

Servlet:

1. 概念
2. 步骤
3. 执行原理
4. 生命周期
5. Servlet3.0 注解配置
6. Servlet的体系结构
 - Servlet -- 接口
 - |
 - GenericServlet -- 抽象类
 - |
 - HttpServlet -- 抽象类
 - * GenericServlet: 将Servlet接口中其他的方法做了默认空实现, 只将service()方法作为抽象
 - * 将来定义Servlet类时, 可以继承GenericServlet, 实现service()方法即可
 - * HttpServlet: 对http协议的一种封装, 简化操作
 - 1. 定义类继承HttpServlet
 - 2. 复写doGet/doPost方法
7. Servlet相关配置
 1. urlpartten:Servlet访问路径
 1. 一个Servlet可以定义多个访问路径 : @WebServlet({" /d4", "/dd4", "/ddd4" })
 2. 路径定义规则:
 1. /xxx: 路径匹配
 2. /xxx/xxx: 多层路径, 目录结构
 3. *.do: 扩展名匹配

HTTP:

- * 概念: Hyper Text Transfer Protocol 超文本传输协议
 - * 传输协议: 定义了, 客户端和服务器端通信时, 发送数据的格式
 - * 特点:
 1. 基于TCP/IP的高级协议
 2. 默认端口号:80
 3. 基于请求/响应模型的:一次请求对应一次响应
 4. 无状态的: 每次请求之间相互独立, 不能交互数据
 - * 历史版本:
 - * 1.0: 每一次请求响应都会建立新的连接
 - * 1.1: 复用连接
- * 请求消息数据格式
 1. 请求行
 - 请求方式 请求url 请求协议/版本
 - GET /login.html HTTP/1.1
 - * 请求方式:
 - * HTTP协议有7中请求方式, 常用的有2种
 - * GET:
 1. 请求参数在请求行中, 在url后。
 2. 请求的url长度有限制的
 3. 不太安全
 - * POST:

```
1. 请求参数在请求体中
2. 请求的url长度没有限制的
3. 相对安全
2. 请求头: 客户端浏览器告诉服务器一些信息
  请求头名称: 请求头值
  * 常见的请求头:
    1. User-Agent: 浏览器告诉服务器, 我访问你使用的浏览器版本信息
      * 可以在服务器端获取该头的信息, 解决浏览器的兼容性问题

    2. Referer: http://localhost/login.html
      * 告诉服务器, 我(当前请求)从哪里来?
      * 作用:
        1. 防盗链:
        2. 统计工作:
3. 请求空行
  空行, 就是用于分割POST请求的请求头, 和请求体的。
4. 请求体(正文):
  * 封装POST请求消息的请求参数的
* 字符串格式:
  POST /login.html      HTTP/1.1
  Host: localhost
  User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:60.0) Gecko/20100101
Firefox/60.0
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
  Accept-Encoding: gzip, deflate
  Referer: http://localhost/login.html
  Connection: keep-alive
  Upgrade-Insecure-Requests: 1

  username=zhangsan
```

* 响应消息数据格式

Request:

- request对象和response对象的原理
 - request和response对象是由服务器创建的。我们来使用它们
 - request对象是来获取请求消息, response对象是来设置响应消息
- request对象继承体系结构:

```
ServletRequest      -- 接口
| 继承
HttpServletRequest  -- 接口
| 实现
org.apache.catalina.connector.RequestFacade 类(tomcat)
```
- request功能:
 - 获取请求消息数据
 - 获取请求行数据

```
* GET /day14/demo1?name=zhangsan HTTP/1.1
```
 - 方法:

