今日任务

1.Servlet技术

1.1 什么是Servlet

1. Servlet 是 JavaEE 规范之一。规范就是接口
2. Servlet 就 JavaWeb 三大组件之一。三大组件分别是：Servlet 程序、Filter 过滤器、Listener 监听器。
3. Servlet 是运行在服务器上的一个 java 小程序，它可以接收客户端发送过来的请求，并响应数据给客户端。

1.2 手动实现Servlet程序

1. 编写一个类去实现 Servlet 接口
2. 实现 service 方法，处理请求，并响应数据
3. 到 web.xml 中去配置 servlet 程序的访问地址

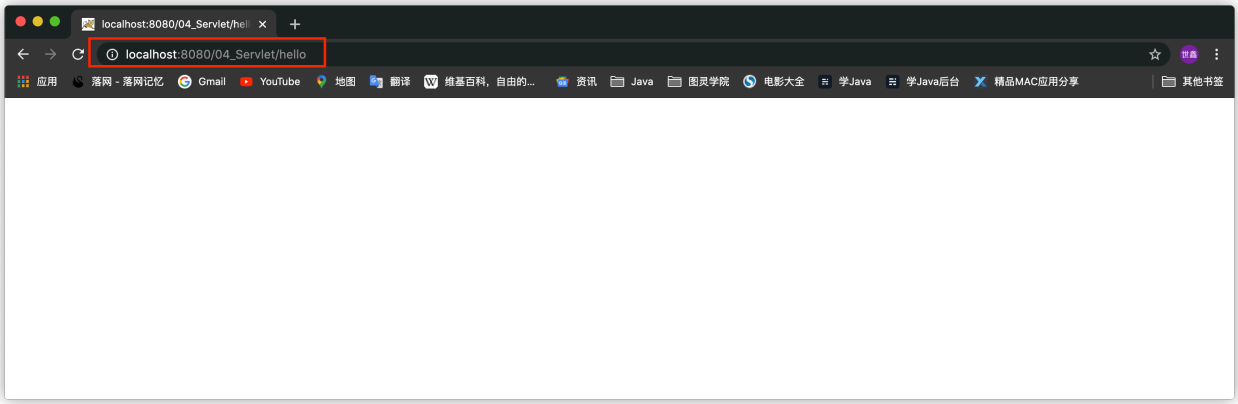
Servlet 程序的示例代码：

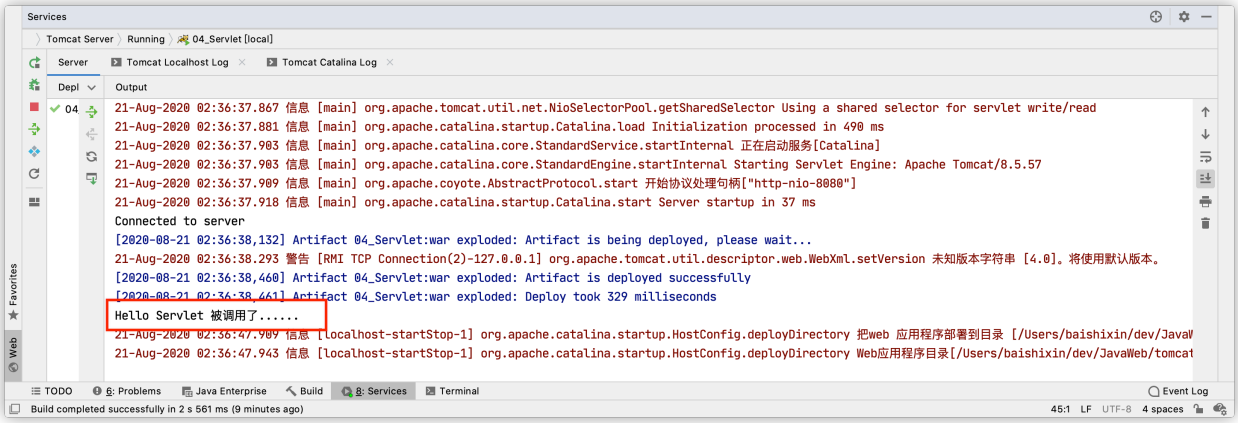
```
package com.admiral.servlet;

import javax.servlet.*;
import java.io.IOException;

public class HelloServlet implements Servlet {
    @Override
    public void init(ServletConfig servletConfig) throws ServletException {

    }

    @Override
    public ServletConfig getServletConfig() {
        return null;
    }


    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {
        System.out.println("Hello Servlet 被调用了......");
    }
```

```
    @Override
    public String getServletInfo() {
        return null;
    }

    @Override
    public void destroy() {

    }
}
```

