

第九天 **Servlet**

今日内容介绍

- ◆ 案例一：完成用户登录的功能
- ◆ 案例二：记录系统登录成功后，系统被访问多少次

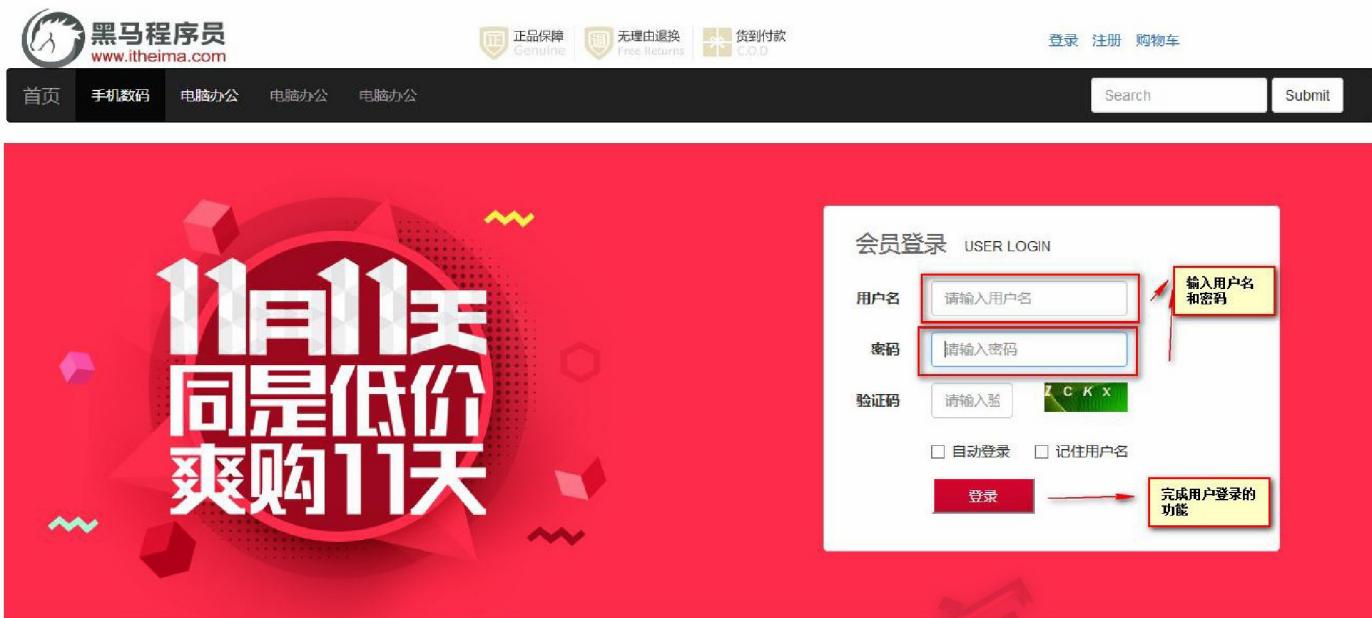
今日内容学习目标

- ◆ 可以独立编写 **Servlet** 程序
- ◆ 可以使用 `request` 接收请求参数
- ◆ 了解 `ServletConfig` 的使用
- ◆ 掌握 `ServletContext` 对象的使用

第1章 案例一：使用 **Servlet** 完成一个用户登录 的案例.

1.1 案例需求：

在网站的首页上，登录的链接，点击登录的链接，可以跳转到登录的页面，在登录的页面中输入用户名和密码点击登录的案例。完成登录的功能。



1.2 相关知识点：

1.2.1 Servlet 的概述

1.2.1.1 什么是 Servlet

Servlet 运行在服务端的 Java 小程序，是 sun 公司提供一套规范，用来处理客户端请求、响应给浏览器的动态资源。

Servlet 是 JavaWeb 三大组件之一（Servlet、Filter、Listener），且最重要。

1.2.1.2 Servlet 的作用

用来处理从客户端发送过来的请求，并对该请求作出响应。

Servlet 的任务有：

1. 获取请求数据
2. 处理请求
3. 完成响应

1.2.2 Servlet 的入门

1.2.2.1 准备工作

Servlet 规范要求：Servlet 程序需要编写实现类，并在 web.xml 进行配置。

- 实现类：通常继承 javax.servlet.http.HttpServlet 类，并复写 doGet 和 doPost 方法。
 - doGet()方法用于处理 get 请求。
 - doPost()方法用于处理 post 请求。

