* 学习目标

  * 能够掌握网络编程案例之编写简易版tomcat

    * ServetSocket

      * 文件路径：怎么找到

      * 响应：写协议和内容

  * 能够理解Tomcat的概述

    * J2EE定义规范：JAVAEE8 --tomcat8

    * Servlet,JSP,EL,WebSocket,...

  * 能够安装和使用Tomcat

    * tomcat 8.0

  * 能够在idea上创建和运行Web项目

  * 能够导出war部署web项目到tomcat上

  * 能够理解Http协议的概述

  * 能够理解URL和URI的概述

  * 能够掌握使用chrome分析请求和响应

  * 能够掌握Http常见的协议

  * 能够理解Servlet的概述

  * 能够掌握Servlet的HelloWorld开发

  * 能够掌握Servlet的执行流程

  * 能够掌握Servlet的生命周期

      * 构造器-->init -->service(多次)-->destroy

  * 能够掌握Servlet创建方式

      * 创建Servlet方式：

          * 实现Servlet 接口 -- 配置xml

          * 继承GenericServlet--配置xml

          * 继承HttpServlet---配置xml

---

* 回顾
  * 网络编程
  * 三要素
    * 协议：OSI 七层协议，TCP/IP 五层，四层协议
      * 物理层，数据链路层，网络层，传输层，会话层，表现层，应用层
      * 物理层，数据链路层，网络层，传输层，应用层
      * 网络接口层，网络层，传输层，应用层
    * IP
      * IPv4
      * IPv6
      * ping，ipconfig
    * 端口号
      * 0-65535:1024开始
  * Socket,ServerSocket