web.xml中的代码

```
<?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>
        <!--    给Servlet起的一个别名(通常我们就使用Servlet名)    -->
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>com.admiral.servlet.HelloServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

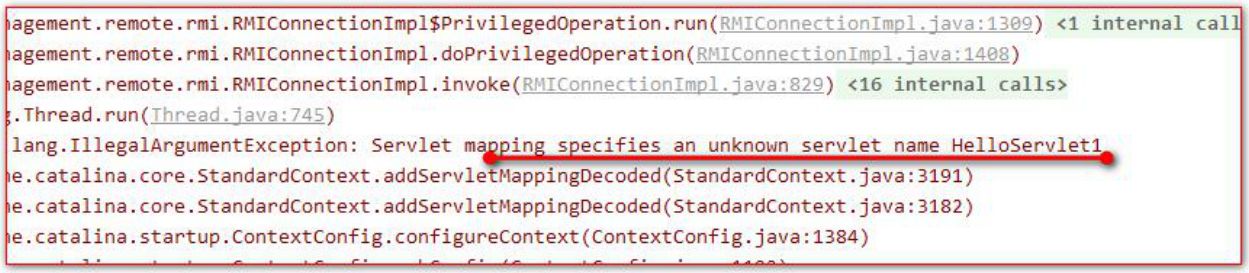




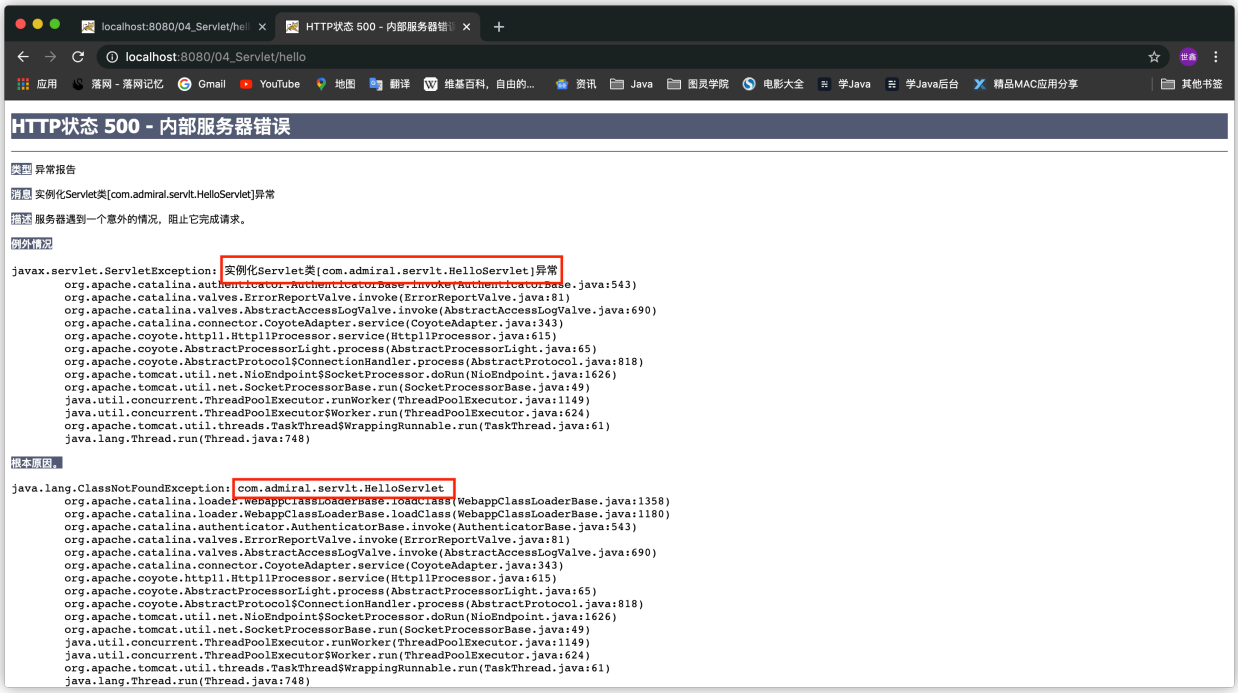
1.3 常见的错误

1.3.1 url-pattern 中配置的路径没有以斜杠打头。

1.3.2 servlet-name 配置的值不存在：



1.3.3 servlet-class 标签的全类名配置错误：



1.4 url 地址到 Servlet 程序的访问

1.5 Servlet 的生命周期

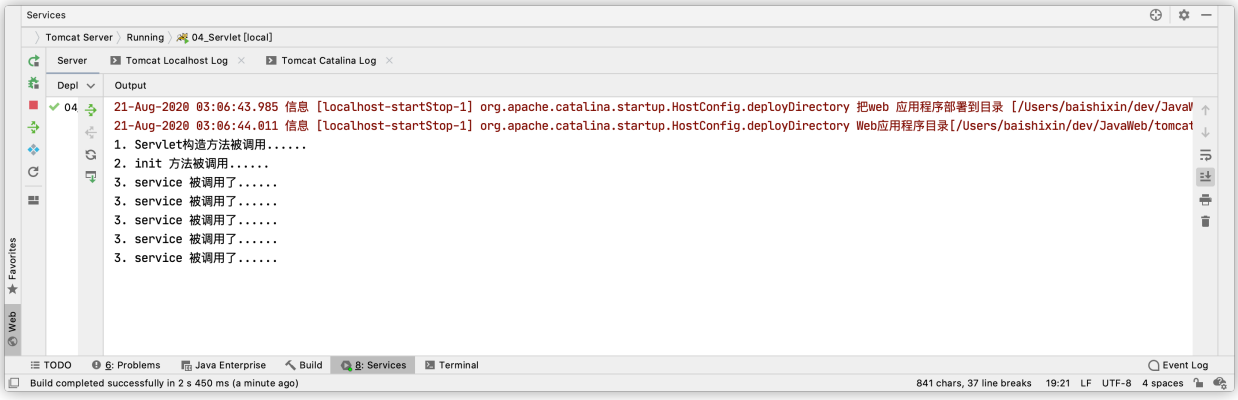
1. 执行 Servlet 构造器方法
2. 执行 init 初始化方法
3. 执行 service 方法
4. 执行 destroy 销毁方法

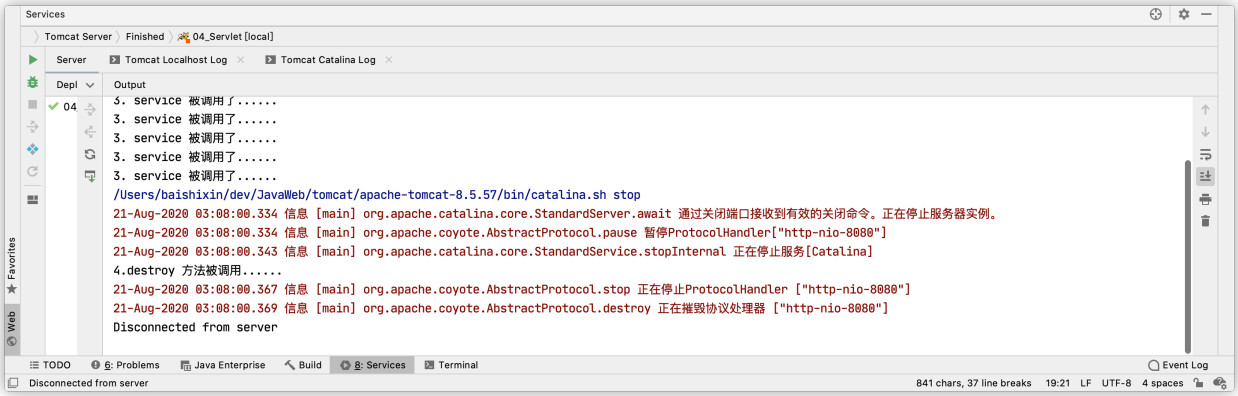
第一、二步，是在第一次访问，的时候创建 Servlet 程序会调用。
第三步，每次访问都会调用。
第四步，在 web 工程停止的时候调用。

示例代码:

```
package com.admiral.servlet;

import javax.servlet.*;
import java.io.IOException;

public class HelloServlet implements Servlet {
    public HelloServlet() {
        System.out.println("1. Servlet构造方法被调用.....");
    }
```

```
    @Override
    public void init(ServletConfig servletConfig) throws ServletException {
        System.out.println("2. init 方法被调用......");

    }

    @Override
    public ServletConfig getServletConfig() {
        return null;
    }


    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {
        System.out.println("3. service 被调用了......");
    }

    @Override
    public String getServletInfo() {
        return null;
    }

    @Override
    public void destroy() {
        System.out.println("4.destroy 方法被调用......");
    }
}
```





1.6 GET 和 POST 请求的分发处理

```
package com.admiral.servlet;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

public class HelloServlet implements Servlet {
    public HelloServlet() {
        System.out.println("1. Servlet构造方法被调用......");
    }

    @Override
    public void init(ServletConfig servletConfig) throws ServletException {
        System.out.println("2. init 方法被调用......");

    }

    @Override
    public ServletConfig getServletConfig() {
        return null;
    }


    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {
        System.out.println("3. service 被调用了......");
        HttpServletRequest httpServletRequest = (HttpServletRequest)
servletRequest;
        String method = httpServletRequest.getMethod();
        if("GET".equals(method)){
            doGet();
```

```
        }else if("POST".equals(method)){
            doPost();
        }

    }

    public void doGet(){
        System.out.println("doGet");
        System.out.println("doGet");
    }

    public void doPost(){
        System.out.println("doPost");
        System.out.println("doPost");
    }

    @Override
    public String getServletInfo() {
        return null;
    }

    @Override
    public void destroy() {
        System.out.println("4.destroy 方法被调用......");
    }
}
```

1.7 HttpServlet 实现 Servlet 程序

一般在实际项目开发中，都是使用继承 HttpServlet 类的方式去实现 Servlet 程序。

1、编写一个类去继承 HttpServlet 类

2、根据业务需要重写 doGet 或 doPost 方法

3、到 web.xml 中的配置 Servlet 程序的访问地址

```
package com.admiral.servlet;

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

```
/**
 * @author 白世鑫
 * @title: HelloServlet2
 * @projectName JavaWeb
 * @description:
 * @date 2020/8/21  3:55 上午
 */
public class HelloServlet2 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("HelloServlet2 doGet 方法");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("HelloServlet2 doPost 方法");
    }
}
```

1.8 使用 IDEA 创建 Servlet 程序

2. ServletConfig 类

ServletConfig 类从类名上来看，就知道是 Servlet 程序的配置信息类。

Servlet 程序和 ServletConfig 对象都是由 Tomcat 负责创建，我们负责使用。

Servlet 程序默认是第一次访问的时候创建，ServletConfig 是每个 Servlet 程序创建时，就创建一个对应的 ServletConfig 对 象。