配置信息：在 web.xml 进行配置。

1.2.2.2 编写步骤

1. 创建类，继承 HttpServlet，复写 doGet 和 doPost 方法



```
package cn.itcast.demo01.a_servlet;

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

public class HelloServlet extends HttpServlet{

    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        System.out.println("get 请求将执行");
    }

    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        System.out.println("post 请求将执行");
    }
}
```

2. 编写配置文件



```
<!-- demo01 servlet hello start -->
<!-- demo01 servlet hello end -->
```

File structure:

- WebContent
 - 01.http
 - META-INF
 - WEB-INF
 - lib
 - web.xml

Content of web.xml:

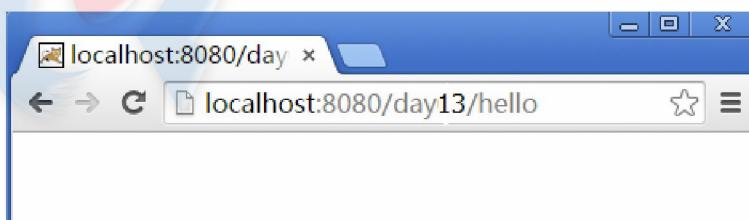
```
<!-- demo01 servlet hello start -->

<!-- demo01 servlet hello end -->
```

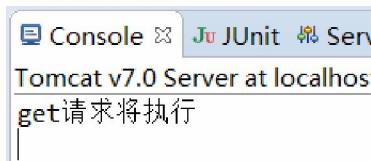
3. 浏览器访问

在浏览器地址栏输入：<http://localhost:8080/day13/hello>

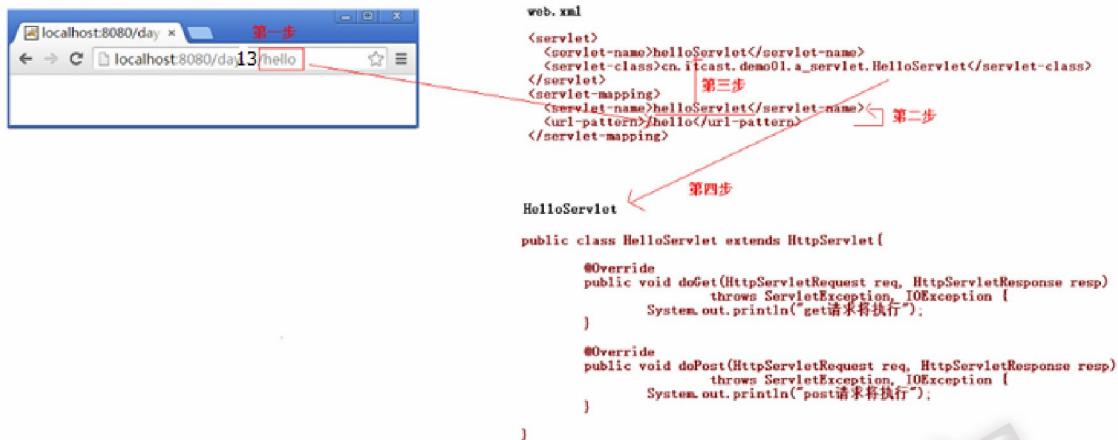
- 浏览器显示空白页面



- 控制打印信息



1.2.2.3 Servlet 的执行的流程详解:



1.2.3 Request 接收请求参数

1.2.3.1 Request 接收请求参数的概述

请求参数

`/day09/demo01ApiServlet?username=jack&password=1234` HTTP/1.1
 Accept: application/x-ms-application, image/jpeg, application/xml+xml, image/gif,
 Accept-Language: ar-AE,ja-JP;q=0.8,ko-KR;q=0.5,zh-CN;q=0.3
 User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0;
 Accept-Encoding: gzip, deflate
 Host: localhost:8080
 Connection: Keep-Alive
 username=jack&username=1234

GET 请求方式
 POST 请求方式
`getParameter(String name)`
`getParameterValues(String name)` 获得请求参数
`getParameterMap()`