* 能够掌握网络编程案例之编写简易版tomcat

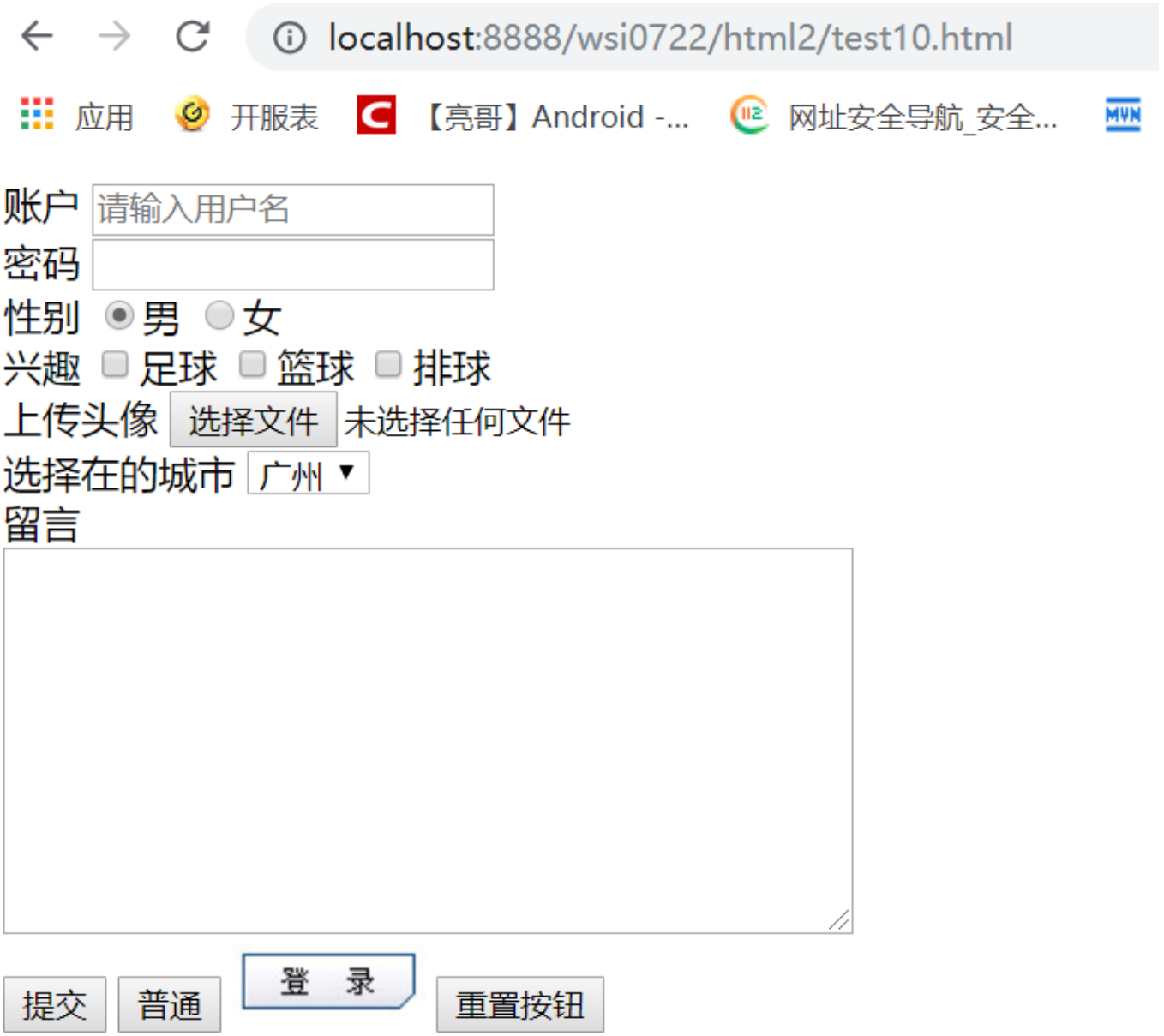
```
案例一：获取浏览器访问服务器的请求信息
 服务器端
public class WebServer {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket=new ServerSocket(8888);
        System.out.println("tomcat 服务器启动成功...");
        while(true){
            Socket server = serverSocket.accept();
            ThreadPoolUtils.getPool().execute(new Runnable() {
                @Override
                public void run() {
                    try {
                        InputStream is = server.getInputStream();
                        byte[] b=new byte[1024];
```

```
                        int len = is.read(b);
                        String msg=new String(b,0,len);
                        System.out.println(msg);
                        is.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    }
}

 * 复制之前html2的项，放在项目下
 * 访问：http://localhost:8888/wsi0722/html2/test1.html
 * 结果：
tomcat 服务器启动成功...
GET /wsi0722/html2/test1.html HTTP/1.1
Host: localhost:8888
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
Sec-Fetch-User: ?1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en-AU;q=0.8,en;q=0.7
Cookie: Hm_lvt_aa5c701f4f646931bf78b6f40b234ef5=1554828318,1554945605; Webstorm

* 案例二：服务器
public class WebServer {
    public static void main(String[] args) throws IOException {
        ServerSocket serverSocket=new ServerSocket(8888);
        System.out.println("tomcat 服务器启动成功...");
        while(true){
            Socket server = serverSocket.accept();
            ThreadPoolUtils.getPool().execute(new Runnable() {
                @Override
                public void run() {
```

```
                    try {
                        BufferedReader br=new BufferedReader(new InputStreamRea
                        String request = br.readLine();
                        String path=request.split(" ")[1];
                        System.out.println(path);
                        FileInputStream is=new FileInputStream(path);
                        OutputStream os = server.getOutputStream();
                        // 写入HTTP协议响应头,固定写法
                        os.write("HTTP/1.1 200 OK\r\n".getBytes());
                        os.write("Content-Type:text/html\r\n".getBytes());
                        // 必须要写入空行,否则浏览器不解析
                        os.write("\r\n".getBytes());
                        byte[] buffer=new byte[1024];
                        int len=-1;
                        while((len=is.read(buffer))!=-1){
                            os.write(buffer,0,len);
                        }
                        os.flush();
                        // 释放资源
                     is.close();
                     os.close();
                     br.close();
                     server.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    }
}
* 复制之前html2的项，放在项目下
* 访问：http://localhost:8888/wsi0722/html2/test10.html
* 结果可以下浏览器上看到效果
```



