GET/POST 两种请求

- 动态加载页面内容，如异步加载新闻列表、学术论文、研究项目等；
- 在不刷新页面的情况下提交表单数据，如搜索框的实时搜索、登录验证、留言评论等；

例子1: 地图显示

```
let map = document.getElementById("map")
map.addEventListener("click", function (e) {
    // Step1 创建请求对象
    let rqst;
    if (window.XMLHttpRequest) rqst = new XMLHttpRequest();
    else rqst = new ActiveXObject("Microsoft.XMLHTTP");
    // Step2 设置请求对象的请求行/请求头
    const myKey = "91661a899863b23d58673be0e52e43a5"
    let url = "https://restapi.amap.com/v3/staticmap?
location=112.941658,28.149713&zoom=13&key=91661a899863b23d58673be0e52e43a5"
    rqst.responseType = 'blob'
    rqst.open('GET', url, true)
    // Step3 设置回调函数
    rqst.onreadystatechange = () => {
        if (rqst.readyState === 4 && rqst.status === 200) {
            //let blob = new Blob([rqst.response], { type: "image/png" });
            let img = document.createElement("img");
            img.src = URL.createObjectURL(rqst.response);
            document.body.appendChild(img);
        }
    }
    rqst.send(null)
})
```

例子2: 天气显示

```
let weather = document.getElementById("weather")

weather.addEventListener("click", function (e) {
    // Step1 创建请求对象
    let rqst;
    if (window.XMLHttpRequest) rqst = new XMLHttpRequest();
    else rqst = new ActiveXObject("Microsoft.XMLHTTP");
    // Step2 设置请求对象的请求行/请求头
    const myKey = "91661a899863b23d58673be0e52e43a5"
    let myCity = "430104"
    const api = "https://restapi.amap.com/v3/weather/weatherInfo?"
    let url = api + `?key=${myKey}` + `&city=${myCity}`
    rqst.open('GET', url, true)
    // Step3 设置回调函数
    rqst.onreadystatechange = () => {
        if (rqst.readyState === 4 && rqst.status === 200) {
```

```
            const rsps = JSON.parse(rqst.response);
            const weatherData = rsps.lives[0]
            let ul = document.createElement("ul")
            for (let item in weatherData) {
                let li = document.createElement('li')
                li.innerText = item + " : " + weatherData[item]
                ul.appendChild(li)
            }
            document.body.appendChild(ul)
        }
    }
    // Step4 发送请求（请求体作为参数, Get方法则为null）
    rqst.send(null)
})
```

例子3: 检查电话号码

```
function checktelcaptcha(){
    var userid = document.querySelector("#userid").value;
    var usertel = document.querySelector("#usertel").value;
    var captcha = document.querySelector("#captcha").value;
    // 送异步请求
    // 1.创建核心对象
    var xmlhttp;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.open("GET","ForgetServlet?usertel="+usertel+"&&captcha="+captcha +
"&&userid=" + userid,true);

    //3.发送请求
    xmlhttp.send();
// 4.回调函数
    xmlhttp.onreadystatechange=function()
    {
        //判断readyState就绪状态是否为4，判断status响应状态码是否为200
        if (xmlhttp.readyState==4 )
        {
            if(xmlhttp.status==200){
                //获取服务器的响应结果
                var responseText = xmlhttp.responseText;
                document.querySelector("#showWarnTip").innerText = responseText;
            }
        }
    }

}
```

例子4: 登录请求处理

```
// 获取登录表单元素
const loginForm = document.getElementById('login-form');

// 监听表单提交事件
loginForm.addEventListener('submit', function(event) {
  event.preventDefault(); // 阻止表单默认提交行为

  // 获取表单中的用户名和密码
  const username = loginForm.elements.username.value;
  const password = loginForm.elements.password.value;

  // 发送 AJAX 请求
  const xhr = new XMLHttpRequest();
  xhr.open('POST', '/login', true);
  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
        // 200代表请求已成功处理，并返回了响应数据。
        if (xhr.status === 200) {
            // 验证成功
            const response = JSON.parse(xhr.responseText);
            if (response.success) {
              // 登录成功，跳转到主页

            } else {
              // 登录失败，显示错误提示
              const errorMessage = document.getElementById('error-message');
              errorMessage.textContent = response.message;
            }
      } else {
        // 显示网络错误提示
        const errorMessage = document.getElementById('error-message');
        errorMessage.textContent = '网络错误，请重试。';
      }
  };
  xhr.send(`username=${encodeURIComponent(username)}&password=${encodeURICompone
nt(password)}`);
});
```