方法名	描述
<code>String getParameter(String name)</code>	获得指定参数名对应的值。如果没有返回 null，如果有多个获得第一个。 例如: <code>username=jack</code>
<code>String[] getParameterValues(String name)</code>	获得指定参数名对应的所有的值。 例如: <code>hobby=抽烟&hobby=喝酒</code>
<code>Map<String, String[]> getParameterMap()</code>	获得所有的请求参数。 <code>key</code> 为参数名 <code>value</code> 为 <code>key</code> 对应的所有的值。
<code>setCharacterEncoding(String env)</code>	设置请求体的编码，用于解决 POST 请求参数乱码问题

1.2.3.2 Request 接收请求参数入门

- 编写步骤

1. 编写表单，提供表单字段：username、password、hobby，以 post 方式提交

```
▲ 📁 WebContent
  ▶ 📁 01.http
  ▲ 📁 02.request
    📜 form.html
```

```
<form action="..../demo01ParamServlet" method="post">
    用户名: <input type="text" name="username" value="jack" /> <br/>
    密码: <input type="text" name="password" value="1234" /> <br/>
    爱好: <input type="checkbox" name="hobby" value="抽烟" checked="checked"/> 抽烟
          <input type="checkbox" name="hobby" value="喝酒" checked="checked" /> 喝酒
          <input type="checkbox" name="hobby" value="烫头" /> 烫头 <br/>
    <input type="submit" value="post 提交" />
</form>
```

2. 编写 Servlet 实现类，使用对应方法获得请求参数，并处理 POST 中文乱码。

```
▲ 📁 cn.itcast.demo01.b_request
  ▶ 📜 Demo01ParamServlet.java
```

```
public class Demo01ParamServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

        //0 请求参数中文乱码
        request.setCharacterEncoding("UTF-8");

        //1 使用参数名获得一个值
        //1.1 获得用户名
        String username = request.getParameter("username");
        //1.2 获得密码
        String password = request.getParameter("password");
        System.out.println(username + " : " + password);

        //2 使用参数名获得一组值
        String[] hobbies = request.getParameterValues("hobby");
        System.out.println(Arrays.toString(hobbies));

        //3 获得所有数据，遍历 Map
        System.out.println("-----");
```

```
Map<String, String[]> allData = request.getParameterMap();
for (Map.Entry<String, String[]> entry : allData.entrySet()) {
    System.out.print(entry.getKey());
    System.out.print( " --> ");
    System.out.println(Arrays.toString(entry.getValue()));
}

}

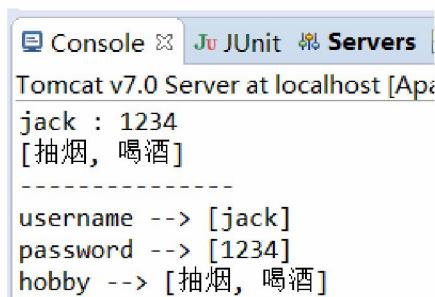
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    /* 开发中我们发现，doPost 和 doGet 两个方法编写的内容相同
     * 所以常使用一个调用另一个，此处我们使用 doPost 调用 doGet
     * 所有的内容之后都编写 doGet 方法中
     */
    doGet(request, response);
}

}
```

3. 编写 Servlet，配置文件

```
<servlet>
    <servlet-name>Demo01ParamServlet</servlet-name>
    <servlet-class>cn.itcast.demo01.b_request.Demo01ParamServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Demo01ParamServlet</servlet-name>
    <url-pattern>/demo01ParamServlet</url-pattern>
</servlet-mapping>
```