* 能够理解Tomcat的概述

  * ApacheTomcat软件是Java Servlet、JavaServer页面、Java表达式语言和JavaWebSocket技术的开源实现。

  * 规范是有J2EE(Java 2 Platform Enterprise Edition)定义的

* 能够安装和使用Tomcat

  * 参考:tomcat安装和使用

* 能够在idea上创建和运行Web项目

  * 参考：01-xidea创建web项目

* 能够导出war部署web项目到tomcat上

  * 在idea上到出war

    * 参考：01-在idea上导出war包

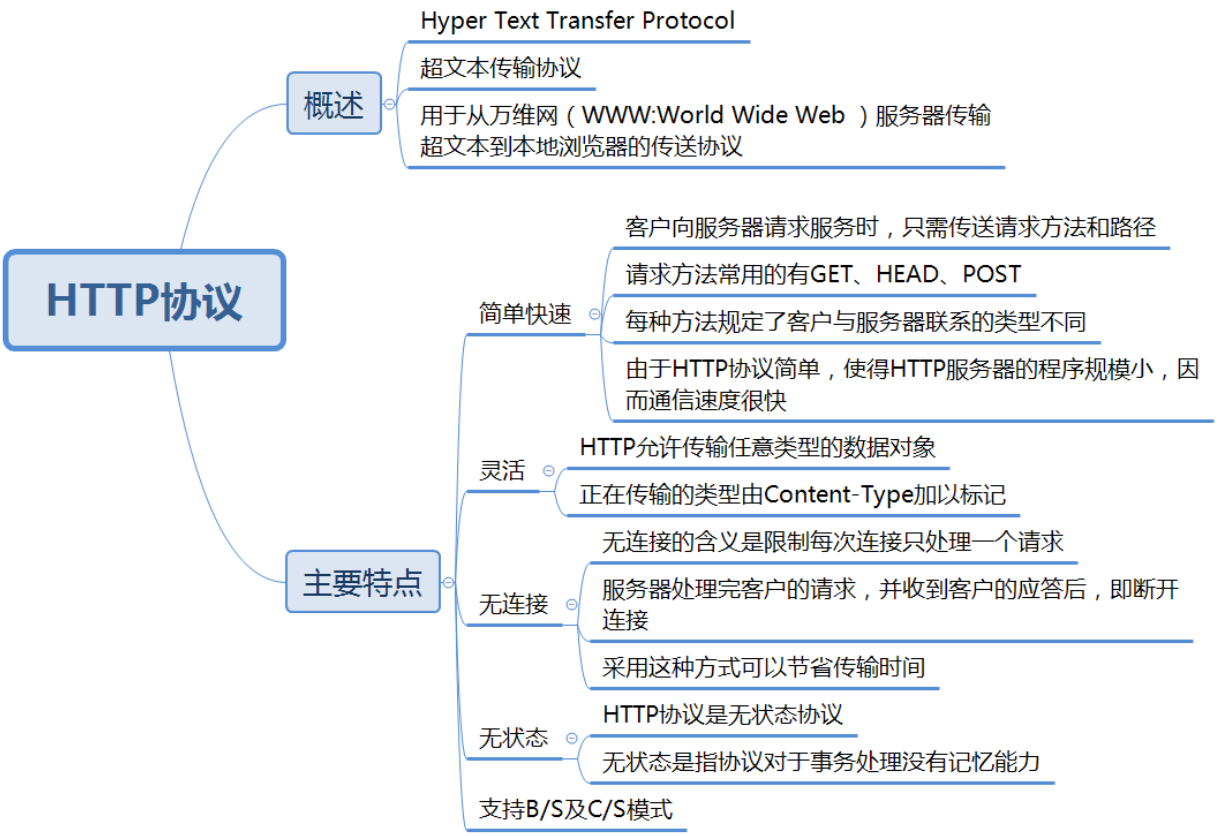