1. 获取请求方式 : GET
 - * String getMethod()
2. (*)获取虚拟目录: /day14
 - * String getContextPath()
3. 获取Servlet路径: /demo1
 - * String getServletPath()
4. 获取get方式请求参数: name=zhangsan
 - * String getQueryString()
5. (*)获取请求URI: /day14/demo1
 - * String getRequestURI(): /day14/demo1
 - * StringBuffer getRequestURL() :http://localhost/day14/demo1
 - * URL:统一资源定位符 : http://localhost/day14/demo1 中华人民共和国
 - * URI: 统一资源标识符 : /day14/demo1 共和国
6. 获取协议及版本: HTTP/1.1
 - * String getProtocol()
7. 获取客户机的IP地址:
 - * String getRemoteAddr()
2. 获取请求头数据
 - * 方法:
 - * (*)String getHeader(String name):通过请求头的名称获取请求头的值
 - * Enumeration<String> getHeaderNames():获取所有的请求头名称
3. 获取请求体数据:
 - * 请求体: 只有POST请求方式, 才有请求体, 在请求体中封装了POST请求的请求参数
 - * 步骤:
 1. 获取流对象
 - * BufferedReader getReader(): 获取字符输入流, 只能操作字符数据
 - * ServletInputStream getInputStream(): 获取字节输入流, 可以操作所有类型数据
 - * 在文件上传知识点后讲解
 2. 再从流对象中拿数据

2. 其他功能:

1. 获取请求参数通用方式: 不论get还是post请求方式都可以使用下列方法来获取请求参数
 1. String getParameter(String name):根据参数名称获取参数值
username=zs&password=123
 2. String[] getParameterValues(String name):根据参数名称获取参数值的数组
hobby=xx&hobby=game
 3. Enumeration<String> getParameterNames():获取所有请求的参数名称
 4. Map<String,String[]> getParameterMap():获取所有参数的map集合
- * 中文乱码问题:
 - * get方式: tomcat 8 已经将get方式乱码问题解决了
 - * post方式: 会乱码
 - * 解决: 在获取参数前, 设置request的编码request.setCharacterEncoding("utf-8");

2. 请求转发：一种在服务器内部的资源跳转方式

1. 步骤：

1. 通过request对象获取请求转发器对象：RequestDispatcher

getRequestDispatcher(String path)

2. 使用RequestDispatcher对象来进行转发：forward(ServletRequest request, ServletResponse response)

2. 特点：

1. 浏览器地址栏路径不发生变化
2. 只能转发到当前服务器内部资源中。
3. 转发是一次请求

3. 共享数据：

- * 域对象：一个有作用范围的对象，可以在范围内共享数据
- * request域：代表一次请求的范围，一般用于请求转发的多个资源中共享数据
- * 方法：

1. void setAttribute(String name, Object obj): 存储数据
2. Object getAttribute(String name): 通过键获取值
3. void removeAttribute(String name): 通过键移除键值对

4. 获取ServletContext：

- * ServletContext getServletContext()

案例：用户登录

* 用户登录案例需求：

1. 编写login.html登录页面
username & password 两个输入框
2. 使用Druid数据库连接池技术, 操作mysql, day14数据库中user表
3. 使用JdbcTemplate技术封装JDBC
4. 登录成功跳转到SuccessServlet展示：登录成功！用户名, 欢迎您
5. 登录失败跳转到FailServlet展示：登录失败，用户名或密码错误

* 分析

* 开发步骤

1. 创建项目，导入html页面，配置文件，jar包
2. 创建数据库环境

```
CREATE DATABASE day14;
USE day14;
CREATE TABLE USER(
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(32) UNIQUE NOT NULL,
    PASSWORD VARCHAR(32) NOT NULL
);
```
3. 创建包cn.itcast.domain, 创建类User

```
package cn.itcast.domain;

/**
 * 用户的实体类
 */
```

```
public class User {

    private int id;
    private String username;
    private String password;
```

```
    setter,getter.....
}
```