2.1 ServletConfig 类的三大作用

1. 可以获取 Servlet 程序的别名 servlet-name 的值
2. 获取初始化参数 init-param
3. 获取 ServletContext 对象

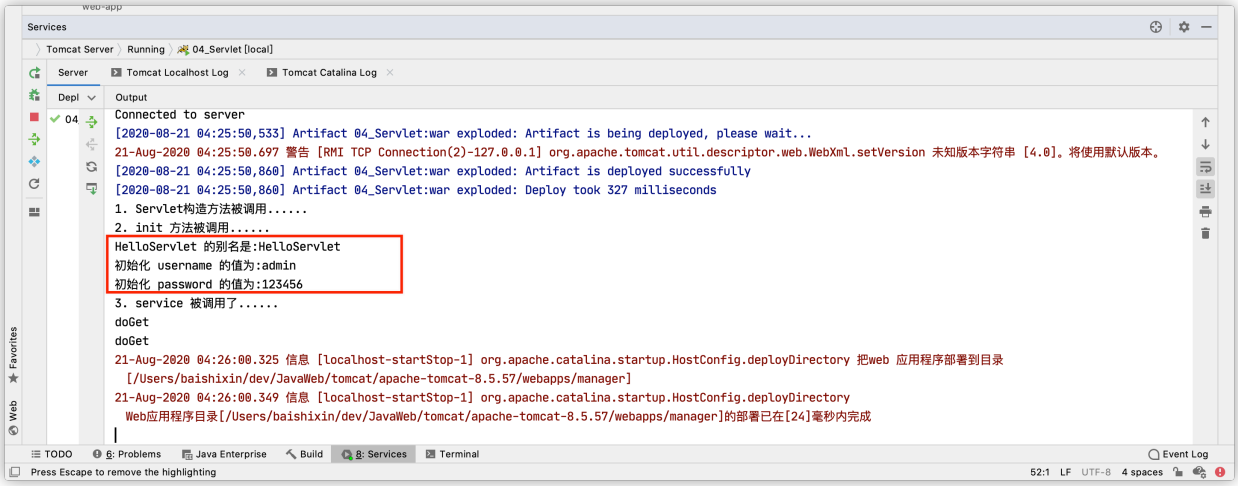
Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<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>
        <!--    给Servlet起的一个别名(通常我们就使用Servlet名)    -->
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>com.admiral.servlet.HelloServlet</servlet-class>
        <init-param>
            <param-name>username</param-name>
            <param-value>admin</param-value>
        </init-param>
        <init-param>
            <param-name>password</param-name>
            <param-value>123456</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

```java
package com.admiral.servlet;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

public class HelloServlet implements Servlet {
    public HelloServlet() {
        System.out.println("1. Servlet构造方法被调用......");
    }

    @Override
    public void init(ServletConfig servletConfig) throws ServletException {
        System.out.println("2. init 方法被调用......");
        String servletName = servletConfig.getServletName();
        System.out.println("HelloServlet 的别名是:" + servletName);


        String username = servletConfig.getInitParameter("username");
        System.out.println("初始化 username 的值为:" + username);
```

```
        String password = servletConfig.getInitParameter("password");
        System.out.println("初始化 password 的值为:" + password);

        ServletContext servletContext = servletConfig.getServletContext();
        System.out.println(servletContext);
    }

    @Override
    public ServletConfig getServletConfig() {
        return null;
    }


    @Override
    public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {
        System.out.println("3. service 被调用了......");
        HttpServletRequest httpServletRequest = (HttpServletRequest)
servletRequest;
        String method = httpServletRequest.getMethod();
        if("GET".equals(method)){
            doGet();
        }else if("POST".equals(method)){
            doPost();
        }

    }

    public void doGet(){
        System.out.println("doGet");
        System.out.println("doGet");
    }

    public void doPost(){
        System.out.println("doPost");
        System.out.println("doPost");
    }

    @Override
    public String getServletInfo() {
        return null;
    }

    @Override
    public void destroy() {
        System.out.println("4.destroy 方法被调用......");
    }
}
```



3.ServletContext 类

3.1 什么是 ServletContext

1. ServletContext 是一个接口，它表示 Servlet 上下文对象
2. 一个 web 工程，只有一个 ServletContext 对象实例。
3. ServletContext 对象是一个域对象。
4. ServletContext 是在 web 工程部署启动的时候创建。在 web 工程停止的时候销毁。

什么是域对象?

域对象，是可以像 Map 一样存取数据的对象，叫域对象。

这里的域指的是存取数据的操作范围，整个 web 工程。

	存数据	取数据	删除数据
Map	put()	get()	remove()
域对象	setAttribute()	getAttribute()	removeAttribute()

3.2 ServletContext 类的四个作用

1. 取 web.xml 中配置的上下文参数 context-param
2. 获取当前的工程路径，格式: /工程路径
3. 获取工程部署后在服务器硬盘上的绝对路径
4. 像 Map 一样存取数据

Web.xml

```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">

    <context-param>
        <param-name>username</param-name>
        <param-value>xiaobaibai</param-value>
    </context-param>
    <context-param>
        <param-name>password</param-name>
        <param-value>woaini</param-value>
    </context-param>


    <servlet>
        <!--    给Servlet起的一个别名(通常我们就使用Servlet名)    -->
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>com.admiral.servlet.HelloServlet</servlet-class>
        <init-param>
            <param-name>username</param-name>
            <param-value>admin</param-value>
        </init-param>
        <init-param>
            <param-name>password</param-name>
            <param-value>123456</param-value>
        </init-param>
    </servlet>
    <servlet>
        <servlet-name>HelloServlet2</servlet-name>
        <servlet-class>com.admiral.servlet.HelloServlet2</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ContextServlet</servlet-name>
        <servlet-class>com.admiral.servlet.ContextServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ContextServlet1</servlet-name>
        <servlet-class>com.admiral.servlet.ContextServlet1</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ContextServlet2</servlet-name>
        <servlet-class>com.admiral.servlet.ContextServlet2</servlet-class>
    </servlet>


    <servlet-mapping>
```

```
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>HelloServlet2</servlet-name>
        <url-pattern>/hello2</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ContextServlet</servlet-name>
        <url-pattern>/context</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ContextServlet1</servlet-name>
        <url-pattern>/context1</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ContextServlet2</servlet-name>
        <url-pattern>/context2</url-pattern>
    </servlet-mapping>
</web-app>
```

ContextServlet

```
package com.admiral.servlet;

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

/**
 * @author 白世鑫
 * @title: HelloServlet2
 * @projectName JavaWeb
 * @description:
 * @date 2020/8/21  3:55 上午
 */
public class ContextServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("HelloServlet2 doGet 方法");

        //1.获取 web.xml 配置文件中配置的上下文参数 context-param
```

```
        ServletContext servletContext = getServletContext();
        String username = servletContext.getInitParameter("username");
        String password = servletContext.getInitParameter("password");
        System.out.println("context-param 参数 username 的值为:" + username);
        System.out.println("context-param 参数 password 的值为:" + password);

        //2.获取当前工程路径
        System.out.println("当前工程的路径为:" +
servletContext.getContextPath());

        System.out.println("当前工程的绝对路径为:" +
servletContext.getRealPath("/"));



    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("HelloServlet2 doPost 方法");
    }
}
```