搜索框

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>搜索框</title>
  <style>
    body {
        background-color: rgb(117, 117, 235);
        padding: 0;
        margin: 0;
    }
    .container {
        height: 50px;
```

```
        margin-top: 20px;
    }
    .show {
        display: none;
        float: right;
        margin-right: 1.5rem;
        height: 50px; /*父元素必须给出高度*/
    }
    .search {
        border: 1px solid rgb(220, 219, 219);
        border-radius: 1rem;
        width: 400px;
        display: flex;
        align-items: center;
        padding-left: 1rem;
    }
    .show input {
        width: 330px;
        height: 70%;
        background-color: rgb(117, 117, 235);
        color: white;
        border: none;
        margin-right: 0.5rem;
    }
    input:focus {
        outline: none;
    }
    .show .close {
        height: 50px;
        display: flex;
        align-items: center;
        margin-left: 0.5rem;
    }
    .show .close img {
        width: 30px;
    }
    .init {
        float: right;
        margin-right: 27px;
    }
    .init img {
        width: 25px;
        margin-top: 50%;
    }

.highlight {
    color: red;
    font-weight: bold;
}
  </style>
</head>
<body>
  <div class="container">

    <div class="show" id="show">
```

```
        <div class="search" >
            <!-- 真正的输入框 -->
            <input type="text" id="keyword">
            <!-- 点击搜索的图标 -->
            <img src="https://www.csu.edu.cn/images/i-so.png" id="go">
        </div>
        <!-- 关闭页面的图标 -->
        <div class="close" id="close">
            <img src="https://www.csu.edu.cn/images/i-close.png">
        </div>
    </div>
    <!-- 打开搜索框的图标 -->
    <div class="init" id="init">
        <img src="https://www.csu.edu.cn/images/so-btn.png">
    </div>

  </div>
  <script>
    // 获得白色搜索图标
    let init_search = document.getElementById("init");
    // 获得需要展开的搜索框
    let show_search = document.getElementById("show");
    // 添加点击事件
    init_search.addEventListener('click', () => {
        init_search.style.display = 'none';
        show_search.style.display = 'flex';
    })
    // 或许关闭图标
    let close_search = document.getElementById("close");
    // 添加关闭事件
    close_search.addEventListener('click', () => {
        init_search.style.display = 'block';
        show_search.style.display = 'none';
    })

    /* 下面是Ajax部分，用于提交用户的搜索参数到服务器 */
    // 获得黄色的搜索图标
    let search = document.getElementById("go");
    search.addEventListener('click', () => {
        // 获取输入关键词
        const keyword = document.getElementById("keyword").value;

        // 发起Ajax请求
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "searchServlet");
        xhr.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
        xhr.onreadystatechange = function() {
            if (xhr.readyState === XMLHttpRequest.DONE) {
                if (xhr.status === 200) {
                    // 请求成功，解析响应数据
                    var responseData = JSON.parse(xhr.responseText);

                    // 如果搜索成功，将结果渲染到页面上
                    if (responseData.success) {
```

```
                        var resultArray = responseData.results;
                        var html = "";
                        for (var i = 0; i < resultArray.length; i++) {
                            var result = resultArray[i];
                            var title = result.title;
                            var content = result.content;
                            var ptime = result.ptime;

