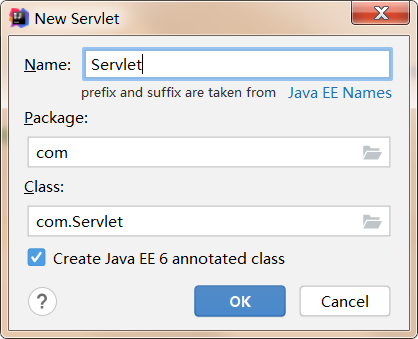
实习二 Servlet技术

1. 实习要求与目的
2. 掌握 Servlet 的工作机制及编写方法
3. 掌握 Filter工作机制及编写方法
4. 了解 Servlet 生命周期事件
5. 掌握数据库访问方法
6. 实习内容
7. Servlet
8. 第一个servlet

创建一个Java Web的maven工程。在工程结构窗格的“java”目录上点击右键，选择“new”->“Package”创建“com”包；然后在“com”上点击右键选择“new”->“new Create Servlet”打开如下对话框：



Name：输入Servlet类的名称；

Package：新建类所在包；

Class：新建类的完整类名；

Create Jave EE 6 annotated class：是否使用注解对Servlet进行配置；

点击“OK”按钮，完成Servlet的创建。修改新建Servlet的代码内容如下：

@WebServlet(name = "firstServlet",urlPatterns = "/serv")

public class FirstServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request,response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.getWriter().println("欢迎来到天涯海角@@@");

}

}

启动服务器并访问servlet，查看结果是否正常；若不正常如何修改。

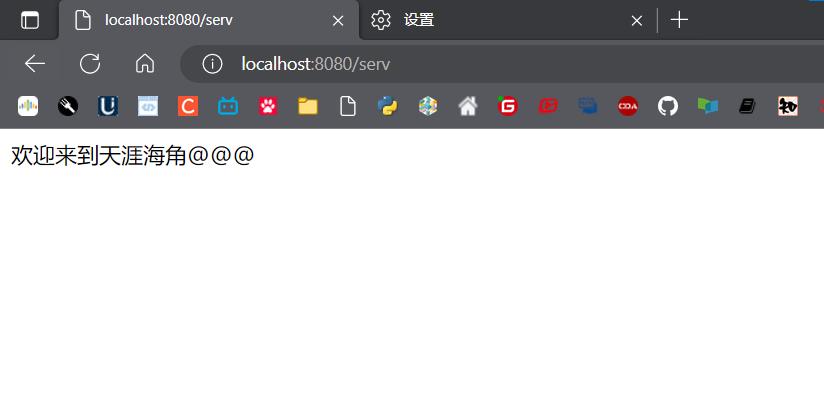
乱码了，需要修改编码类型

response.setContentType("text/html");

response.setCharacterEncoding("utf-8");

request.setCharacterEncoding("utf-8");

效果图如下：



1. 表单处理

改造实习一中的2.1（表单处理）。新建一个servlet接收表单提交的内容，并对提交内容进行处理，然后交由result.jsp显示结果。

Post.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<form action="/get" method="post">

姓名：

<input type="text" name="name">

<br>

性别：

<input type="radio" name="sex" value="male">男

<input type="radio" name="sex" value="female">女

<br>

年龄：

<input type="text" name="age">

<br>

爱好：

<input type="checkbox" name="hobby" value="sport">运动

<input type="checkbox" name="hobby" value="travel">旅游

<input type="checkbox" name="hobby" value="reading">阅读

<input type="checkbox" name="hobby" value="PE">体育

<br>

<input type="submit" value="提交">

</form>

</body>

</html>

Servlet文件Get.java

package com.servlet;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.util.Arrays;

public class get extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

String name = req.getParameter("name");

int age = Integer.parseInt(req.getParameter("age"));

String sex = req.getParameter("sex");

String hobbies = Arrays.toString(req.getParameterValues("hobby"));

HttpSession session = req.getSession();

if(age < 16){

session.setAttribute("error\_my", "年纪小于16");

} else {

session.setAttribute("name", name);

session.setAttribute("age", age);

session.setAttribute("sex", sex);

session.setAttribute("hobbies", hobbies);

}

resp.sendRedirect("result.jsp");

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req, resp);

}

}

在web.xml中注册get.java

<servlet>

<servlet-name>get</servlet-name>

<servlet-class>com.servlet.get</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>get</servlet-name>

<url-pattern>/get</url-pattern>

</servlet-mapping>

Result.jsp

<%--

Created by IntelliJ IDEA.