4. 创建包cn.itcast.util,编写工具类JDBCUtils

```
package cn.itcast.util;
```

```
import com.alibaba.druid.pool.DruidDataSourceFactory;
```

```
import javax.sql.DataSource;
import javax.xml.crypto.Data;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Properties;
```

```
/**
```

```
 * JDBC工具类 使用Durid连接池
```

```
 */
```

```
public class JDBCUtils {
```

```
    private static DataSource ds ;
```

```
    static {
```

```
        try {
```

```
            //1.加载配置文件
```

```
            Properties pro = new Properties();
```

```
            //使用ClassLoader加载配置文件, 获取字节输入流
```

```
            InputStream is =
```

```
JDBCUtils.class.getClassLoader().getResourceAsStream("druid.properties");
```

```
            pro.load(is);
```

```
            //2.初始化连接池对象
```

```
            ds = DruidDataSourceFactory.createDataSource(pro);
```

```
        } catch (IOException e) {
```

```
            e.printStackTrace();
```

```
        } catch (Exception e) {
```

```
            e.printStackTrace();
```

```
        }
```

```
    }
```

```
/**
```

```
 * 获取连接池对象
```

```
 */
```

```
public static DataSource getDataSource(){
```

```
    return ds;
```

```
}
```

```
/**
 * 获取连接Connection对象
 */
public static Connection getConnection() throws SQLException {
    return ds.getConnection();
}
}
```

5. 创建包cn.itcast.dao,创建类UserDao,提供login方法

```
package cn.itcast.dao;

import cn.itcast.domain.User;
import cn.itcast.util.JDBCUtils;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;

/**
 * 操作数据库中User表的类
 */
public class UserDao {

    //声明JdbcTemplate对象共用
    private JdbcTemplate template = new JdbcTemplate(JDBCUtils.getDataSource());

    /**
     * 登录方法
     * @param loginUser 只有用户名和密码
     * @return user包含用户全部数据,没有查询到,返回null
     */
    public User login(User loginUser){
        try {
            //1.编写sql
            String sql = "select * from user where username = ? and password = ?";
            //2.调用query方法
            User user = template.queryForObject(sql,
                new BeanPropertyRowMapper<User>(User.class),
                loginUser.getUsername(), loginUser.getPassword());

            return user;
        } catch (DataAccessException e) {
            e.printStackTrace();//记录日志
            return null;
        }
    }
}
```

6. 编写cn.itcast.web.servlet.LoginServlet类

```
package cn.itcast.web.servlet;
```

```
import cn.itcast.dao.UserDao;
import cn.itcast.domain.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

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

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        //1.设置编码
        req.setCharacterEncoding("utf-8");
        //2.获取请求参数
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        //3.封装user对象
        User loginUser = new User();
        loginUser.setUsername(username);
        loginUser.setPassword(password);

        //4.调用UserDao的login方法
        UserDao dao = new UserDao();
        User user = dao.login(loginUser);

        //5.判断user
        if(user == null){
            //登录失败
            req.getRequestDispatcher("/failServlet").forward(req,resp);
        }else{
            //登录成功
            //存储数据
            req.setAttribute("user",user);
            //转发
            req.getRequestDispatcher("/successServlet").forward(req,resp);
        }
    }

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

7. 编写FailServlet和SuccessServlet类

```

@WebServlet("/successServlet")
public class SuccessServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        //获取request域中共享的user对象
        User user = (User) request.getAttribute("user");

        if(user != null){
            //给页面写一句话

            //设置编码
            response.setContentType("text/html;charset=utf-8");
            //输出
            response.getWriter().write("登录成功! "+user.getUsername()+" ,欢迎您");
        }
    }
}

```

```

}

```

```

@WebServlet("/failServlet")
public class FailServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        //给页面写一句话

        //设置编码
        response.setContentType("text/html;charset=utf-8");
        //输出
        response.getWriter().write("登录失败, 用户名或密码错误");

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        this.doPost(request,response);
    }
}

```


8. login.html中form表单的action路径的写法

* 虚拟目录+Servlet的资源路径

9. BeanUtils工具类，简化数据封装

* 用于封装JavaBean的

1. JavaBean：标准的Java类

1. 要求：

1. 类必须被public修饰
2. 必须提供空参的构造器
3. 成员变量必须使用private修饰
4. 提供公共setter和getter方法

2. 功能：封装数据

2. 概念：

成员变量：

属性：setter和getter方法截取后的产物

例如：getUsername() --> Username--> username

3. 方法：

1. setProperty()
2. getProperty()
3. populate(Object obj , Map map):将map集合的键值对信息，封装到对应的JavaBean对象中