                            // 针对标题和消息标红关键字
                            title = highlightKeyword(title, keyword);
                            content = highlightKeyword(content, keyword);
                            // 构造html结构，重新绘制
                            html += "<div class='result-item'>" +
                                "<a href="url">"
                            "<h2>" + title + "</h2>" +
                                "<p>" + content + "</p>" +
                                "<p>" + ptime + "</p>" +
                                "</a>" +
                                "</div>";
                        }
                        searchResults.innerHTML = html;
                    } else {
                        // 如果搜索失败，显示错误消息
                        showError(responseData.message);
                    }
                } else {
                    // 请求失败，显示错误消息
                    showError("网络错误");
                }
            }
        };
        // 发送请求
        xhr.send("keyword=" + encodeURIComponent(keyword));
    });

    // 显示错误消息的函数
    function showError(message) {
        alert(message);
    }

    // 标红关键字的函数
    function highlightKeyword(str, keyword) {
        var re = new RegExp(keyword, "gi");
        return str.replace(re, "<span class='highlight'>$&</span>");
    }
  </script>
</body>
</html>
```

对于 GET 请求，可以使用以下代码：

```
javascriptCopy codeconst xhr = new XMLHttpRequest();
const username = "myusername";
const password = "mypassword";
const url = "/login?username=" + username + "&password=" + password;
xhr.open('GET', url, true);
xhr.send();
```

对于 POST 请求，可以使用以下代码：

```
javascriptCopy codeconst xhr = new XMLHttpRequest();
const username = "myusername";
const password = "mypassword";
const url = "/login";
xhr.open('POST', url, true);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
const data = "username=" + encodeURIComponent(username) + "&password=" +
encodeURIComponent(password);
xhr.send(data);
```

可以看到，GET 请求将参数添加到 URL 中，而 POST 请求则将参数添加到请求主体中。此外，POST 请求需要设置 `Content-Type` 头部，指定发送的数据格式。在上面的代码中，我们使用了 `application/x-www-form-urlencoded` 格式，即表单格式。

> 在HTTP协议中，GET和POST是HTTP请求方法。它们的主要区别在于：
>
> 1. 数据传输方式：GET方法使用URL传递数据，而POST方法使用请求体传递数据。
> 2. 安全性：GET方法请求的数据会显示在URL上，而POST方法请求的数据则不会显示在URL上，因此POST方法相对于GET方法更加安全。
> 3. 数据大小限制：由于GET方法将数据以URL参数的形式传递，因此数据大小存在限制，通常不应该超过2KB；而POST方法则可以传递更大的数据量，通常不会受到数据大小的限制。
> 4. 缓存：GET方法默认会被浏览器缓存，而POST方法不会被缓存。
>
> 在实际应用中，通常使用GET方法来请求数据，使用POST方法来提交数据。如果请求的数据只是简单的查询操作，可以使用GET方法，如果请求的数据是需要修改服务器上的数据，则应该使用POST方法。

注意, 对于post来说会调用encodeURIComponent:

`encodeURIComponent()` 是一个 JavaScript 函数，用于将字符串作为 URI 组件进行编码。它会将字符串中的某些字符（例如不安全字符和保留字符）替换为它们的转义序列，以便能够安全地使用它们作为 URL 的一部分。

举个例子，如果你想将字符串 `hello world` 编码为 URI 组件，则可以使用如下代码：

```
javascriptCopy code
var encodedString = encodeURIComponent('hello world');
```

编码后的结果为 `hello%20world`。在实际应用中，常常需要将字符串编码后作为 URL 的参数传递给服务器端，这时候就需要用到 `encodeURIComponent()` 函数。

JSON处理

（一）Servlet处理JSON-List

Gson 和 `JSON.toJSONString()` 都可以将 List 转换为 JSON 字符串。

`JSON.toJSONString()` 是阿里巴巴的 FastJson 库提供的一个静态方法，而 Gson 是 Google 提供的一个 JSON 库。它们都能够实现 JSON 和 Java 对象的互相转换，同时也支持将 Java 集合类型（如 List）转换为 JSON 数组。

以下是将一个 List 转换为 JSON 的示例：

使用 Gson:

```
List<MyObject> myList = new ArrayList<>();
// 将对象添加到 myList
Gson gson = new Gson();
String json = gson.toJson(myList);
```

使用 FastJson:

```
javascriptCopy codeList<MyObject> myList = new ArrayList<>();
// 将对象添加到 myList
String json = JSON.toJSONString(myList);
```