User: 18109

Date: 2022/9/27

Time: 19:35

To change this template use File | Settings | File Templates.

--%>

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

<%@ page import="java.lang.reflect.Array" %>

<%@ page import="java.util.ArrayList" %>

<%@ page import="java.util.Arrays" %>

<html>

<head>

<title>Title</title>

</head>

<body>

<%

if(session.getAttribute("error\_my") == null){

String name = session.getAttribute("name").toString();

String age = session.getAttribute("age").toString();

String sex = session.getAttribute("sex").toString();

String hobbies = session.getAttribute("hobbies").toString();

%>

<h3>你的姓名为：<%=name%></h3>

<h3>你的年龄为：<%=age%></h3>

<h3>你的性别为：<%=sex%></h3>

<h3>你的爱好为：<%=hobbies%></h3>

<%

} else{

%>

<h3>ERROR:你的年龄小于16</h3>

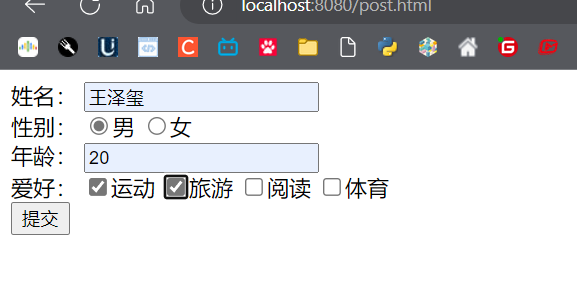
<%

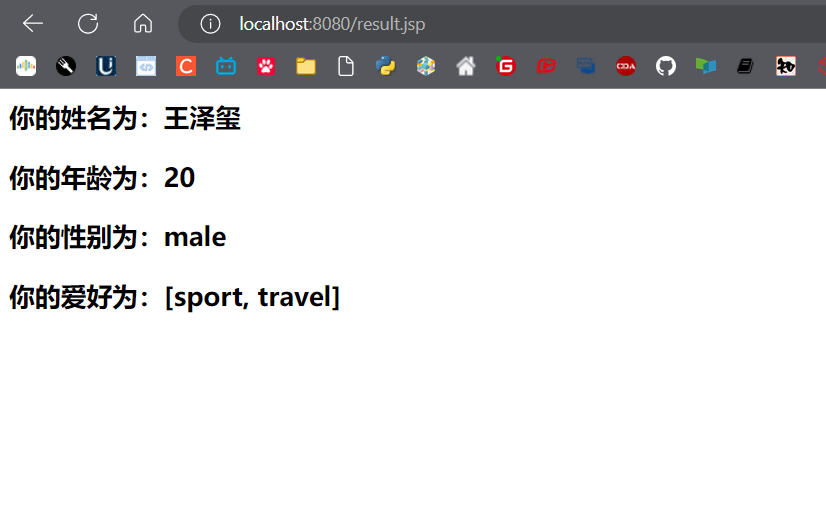
}

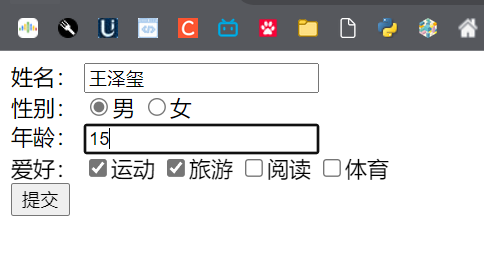
%>

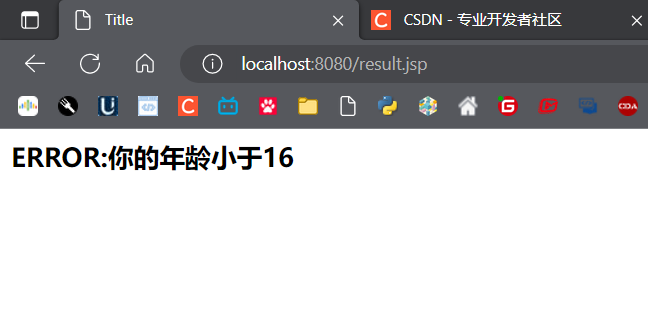
</body>

</html>









1. 过滤器的使用
2. 使用过滤器重写实习一中的2.3（登录验证）。
3. 使用Cookie为（1）中的登录验证实现自动登录功能。（在登录页面添加“自动登录”的复选框，若选择下次访问该应用时使用cookie存储的用户名和口令完成字典登录）。

Login.jsp

<%--

Created by IntelliJ IDEA.