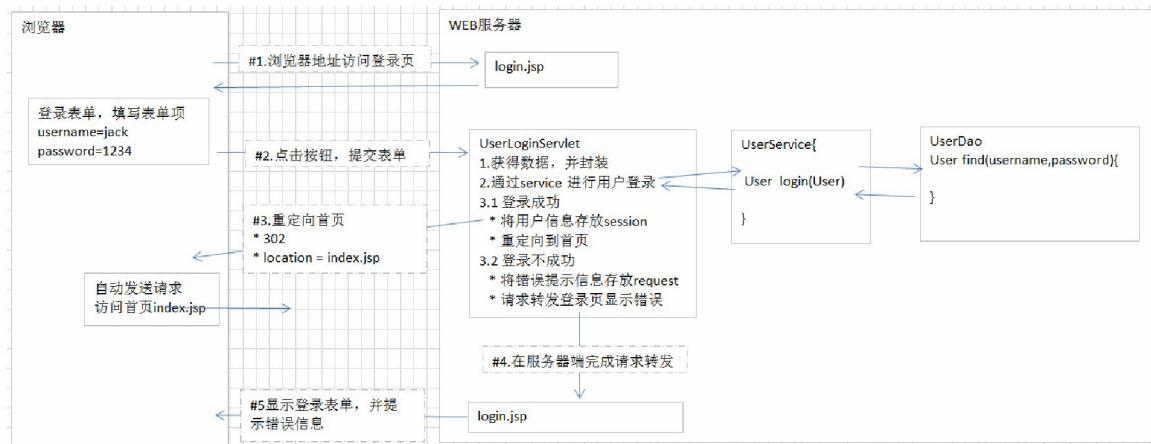
4. 测试，访问“<http://localhost:8080/day13/02.request/form.html>”，并提交表单。控制台显示



The screenshot shows the Eclipse IDE's 'Console' view with the following output:

```
Console JUnit Servers
Tomcat v7.0 Server at localhost [Apache]
jack : 1234
[抽烟, 喝酒]
-----
username --> [jack]
password --> [1234]
hobby --> [抽烟, 喝酒]
```

1.3 案例分析



1.4 代码实现

```
Servlet 的代码

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        try {
            // 1.接收表单提交的参数.
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            // 2.封装到实体对象中.
            User user = new User();
            user.setUsername(username);
            user.setPassword(password);

            // 3.调用业务层处理数据.
            UserService userService = new UserService();

            User existUser = userService.login(user);
            // 4.根据处理结果显示信息(页面跳转).
            if(existUser == null) {
                // 登录失败
                response.getWriter().println("Login Fail...");
            } else{
                // 登录成功
                response.getWriter().println("Login Success...");
            }
        }
    }
}
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

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

}
```

Service 的代码：

```
public class UserService {  
  
    /**  
     * 业务层用户登录的方法:  
     * @param user  
     * @return  
     * @throws SQLException  
     */  
  
    public User login(User user) {  
        // 调用 DAO 完成对数据的访问  
        UserDao userDao = ...  
        return userDao.login(user);  
    }  
}
```

DAO 的代码

```
public class UserDao {  
  
    /**  
     * 用户登录的 DAO 的方法:  
     * @param user  
     * @return  
     * @throws SQLException  
     */  
  
    public User login(User user) throws SQLException {  
        QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());  
        String sql = "select * from user where username=? and password=?";
```

```

        User existUser = queryRunner.query(sql, new BeanHandler<User>(User.class),
user.getUsername(), user.getPassword());
        return existUser;
    }

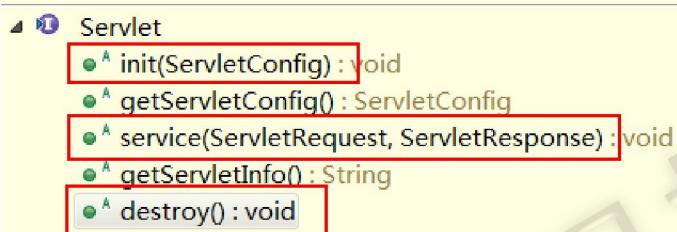
}

```

1.5 总结:

1.5.1 Servlet 的生命周期:(*****)

Servlet 规范规定，所有的 servlet 必须实现 javax.servlet.Servlet 接口。



- 1.第一次调用时，将执行初始化方法：init(ServletConfig)
- 2.每一次调用，都将执行 service(ServletRequest, ServletResponse)方法
- 3.服务器关闭，或项目移除：destroy()方法

JavaEE 规范中提供 Servlet 接口实现类：

GenericServlet: 通用 servlet 实现。没有实现 service

HttpServlet: 与 Http 协议有关的实现。实现 service 方法，完成与 http 协议有关的操作。

request.getMethod() 获得请求方式 (get、post)

如果是 get，将调用 doGet()

如果是 post，将调用 doPost()

生命周期:就是一个对象从创建到销毁的过程.

Servlet 生命周期:Servlet 从创建到销毁的过程.