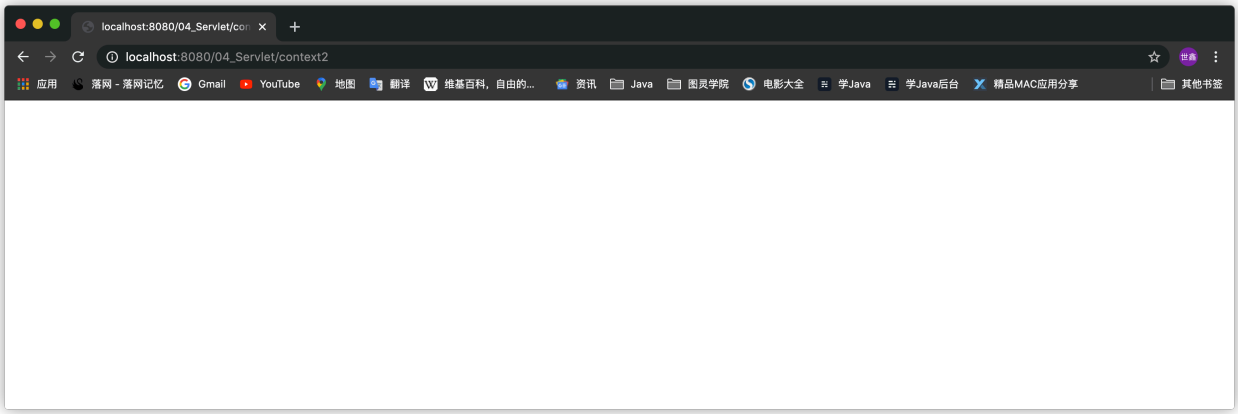
ContextServlet1

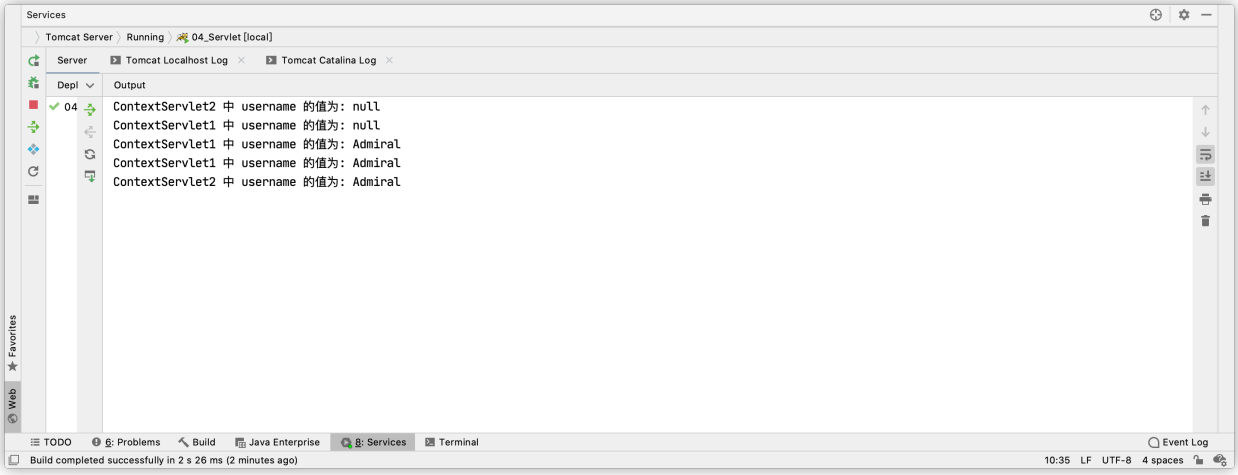
```
package com.admiral.servlet;

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

public class ContextServlet1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        ServletContext servletContext = getServletContext();
```

```
        System.out.println("ContextServlet1 中 username 的值为: " +
servletContext.getAttribute("username"));

        servletContext.setAttribute("username","Admiral");

        System.out.println("ContextServlet1 中 username 的值为: " +
servletContext.getAttribute("username"));
        System.out.println("ContextServlet1 中 username 的值为: " +
servletContext.getAttribute("username"));
    }
}
```

ContextServlet2

```
package com.admiral.servlet;

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

public class ContextServlet2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        ServletContext servletContext = getServletContext();
        System.out.println("ContextServlet2 中 username 的值为: " +
servletContext.getAttribute("username"));
    }
}
```





4.HTTP 协议

4.1 什么是 HTTP 协议

什么是协议?

协议是指双方，或多方，相互约定好，大家都需要遵守的规则，叫协议。 所谓 HTTP 协议，就是指，客户端和服务器之间通信时，发送的数据，需要遵守的规则，叫 HTTP 协议。

HTTP 协议中的数据又叫报文。

4.2 请求的 HTTP 协议格式

客户端给服务器发送数据叫请求。

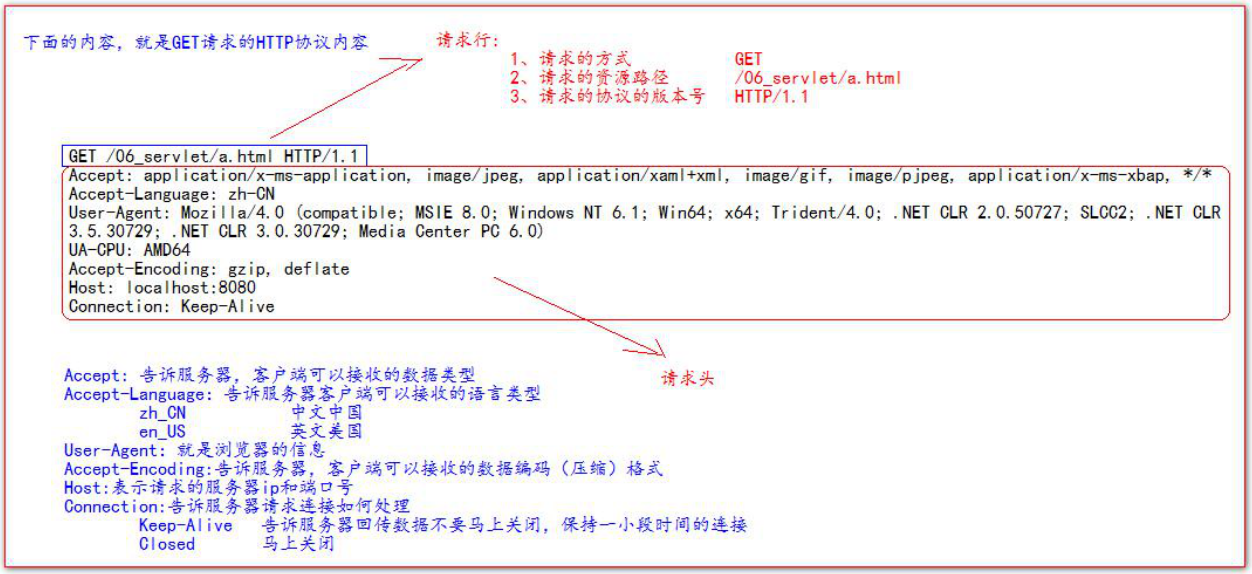
服务器给客户端回传数据叫响应。

请求又分为 GET 请求，和 POST 请求两种

4.2.1 GET 请求

1. 请求行
    1. 请求的方式 GET
    2. 请求的资源路径[+?+请求参数]
    3. 请求的协议的版本号 HTTP/1.1
2. 请求头

    key : value 组成 不同的键值对，表示不同的含义。



4.2.2 POST 请求

1. 请求行

    (1) 请求的方式 POST

    (2) 请求的资源路径[+?+请求参数]