User: 18109

Date: 2022/9/27

Time: 20:31

To change this template use File | Settings | File Templates.

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>登录界面</title>

<%

String account = "";

String password = "";

//获取是否有错

String error = (String) session.getAttribute("error\_my");

if (session.getAttribute("rem") != null) {

//在cookie中查找账号密码

Cookie[] cookies = request.getCookies();

if (cookies != null) {

for (Cookie cookie : cookies) {

if ("account".equals(cookie.getName())) {

account = (String) cookie.getValue();

}

if ("password".equals(cookie.getName())) {

password = (String) cookie.getValue();

}

}

}

%>

<script>

var ju = true;

</script>

<%

}

if(error != null){

%>

<script type="text/javascript">

var ju = false;

//错误弹窗

alert("<%=error%>错误");

</script>

<%

}

%>

</head>

<body>

<form action="admin.jsp" method="post">

账号：<input type="text" name="account" value="<%=account%>"> <br>

密码：<input type="password" name="password" value="<%=password%>"> <br>

自动登录：<input type="checkbox" name="remember" value="true"> <br>

<input type="submit" value="提交">

</form>

<script>

//使用js实现自动提交

if (ju === true){

document.forms[0].submit();

}

</script>

</body>

</html>

过滤器：checkFilter.java

package com.servlet;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.IOException;

//实现过滤器接口

public class check implements Filter {

@Override

public void doFilter(ServletRequest req, ServletResponse resp, FilterChain filterChain) throws IOException, ServletException {

//获取request、response、session对象

HttpServletRequest request = (HttpServletRequest) req;

HttpServletResponse response = (HttpServletResponse) resp;

HttpSession session = request.getSession();

String error = null;

//获取账号密码

String account = (String) req.getParameter("account");

String password = (String) req.getParameter("password");

//判断错误

if (!"123456".equals(account) && !"123456".equals(password)) {

error = "账号和密码";

} else if (!"123456".equals(account)) {

error = "账号";

} else if (!"123456".equals(password)) {

error = "密码";

}

if (error != null) {

//有错

session.setAttribute("error\_my", error);

System.out.println("有错");

response.sendRedirect("login.jsp");

} else {

//无错放行

Cookie cookie1 = new Cookie("account", account);

Cookie cookie2 = new Cookie("password", password);

response.addCookie(cookie1);

response.addCookie(cookie2);

session.setAttribute("rem", true);

session.removeAttribute("error\_my");

filterChain.doFilter(req, resp);

System.out.println("无错");

}

}

@Override

public void init(FilterConfig filterConfig) throws ServletException {

}

@Override

public void destroy() {

Filter.super.destroy();

}

}

监听器在web.xml中注册

<filter>

<filter-name>check</filter-name>

<filter-class>com.servlet.check</filter-class>

</filter>

<filter-mapping>

<filter-name>check</filter-name>

<url-pattern>/admin.jsp</url-pattern>

</filter-mapping>

Admin.jsp

<%--

Created by IntelliJ IDEA.

User: 18109

Date: 2022/9/27

Time: 20:32

To change this template use File | Settings | File Templates.