* 何时创建:用户第一次访问 Servlet 创建 Servlet 的实例 (单实例)

* 何时销毁:当项目从服务器中移除的时候，或者关闭服务器的时候.

用户第一次访问 Servlet 的时候,服务器会创建一个 Servlet 的实例,那么 Servlet 中 init 方法就会执行.

任何一次请求服务器都会创建一个新的线程访问 Servlet 中的 service 的方法.在 service 方法内部根据请求的方式的不同调用 doXXX 的方法.(get 请求调用 doGet,post 请求调用 doPost).当 Servlet 服务器中移除掉,或者关闭服务器,Servlet 的实例就会被销毁,那么 destroy 方法就会执行.

1.5.2 Servlet 的相关的配置:

1.5.2.1 【启动时创建 Servlet】

Servlet 默认是在第一次访问的时候创建的.现在让 Servlet 在服务器启动的时候创建好.进行对 Servlet

的配置：

在 web.xml 中在<servlet></servlet>标签中配置：

* <load-on-startup>2</load-on-startup> --- 传入正整数，整数越小，被创建的优先级就越高。

1.5.2.2 【url-pattern 的配置】

url-pattern 配置方式共有三种：

1. 完全路径匹配 : 以 / 开始

例如：/ServletDemo4 , /aaa/ServletDemo5 , /aaa/bbb/ServletDemo6

2. 目录匹配 : 以 / 开始 需要以 * 结束.

例如：/* (所有) , /aaa/* (aaa 目录下的所有) , /aaa/bbb/*

3. 扩展名匹配 : 不能以 / 开始 以 * 开始的. 例如：*.do , *.action , *.jsp , *.jpg

***** 错误的写法 : /*.do

4. 缺省路径 /

通常情况访问 html 页面时，首先从当前 web 项目的 web.xml 文件寻找匹配路径，如果没有找到，再从 tomcat 默认的 web.xml 匹配，将使用缺省 servlet

tomcat 获得匹配路径时，**优先级顺序：1 > 2 > 3 > 4**

有如下的配置：

```
<servlet>
    <servlet-name>ServletDemo4</servlet-name>
    <servlet-class>com.itheima.a_servlet.ServletDemo4</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>ServletDemo4</servlet-name>
    <url-pattern>/ServletDemo4</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>ServletDemo5</servlet-name>
    <servlet-class>com.itheima.a_servlet.ServletDemo5</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>ServletDemo5</servlet-name>
    <url-pattern>*</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>ServletDemo6</servlet-name>
    <servlet-class>com.itheima.a_servlet.ServletDemo6</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>ServletDemo6</servlet-name>
```