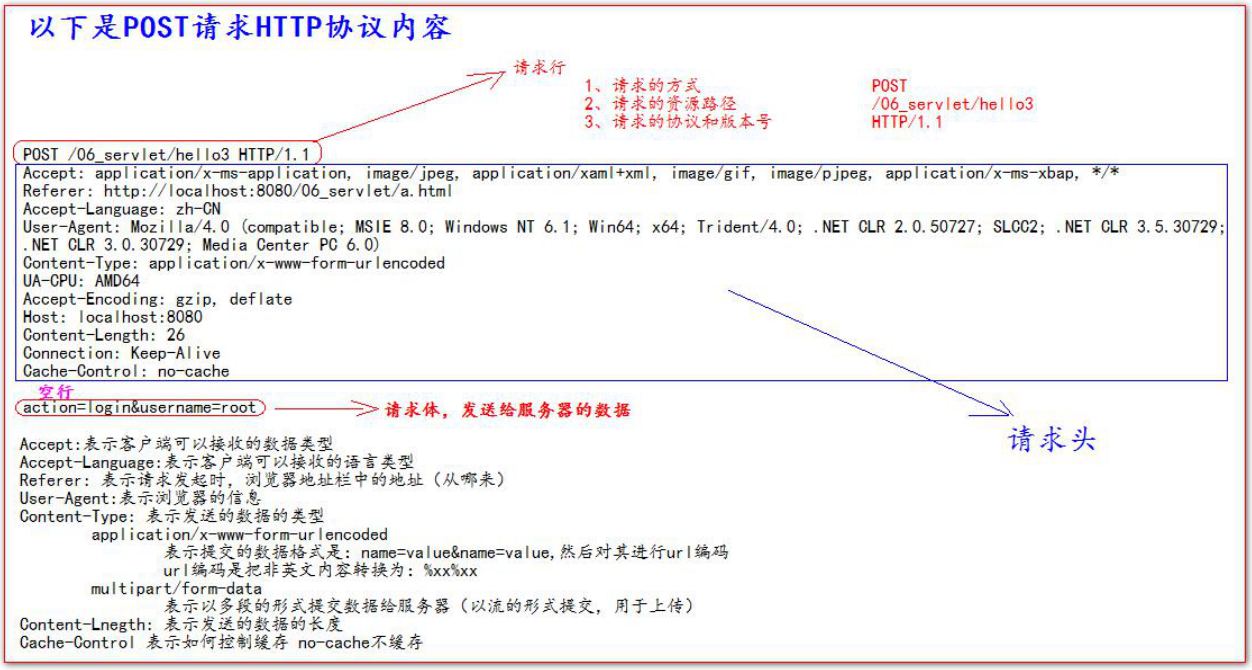
    (3) 请求的协议的版本号 HTTP/1.1

2. 请求头

    key : value 不同的请求头，有不同的含义

空行

3. 请求体 ===>>> 就是发送给服务器的数据



4.2.3 常用请求头的说明

Accept: 表示客户端可以接收的数据类型

Accpet-Languege: 表示客户端可以接收的语言类型

User-Agent: 表示客户端浏览器的信息

Host： 表示请求时的服务器 ip 和端口号

4.3 响应的 HTTP 协议格式

1. 响应行

   (1) 响应的协议和版本号

   (2) 响应状态码

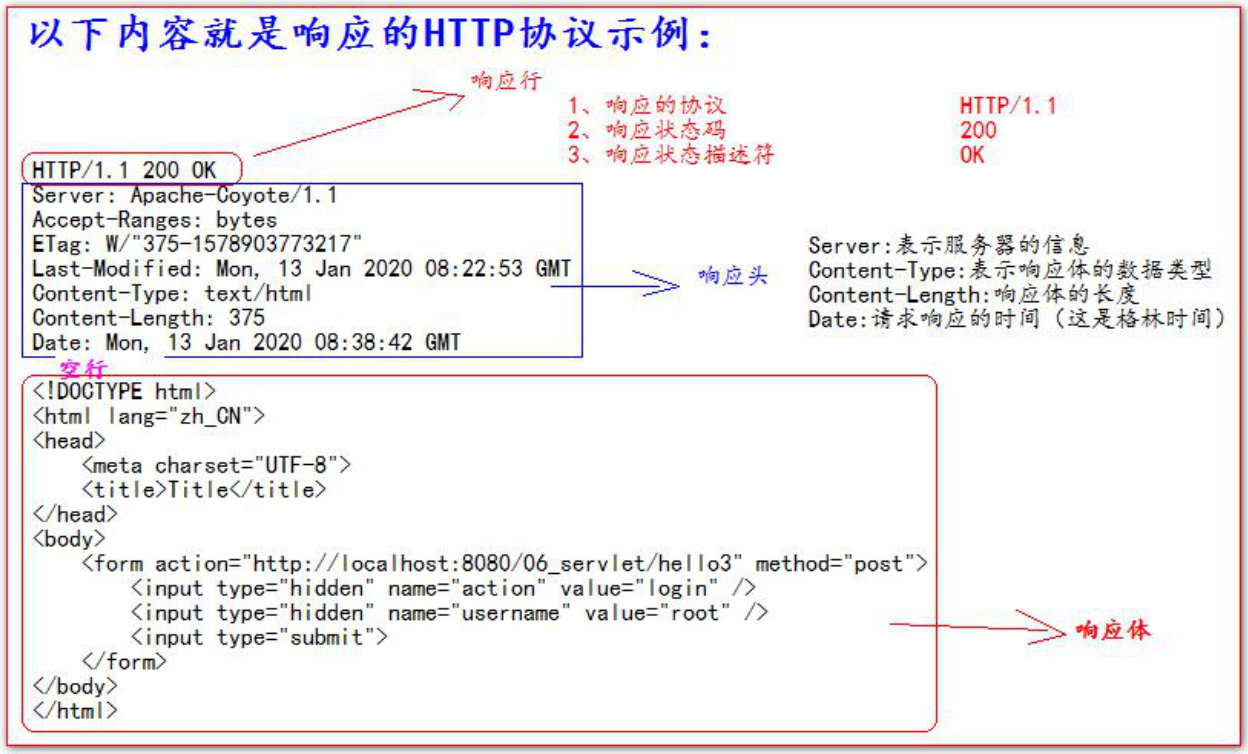
   (3) 响应状态描述符

2. 响应头

   (1) key : value 不同的响应头，有其不同含义

空行

3. 响应体 ---->>> 就是回传给客户端的数据



4.4 常用的响应码说明

200　　表示请求成功

302　表示请求重定向

404　　表示请求服务器已经收到了，但是你要的数据不存在（请求地址错误）

500　　表示服务器已经收到请求，但是服务器内部错误（代码错误）

4.5 MIME 类型说明

MIME 是 HTTP 协议中数据类型。

MIME 的英文全称是"Multipurpose Internet Mail Extensions" 多功能 Internet 邮件扩充服务。MIME 类型的格式是“大类型/小

类型”，并与某一种文件的扩展名相对应。

常见的 MIME 类型：

文件		MIME 类型
超文本标记语言文本	.html , .htm	text/html
普通文本	.txt	text/plain
RTF 文本	.rtf	application/rtf
GIF 图形	.gif	image/gif
JPEG 图形	.jpeg,.jpg	image/jpeg
au 声音文件	.au	audio/basic
MIDI 音乐文件	mid,.midi	audio/midi,audio/x-midi
RealAudio 音乐文件	.ra, .ram	audio/x-pn-realaudio
MPEG 文件	.mpg,.mpeg	video/mpeg
AVI 文件	.avi	video/x-msvideo
GZIP 文件	.gz	application/x-gzip
TAR 文件	.tar	application/x-tar