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>管理界面</title>

</head>

<body>

<h1>

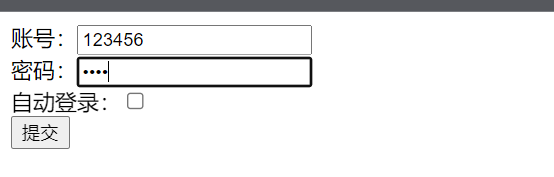
欢迎进入管理界面

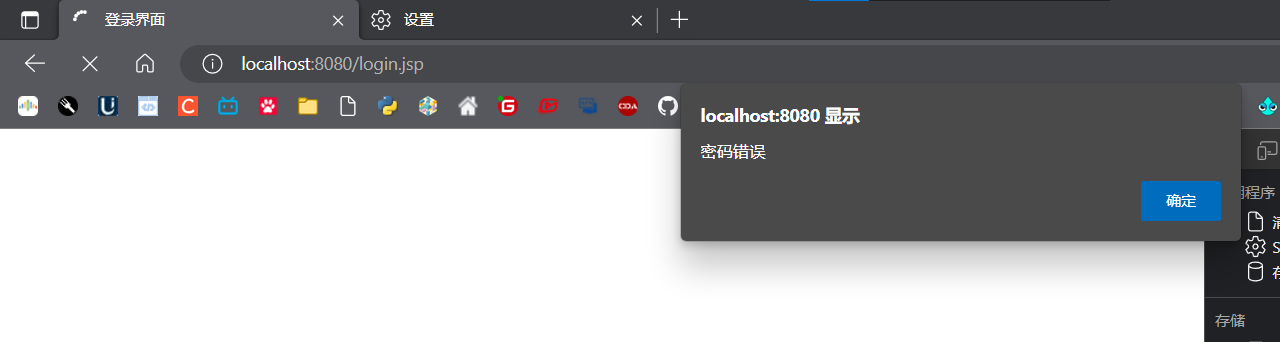
</h1>

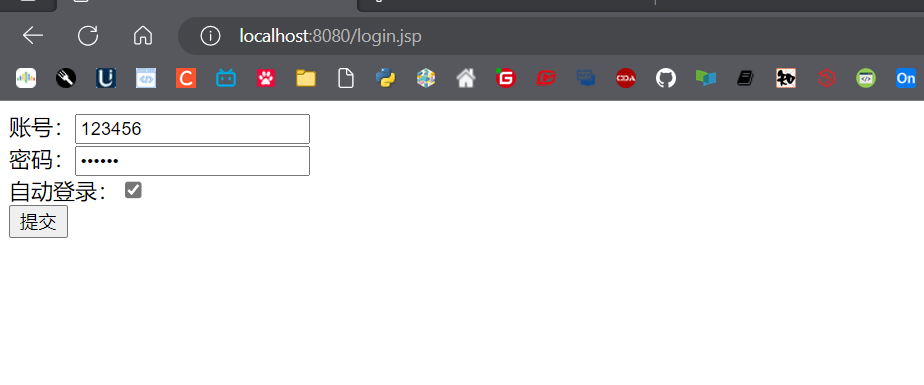
</body>

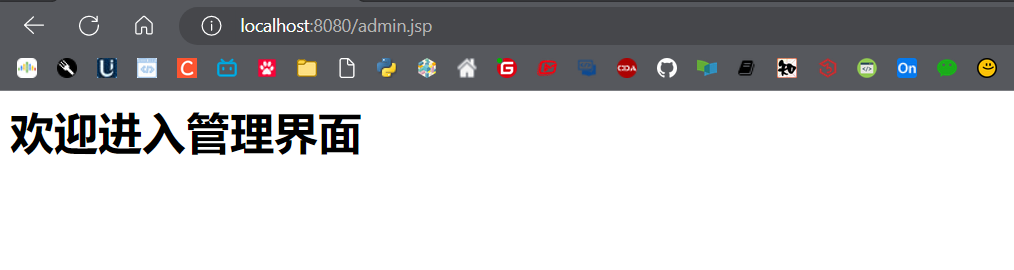
</html>

效果如下：









1. Servlet生命周期事件

使用生命周期事件统计当前应用的在线人数、启动时间、请求数。

创建一个监听器完成如上功能，代码如下：

**public class** EventListener **implements** ServletContextListener,

ServletRequestListener, HttpSessionListener

{

@Override

**public void** sessionCreated(HttpSessionEvent arg0) {

// 当有一个会话创建，则给当前在线人数加 1

**int** cnt = 0;

ServletContext context = arg0.getSession().getServletContext();

Object sessionCnt = context.getAttribute("sessionCnt");

**if**(sessionCnt != **null**){

cnt = (Integer)sessionCnt;