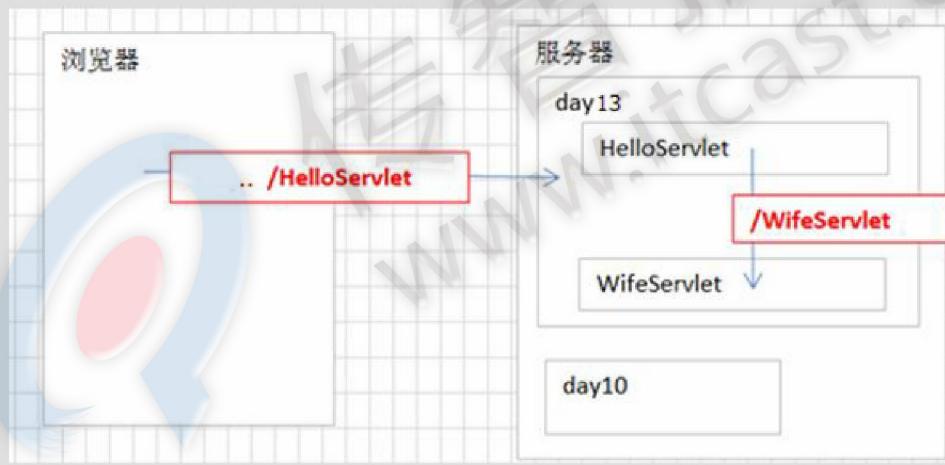
```
<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

如果访问地址：

<http://localhost:8080/day13/ServletDemo4> : 第一个
<http://localhost:8080/day13/aaa.do> : 第二个
***** 完全路径匹配 > 目录匹配 > 扩展名匹配

1.5.3 开发中的路径的编写：

- 相对路径：都是需要找位置相对关系，不能以 / 开始的。
 - * . / 当前路径 .. / 上一级目录
 - * 使用相对路径访问：
 - * <http://localhost:8080/day13/demo4-url/demol.html>
 - * <http://localhost:8080/day13/ServletDemo6>
- 绝对路径：不需要找位置相对关系，以 / 开始的。
 - * 绝对路径中分为客户端路径和服务器端路径：
 - * 客户端路径一定要加工程名。 /day13/ServletDemo6
 - * 服务器端路径不需要加工程名。 /ServletDemo6



第2章 案例三：记录网站的登录成功的人数。

2.1 案例需求：

登录成功后，5 秒后跳转到某个页面，在页面中显示您是第 x 位登录成功的用户。

您是第1位登录成功的用户！

2.2 相关知识点：

2.2.1 ServletContext

服务器启动的时候,为每个 WEB 应用创建一个单独的 ServletContext 对象,我们可以使用这个对象存取数据,用这个对象存取的数据可以在整个 WEB 应用中获得。可以使用如下方法存取数据向 ServletContext 中存数据

void	setAttribute (String name, Object object)
Binds an object to a given attribute name in this servlet context.	

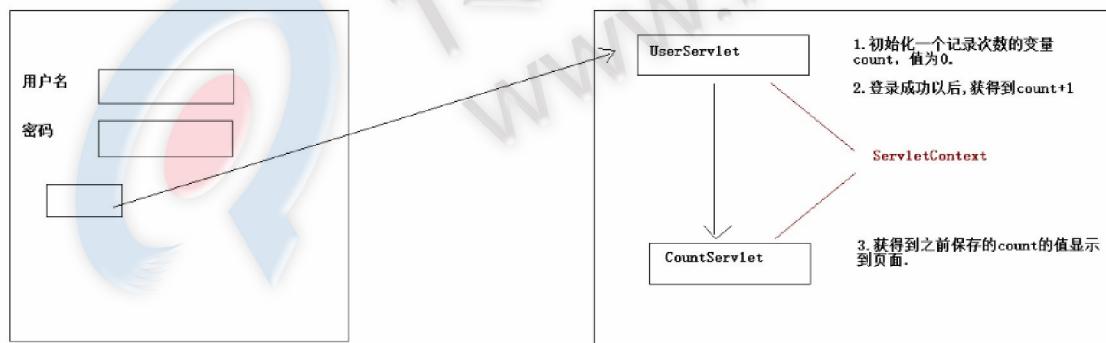
从 ServletContext 中取数据

Object	getAttribute (String name)
Returns the servlet container attribute with the given name, or null if there is no attribute by that name.	

从 ServletContext 中移除数据

void	removeAttribute (String name)
Removes the attribute with the given name from the servlet context.	

2.3 案例分析：



2.4 代码实现：

```
/**  
 * 登录代码的 Servlet  
 */  
public class UserCountServlet extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
  
    @Override
```

```
public void init() throws ServletException {
    // 初始化一个变量 count 的值为 0.
    int count = 0;
    // 将这个值存入到 ServletContext 中.
    this.getServletContext().setAttribute("count", count);
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    try {
        response.setContentType("text/html;charset=UTF-8");
        // 1.接收表单提交的参数.
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        // 2.封装到实体对象中.
        User user = new User();
        user.setUsername(username);
        user.setPassword(password);

        // 3.调用业务层处理数据.
        UserService userService = new UserService();

        User existUser = userService.login(user);
        // 4.根据处理结果显示信息(页面跳转).
        if(existUser == null){
            // 登录失败
            response.getWriter().println("<h1>登录失败: 用户名或密码错误!</h1>");
        }else{
            // 登录成功

            // 记录次数:
            int count = (int) this.getServletContext().getAttribute("count");
            count++;
            this.getServletContext().setAttribute("count", count);

            response.getWriter().println("<h1>      登      录      成      功      :      您
好:"+existUser.getNickname()+"</h1>");
            response.getWriter().println("<h3>页面将在 5 秒后跳转!</h3>");
            response.setHeader("Refresh", "5;url=/day09/CountServlet");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```