  * 手动部署war到tomcat

    * 参考:01-手动部署war到tomcat

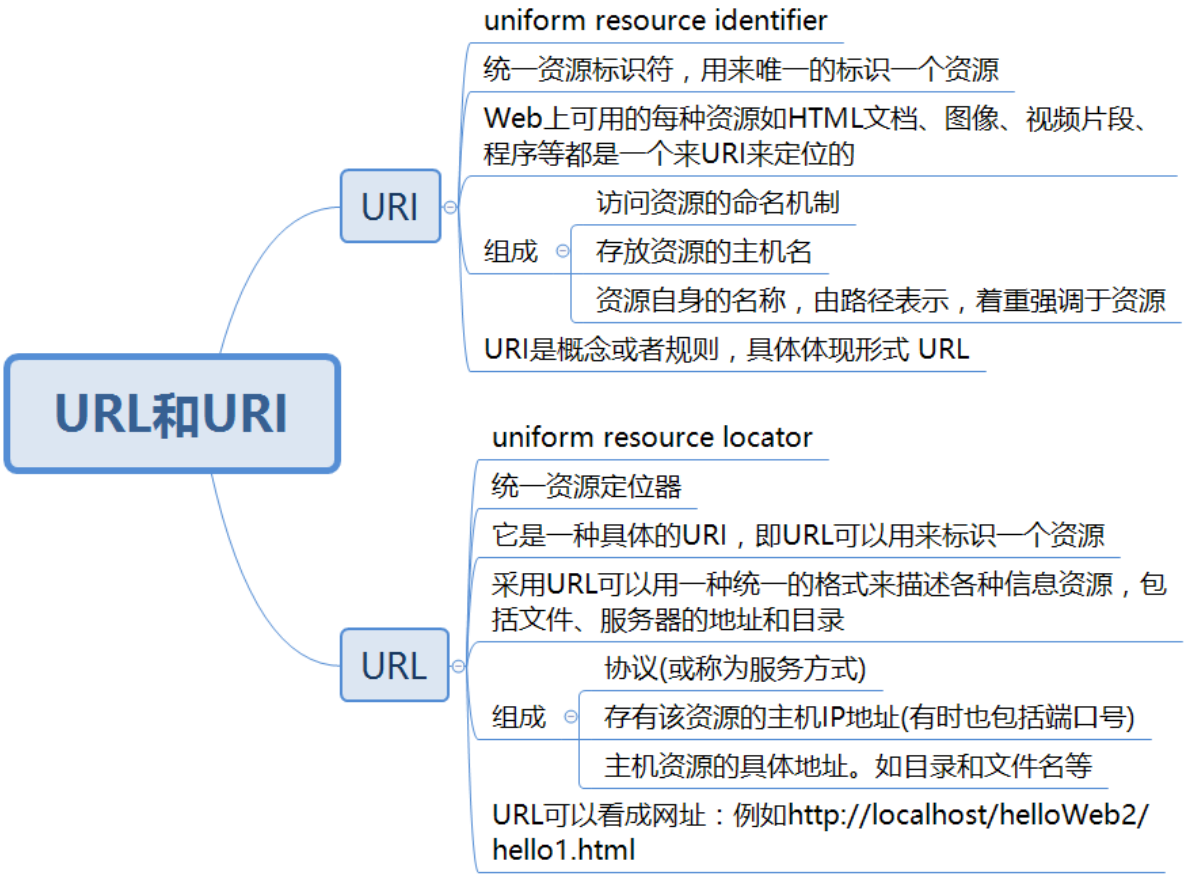
  * 配置默认端口，默认应用，默认主页 ，域名

    *参考: 01-默认端口，默认应用，默认主页,域名

* 能够理解Http协议的概述



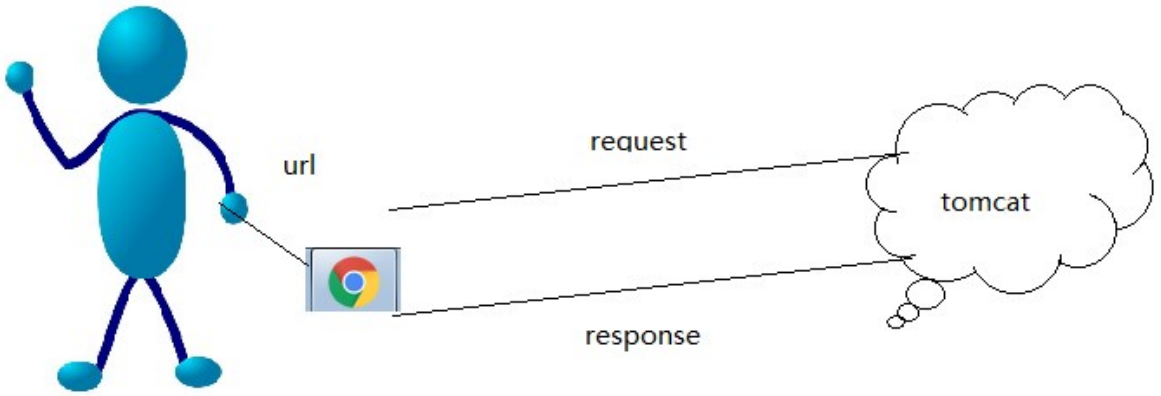
* 能够理解URL和URI的概述



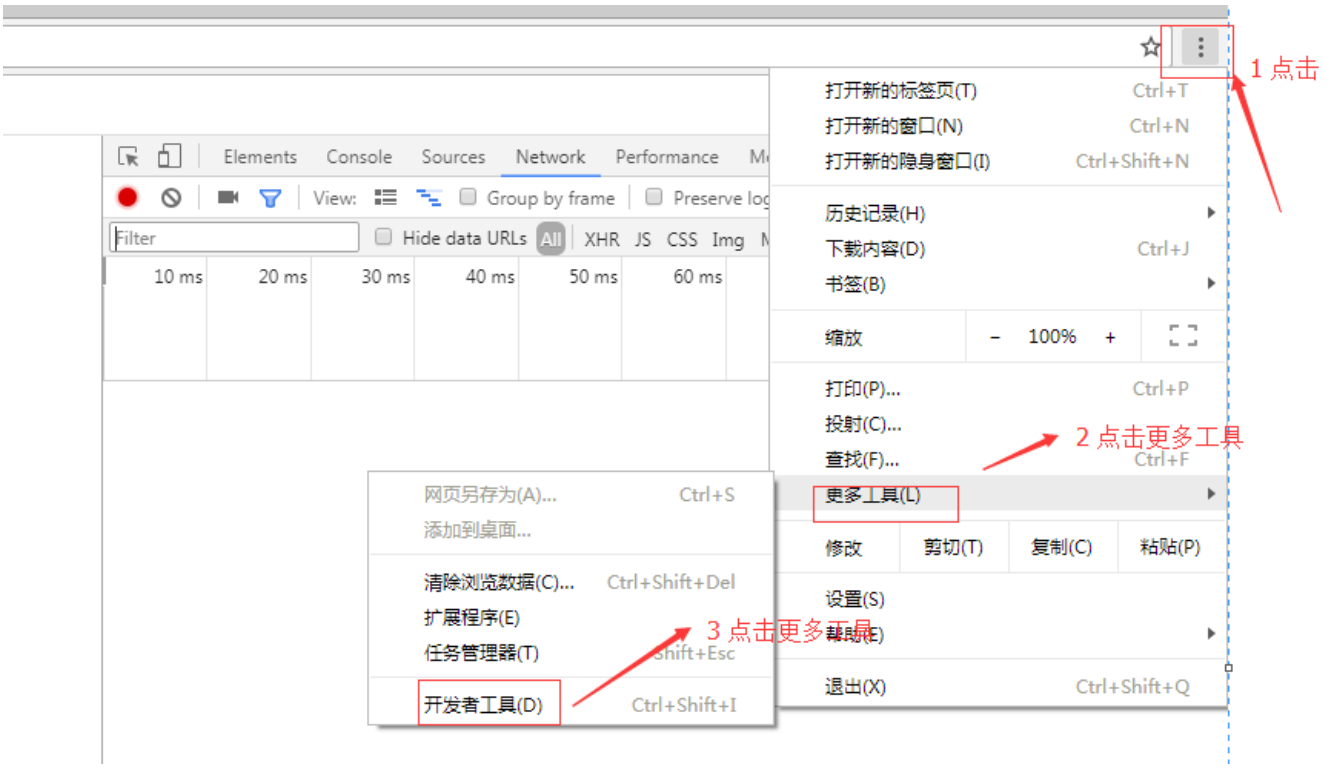
* URL的图解

* 能够掌握使用chrome分析请求和响应

  * HTTP的请求（Request）和HTTP的响应（Response）



* 打开chrome的开发者工具