5. HttpServletRequest 类

每次只要有请求进入 Tomcat 服务器，Tomcat 服务器就会把请求过来的 HTTP 协议信息解析好封装到 Request 对象中。

然后传递到 service 方法（doGet 和 doPost）中给我们使用。我们可以通过 HttpServletRequest 对象，获取到所有请求的

信息。

5.1 HttpServletRequest 类的常用方法

getRequestURI()	获取请求的资源路径
getRequestURL()	获取请求的统一资源定位符（绝对路径）
getRemoteHost()	获取客户端的 ip 地址
getHeader()	获取请求头
getParameter()	获取请求的参数
getParameterValues()	获取请求的参数（多个值的时候使用）
getMethod()	获取请求的方式 GET 或 POST
setAttribute(key, value);	设置域数据
getAttribute(key);	获取域数据
getRequestDispatcher()	获取请求转发对象

5.2 如何获取请求参数

```
package com.admiral.servlet;

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

public class ParameterServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        System.out.println("username = " + username);
        System.out.println("password = " + password);

        String[] hobbies = request.getParameterValues("hobby");
```

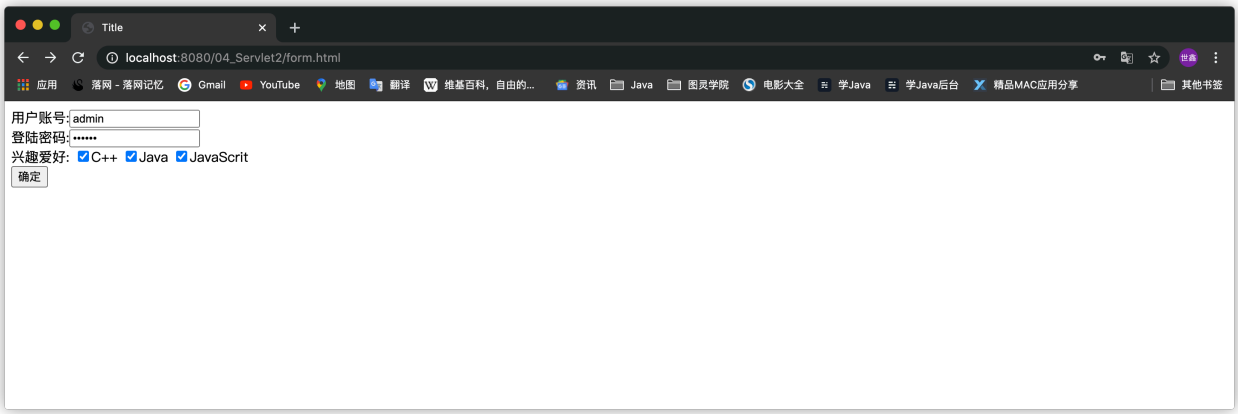
```
        System.out.println("hobby = " + Arrays.asList(hobbies));
    }
}
```

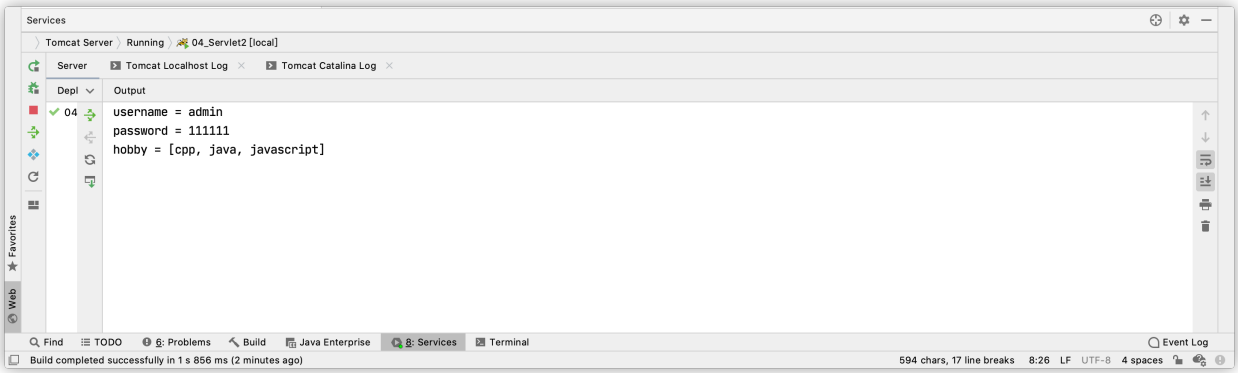
web.xml

```
<?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>RequestAPIServlet</servlet-name>
        <servlet-class>com.admiral.servlet.RequestAPIServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ParameterServlet</servlet-name>
        <servlet-class>com.admiral.servlet.ParameterServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>RequestAPIServlet</servlet-name>
        <url-pattern>/requestAPIServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ParameterServlet</servlet-name>
        <url-pattern>/parameterServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

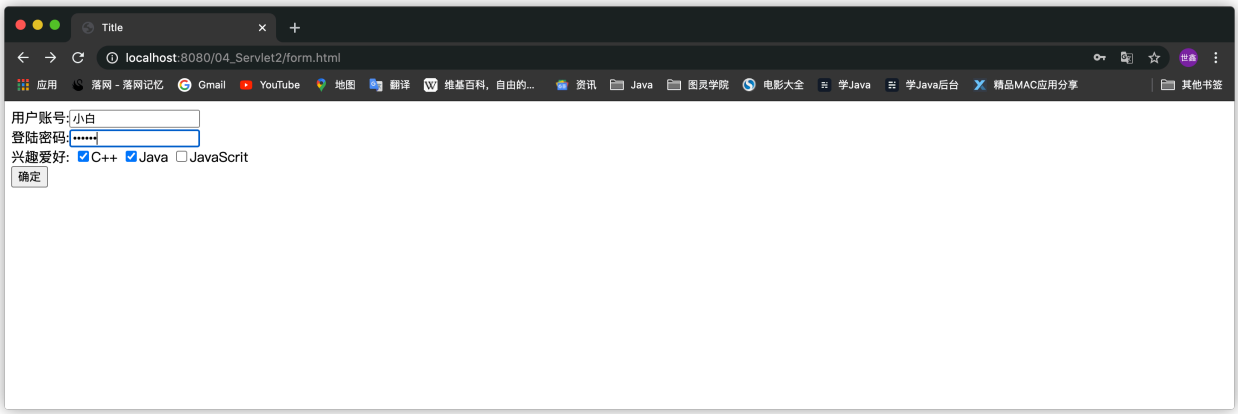
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form action="http://localhost:8080/04_Servlet2/parameterServlet"
method="get">
        用户账号:<input type="text" name="username"><br>
        登陆密码:<input type="password" name="password"><br>
        兴趣爱好:
            <input type="checkbox" name="hobby" value="cpp">C++
            <input type="checkbox" name="hobby" value="java">Java
            <input type="checkbox" name="hobby"
value="javascript">JavaScrit<br>
```

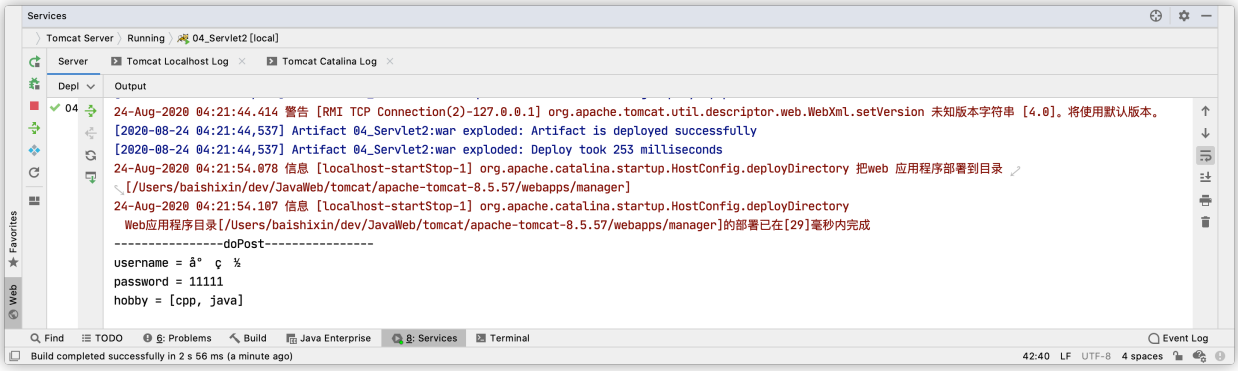
```
        <input type="submit" value="确定">
    </form>
</body>
</html>
```