    }

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

}

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

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// 获得 Count 的值。
response.setContentType("text/html;charset=UTF-8");
int count = (int) this.getServletContext().getAttribute("count");
response.getWriter().println("<h1>您是第"+count+"位登录成功的用户! </h1>");
}

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

}

```

2.5 总结：

2.5.1 ServletConfig:获得 Servlet 的配置信息(了解)

<code>String</code>	<code>getInitParameter(String name)</code> Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
<code>Enumeration</code>	<code>getInitParameterNames()</code> Returns the names of the servlet's initialization parameters as an Enumeration of String objects.
<code>ServletContext</code>	<code>getServletContext()</code> Returns a reference to the ServletContext in which the caller is executing.
<code>String</code>	<code>getServletName()</code> Returns the name of this servlet instance.

- * `String getServletName();` ---获得 Servlet 在 web.xml 中配置的 name 的值.
- * `String getInitParameter(String name);` ---获得 Servlet 的初始化参数的.
- * `Enumeration getInitParameterNames();` ---获得所有 Servlet 的初始化参数的名称.

2.5.2 ServletContext: 读取 WEB 工程下的文件

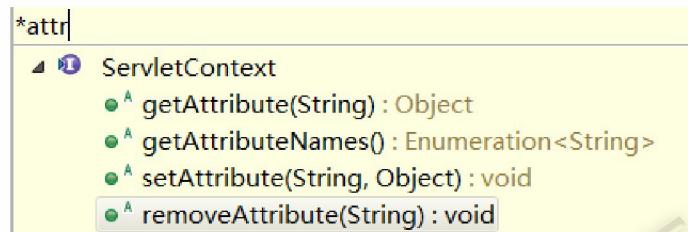
ServletContext 对象，tomcat 为每一个 web 项目单独创建的一个上下文（知上知下贯穿全文）对象。就有功能：

- 可以在多个 servlet 之间共享数据

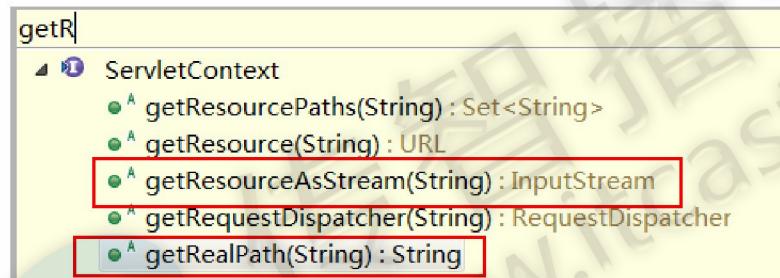
存放：setAttribute()

获得：getAttribute()

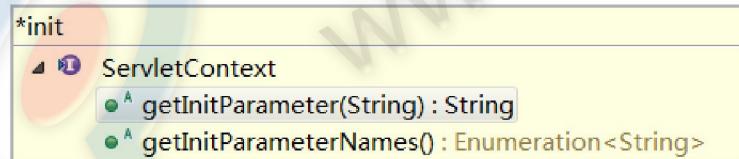
删除：removeAttribute()



- 可以获得当前 WEB 项目中的指定资源（文件）



- 可以进行整个 web 项目初始化数据设置



在 `web.xml` 可以给整个 web 项目配置初始化参数

```

13   <!-- 全局初始化参数（整个项目） -->
14  <context-param>
15      <param-name>参数名</param-name>
16      <param-value>参数值</param-value>
17  </context-param>
18
  
```