Headers Preview Response Timing

▼ General

Request URL: http://localhost:8080/helloWeb2/hello2.html
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:8080
Referrer Policy: no-referrer-when-downgrade

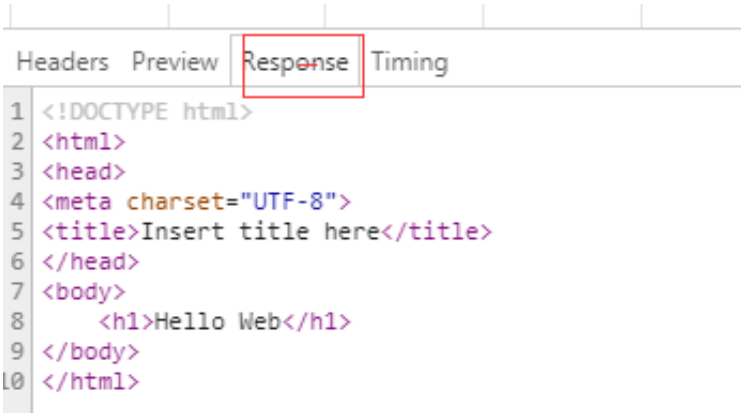
▼ Response Headers view source

Accept-Ranges: bytes
Content-Length: 145
Content-Type: text/html
Date: Thu, 22 Mar 2018 08:39:17 GMT
ETag: W/"145-1521680532000"
Last-Modified: Thu, 22 Mar 2018 01:02:12 GMT
Server: Apache-Coyote/1.1

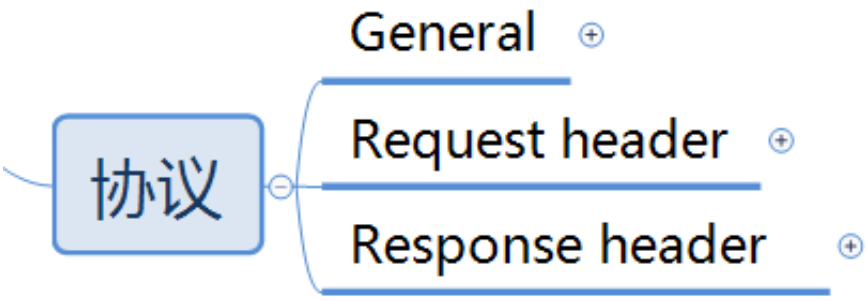
▼ Request Headers view source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Connection: keep-alive
Host: localhost:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36