}

cnt++;

context.setAttribute("sessionCnt", cnt);

}

@Override

**public void** sessionDestroyed(HttpSessionEvent arg0) {

// 当有一个会话关闭时，则给当前在线人数减 1

**int** cnt = 0;

ServletContext context = arg0.getSession().getServletContext();

Object sessionCnt = context.getAttribute("sessionCnt");

**if**(sessionCnt != **null**){

cnt = (Integer)sessionCnt;

cnt--;

}

context.setAttribute("sessionCnt", cnt);

}

@Override

**public void** requestDestroyed(ServletRequestEvent arg0) {}

@Override

**public void** requestInitialized(ServletRequestEvent arg0) {

// 每有一个请求，则计数

**int** cnt = 0;

Object reqcnt = arg0.getServletContext().getAttribute("reqcnt");

**if**(reqcnt != **null**){

cnt = (Integer)reqcnt;

}

cnt++;

arg0.getServletContext().setAttribute("reqcnt", cnt);

}

@Override

**public void** contextDestroyed(ServletContextEvent arg0) {}

@Override

**public void** contextInitialized(ServletContextEvent arg0) {

// 保存应用启动时间

arg0.getServletContext().setAttribute("StartDate", **new** Date());

}

}

创建一个 jsp 页面显示统计结果:

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>

<!DOCTYPE>

<html>

<head>

<title>生命周期事件</title>

</head>

<body>

<p>当前应用启动时间：<%=application.getAttribute("StartDate") %></p>

<p>当前在线人数：<%=application.getAttribute("sessionCnt") %></p>

<p>处理请求数：<%=application.getAttribute("reqcnt") %></p>

</body>

</html>

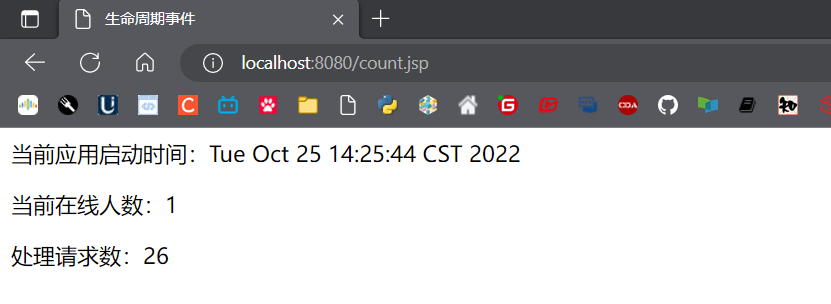
在web.xml界面中注册监听器

<listener>

<listener-class>com.servlet.EventListener</listener-class>

</listener>

效果如下：



1. 总结

分别写出基于配置文件与注解的Servlet和Filter的配置，并分析优缺点！

Web.xml配置文件：

<?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>get</servlet-name>

<servlet-class>com.servlet.get</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>get</servlet-name>

<url-pattern>/get</url-pattern>

</servlet-mapping>

<filter>

<filter-name>check</filter-name>

<filter-class>com.servlet.check</filter-class>

</filter>

<filter-mapping>

<filter-name>check</filter-name>

<url-pattern>/admin.jsp</url-pattern>

</filter-mapping>

<listener>

<listener-class>com.servlet.EventListener</listener-class>

</listener>

</web-app>

注解配置：

Servlet：

@WebServlet(name = "firstServlet",urlPatterns = "/serv")

public class FirstServlet extends HttpServlet {

Filter：

@WebFilter("/admin.jsp")

public class check implements Filter {

|  |  |  |
| --- | --- | --- |
|  | 配置文件 | 注解开发 |
| 优点 | 1. 无需修改源代码即可改变配置 2. 对象关系一目了然 3. 配置文件更加通俗易懂便于debug | 1. 注解和代码写在一起，降低了维护成本 2. 提高开发效率 |
| 缺点 | 1. 解析配置文件时占用系统资源 2. 需要在写源代码的同时写配置文件，开发效率变低 | 1. 修改麻烦，需要修改源代码 2. 不如配置文件简洁明了 3. 功能没有xml配置齐全 |

**实习报告提交地址：**[**www.zqldm.top:81**](http://www.zqldm.top:81)