```
<!-- 全局初始化参数（整个项目） -->
```

```

<context-param>
    <param-name>参数名</param-name>
    <param-value>参数值</param-value>
</context-param>
  
```

在实际开发中，有时候可能会需要读取 Web 应用中的一些资源文件，比如配置文件，图片等。为此，在 `ServletContext` 接口中定义了一些读取 Web 资源的方法，这些方法是依靠 Servlet 容器来实现的。Servlet 容器根据资源文件名相对于 Web 应用的路径，返回关联资源文件的 IO 流、资源文件

在文件系统的绝对路径等。

方法说明	功能描述
<code>Set getResourcePaths(String path)</code>	返回一个 Set 集合，集合中包含资源目录中子目录和文件的路径名称。参数 path 必须以正斜线(/)开始，指定匹配资源的部分路径
<code>String getRealPath(String path)</code>	返回资源文件在服务器文件系统上的真实路径(文件的绝对路径)。参数 path 代表资源文件的虚拟路径，它应该以正斜线开始(/)开始，“/”表示当前 Web 应用的根目录，如果 Servlet 容器不能将虚拟路径转换为文件系统的真实路径，则返回 null
<code>URL getResource(String path)</code>	返回映射到某个资源文件的 URL 对象。参数 path 必须以正斜线(/)开始，“/”表示当前 Web 应用的根目录
<code>InputStream getResourceAsStream(String path)</code>	返回映射到某个资源文件的 InputStream 输入流对象。参数 path 传递规则和 getResource() 方法完全一致

了解了 ServletContext 接口中用于获得 Web 资源路径的方法后，接下来通过一个案例，分步骤演示如何使用 ServletContext 对象读取资源文件

2.5.3 代码实现

- 加载配置文件

```

1 package cn.itcast.servlet;
2 import java.io.*;
3 import java.util.Properties;
4 import javax.servlet.*;
5 import javax.servlet.http.*;
6 public class ReadFileServlet extends HttpServlet {
7     public void doGet(HttpServletRequest request,
8         HttpServletResponse response) throws ServletException, IOException {
9         ServletContext context = this.getServletContext();
10        PrintWriter out = response.getWriter();
11        //获取相对路径中的输入流对象
12        InputStream in = context
13            .getResourceAsStream("/WEB-INF/classes/itcast.properties");
14        Properties pros = new Properties();
15        pros.load(in);
16        out.println("Company=" + pros.getProperty("Company") + "<br>");
17        out.println("Address=" + pros.getProperty("Address") + "<br>");
18    }
19    public void doPost(HttpServletRequest request,
20        HttpServletResponse response) throws ServletException, IOException {
21        this.doGet(request, response);

```

```

22     }
23 }
```

- 开发者需要获取的是资源的绝对路径。接下来，通过使用 `getRealPath(String path)`方法获取资源文件的绝对路径。

```

1 package cn.itcast.servlet;
2 import java.io.*;
3 import java.util.Properties;
4 import javax.servlet.*;
5 import javax.servlet.http.*;
6 public class ReadFileServlet extends HttpServlet {
7     public void doGet(HttpServletRequest request,
8         HttpServletResponse response) throws ServletException, IOException {
9         PrintWriter out = response.getWriter();
10        ServletContext context = this.getServletContext();
11        //获取文件绝对路径
12        String path = context
13            .getRealPath("/WEB-INF/classes/itcast.properties");
14        FileInputStream in = new FileInputStream(path);
15        Properties pros = new Properties();
16        pros.load(in);
17        out.println("Company=" + pros.getProperty("Company") + "<br>");
18        out.println("Address=" + pros.getProperty("Address") + "<br>");
19    }
20    public void doPost(HttpServletRequest request,
21        HttpServletResponse response) throws ServletException, IOException {
22        this.doGet(request, response);
23    }
24 }
25 }
```

2.5.4 初始化参数

- servlet 的初始化参数

```

12<servlet>
13  <servlet-name>UserLoginServlet</servlet-name>
14  <servlet-class>com.itheima.web.servlet.UserLoginServlet</servlet-class>
15  <!-- 初始化参数 -->
16<init-param>
17  <param-name></param-name>
18  <param-value></param-value>
19</init-param>
20</servlet>
```

通过 `ServletConfig` 对象获得

- 整个项目的初始化参数

```
13  <!-- 全局初始化参数（整个项目） -->
14<context-param>
15  <param-name>参数名</param-name>
16  <param-value>参数值</param-value>
17 </context-param>
18
```

通过 ServletContext 对象获得

2.5.5 扩展：类加载器读取文件

```
public static void readFile() throws IOException{
    // 使用类的加载器来读取文件。
    // 类的加载器用来加载 class 文件，将 class 文件加载到内存。
    InputStream is =
ReadFileUtils.class.getClassLoader().getResourceAsStream("db.properties");
    Properties properties = new Properties();
    properties.load(is);

    String driverClass = properties.getProperty("driverClass");
    String url = properties.getProperty("url");
    String username = properties.getProperty("username");
    String password = properties.getProperty("password");

    System.out.println(driverClass);
    System.out.println(url);
    System.out.println(username);
    System.out.println(password);
}
```