5.3 POST 请求的中文乱码





设置 request 编码:

```
request.setCharacterEncoding("UTF-8");
```

> 注意,设置编码一定要在获取参数之前.否则会失效.

```
package com.admiral.servlet;

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

public class ParameterServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        System.out.println("----------------doPost----------------");
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        System.out.println("username = " + username);
        System.out.println("password = " + password);

        String[] hobbies = request.getParameterValues("hobby");
        System.out.println("hobby = " + Arrays.asList(hobbies));
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("----------------doGet----------------");
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        System.out.println("username = " + username);
        System.out.println("password = " + password);
```

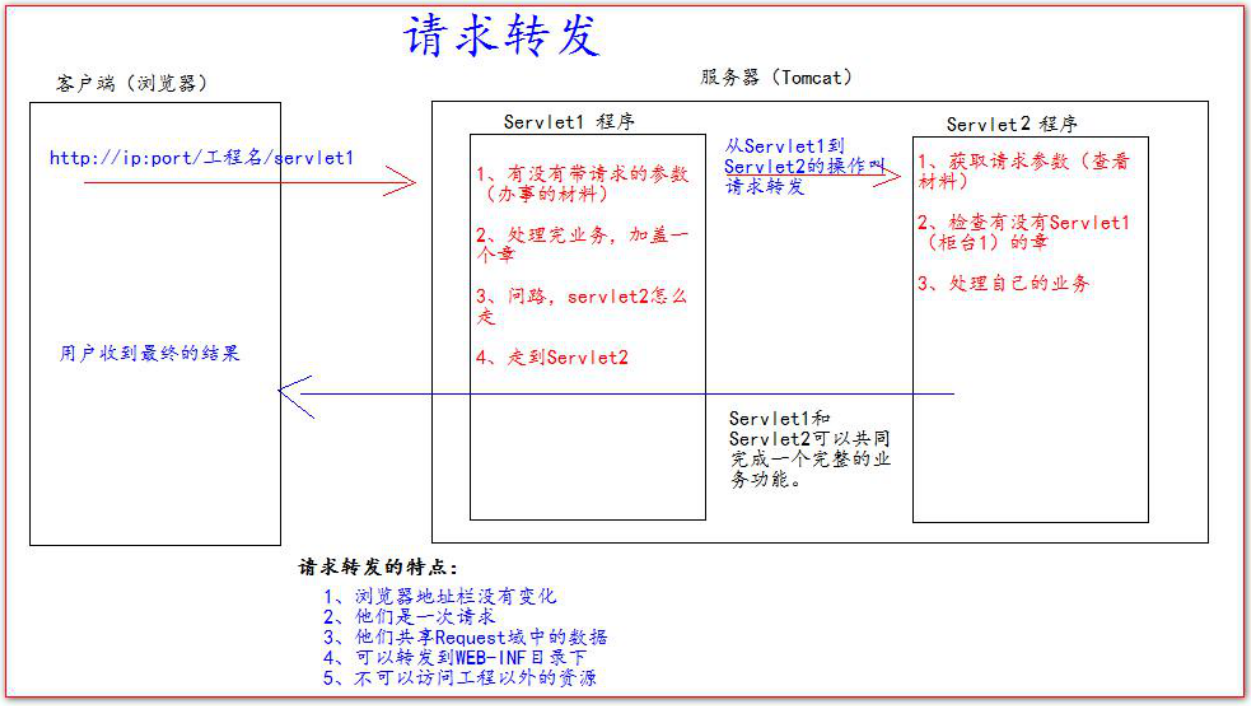
```
        String[] hobbies = request.getParameterValues("hobby");
        System.out.println("hobby = " + Arrays.asList(hobbies));
    }
}
```

5.4 GET请求的中文乱码

```
String username = req.getParameter("username");
//1 先以 iso8859-1 进行编码
//2 再以 utf-8 进行解码
username = new String(username.getBytes("iso-8859-1"), "UTF-8");
```

5.5 请求的转发

请求转发是指，服务器收到请求后，从一次资源跳转到另一个资源的操作叫请求转发。



Servlet1

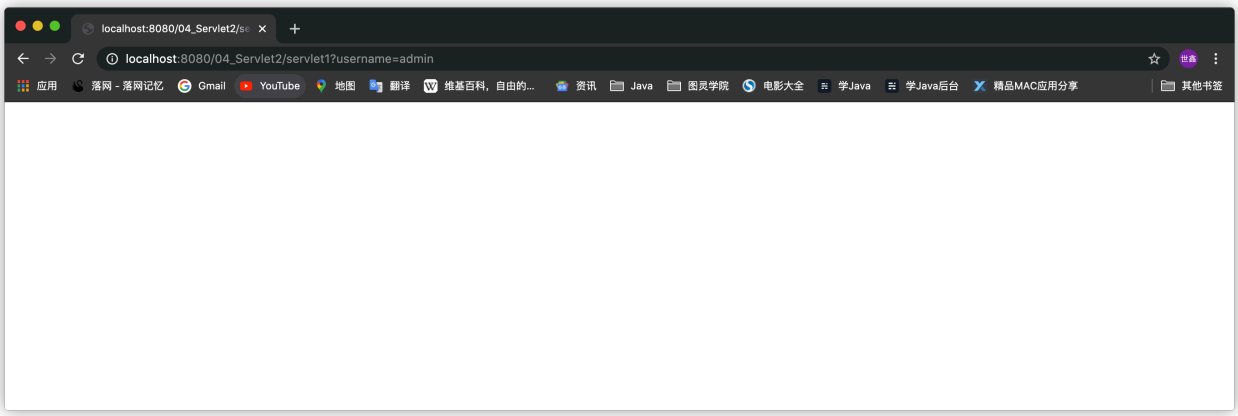
```
package com.admiral.servlet;

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

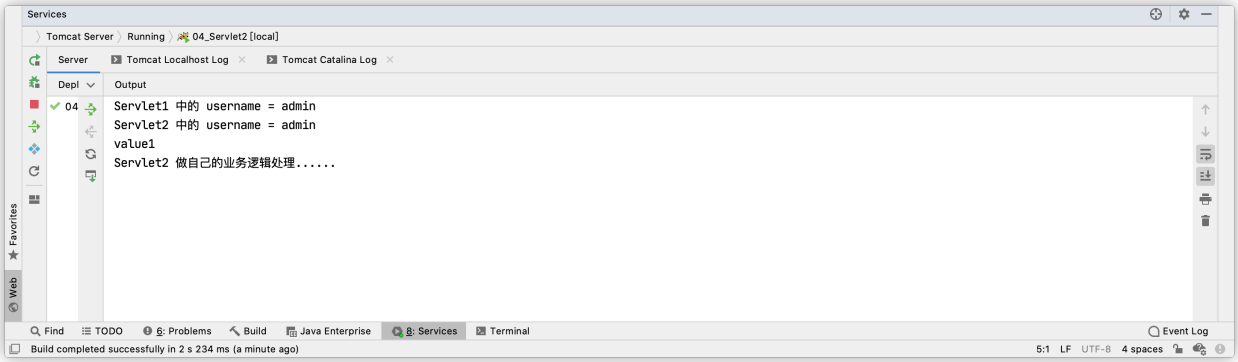
```
public class Servlet1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = request.getParameter("username");
        System.out.println("Servlet1 中的 username = " + username);
        request.setAttribute("key1","value1");

        RequestDispatcher requestDispatcher =
request.getRequestDispatcher("/servlet2");
        requestDispatcher.forward(request,response);

    }
}
```

```
package com.admiral.servlet;

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

public class Servlet2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = request.getParameter("username");
        System.out.println("Servlet2 中的 username = " + username);

        System.out.println(request.getAttribute("key1"));

        System.out.println("Servlet2 做自己的业务逻辑处理......");
    }
}
```

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>Servlet1</servlet-name>
        <servlet-class>com.admiral.servlet.Servlet1</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>Servlet2</servlet-name>
        <servlet-class>com.admiral.servlet.Servlet2</servlet-class>
    </servlet>


    <servlet-mapping>
        <servlet-name>Servlet1</servlet-name>
        <url-pattern>/servlet1</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>Servlet2</servlet-name>
        <url-pattern>/servlet2</url-pattern>
    </servlet-mapping>
</web-app>
```





6. HttpServletResponse 类

HttpServletResponse 类和 HttpServletRequest 类一样。每次请求进来，Tomcat 服务器都会创建一个 Response 对象传

递给 Servlet 程序去使用。HttpServletRequest 表示请求过来的信息，HttpServletResponse 表示所有响应的信息，

我们如果需要设置返回给客户端的信息，都可以通过 HttpServletResponse 对象来进行设置

6.1 两个输出流

字节流

getOutputStream();

常用于下载（传递二进制数据）

字符流

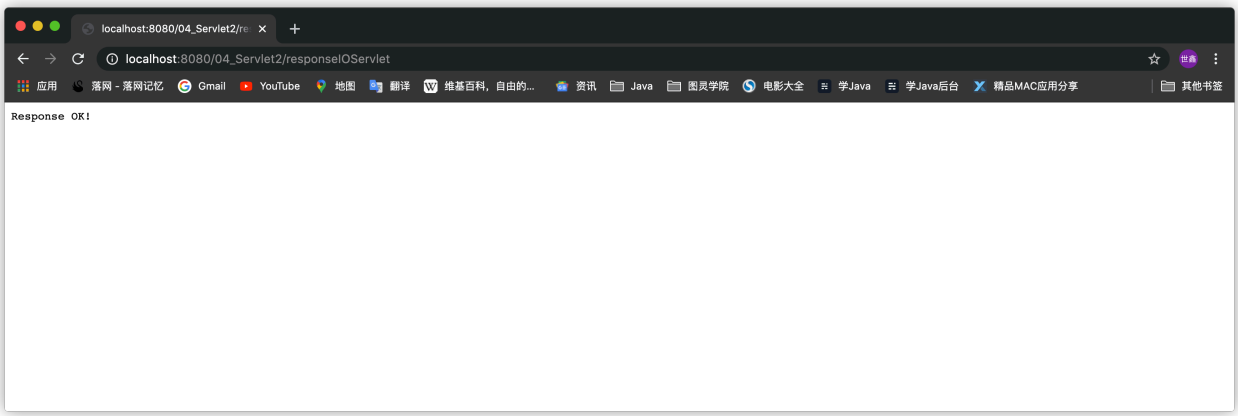
getWriter();

常用于回传字符串（常用）

```
package com.admiral.servlet;

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

```java
import java.io.PrintWriter;

public class ResponseIOServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        PrintWriter writer = response.getWriter();
        writer.write("Response OK!");
    }
}
```

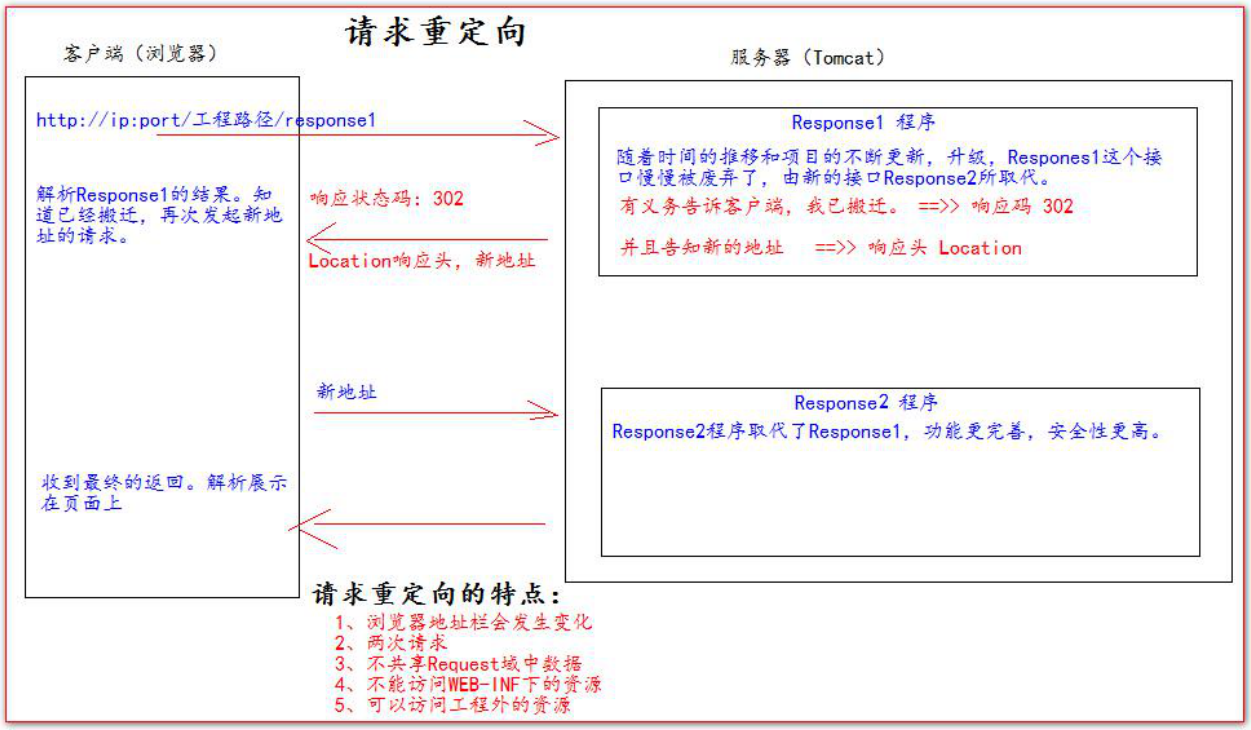
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>ResponseIOServlet</servlet-name>
        <servlet-class>com.admiral.servlet.ResponseIOServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ResponseIOServlet</servlet-name>
        <url-pattern>/responseIOServlet</url-pattern>
    </servlet-mapping>
</web-app>
```



6.2 响应的乱码解决

```java
package com.admiral.servlet;

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

public class ResponseIOServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //方式一
//        response.setCharacterEncoding("UTF-8");
//
//        response.setHeader("Content-Type","text/html; charset=UTF-8");

        //方式二：一定要在获取流之前设置
        response.setContentType("text/html; charset=UTF-8");

        PrintWriter writer = response.getWriter();
        writer.write("小白很帅!");
//        writer.write("Response OK!");
    }
}
```

6.3 请求重定向




```
// 请求重定向的第一种方案：
// 设置响应状态码 302 ，表示重定向，（已搬迁）
resp.setStatus(302);
// 设置响应头，说明 新的地址在哪里 resp.setHeader("Location",
"http://localhost:8080");



//请求重定向的第二种方案（推荐使用）：
resp.sendRedirect("http://localhost:8080");
```