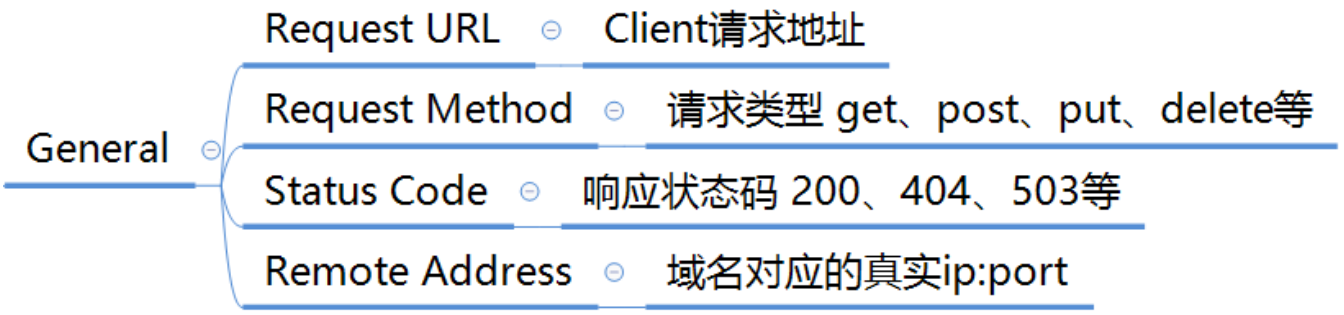
* Response页签



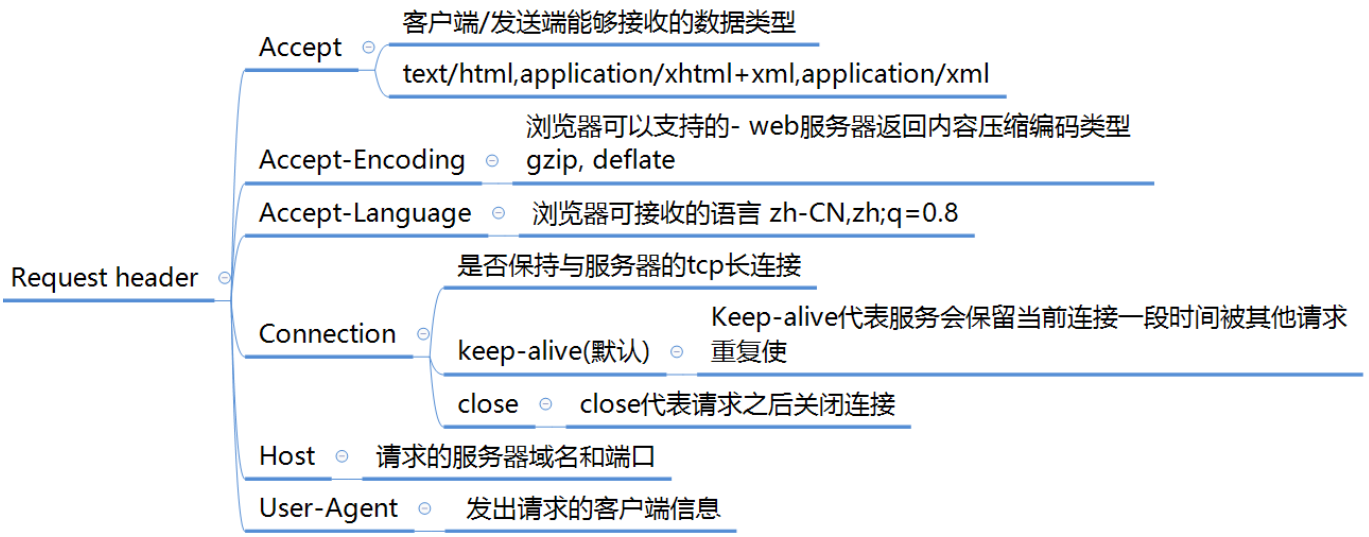
* 能够掌握Http常见的协议



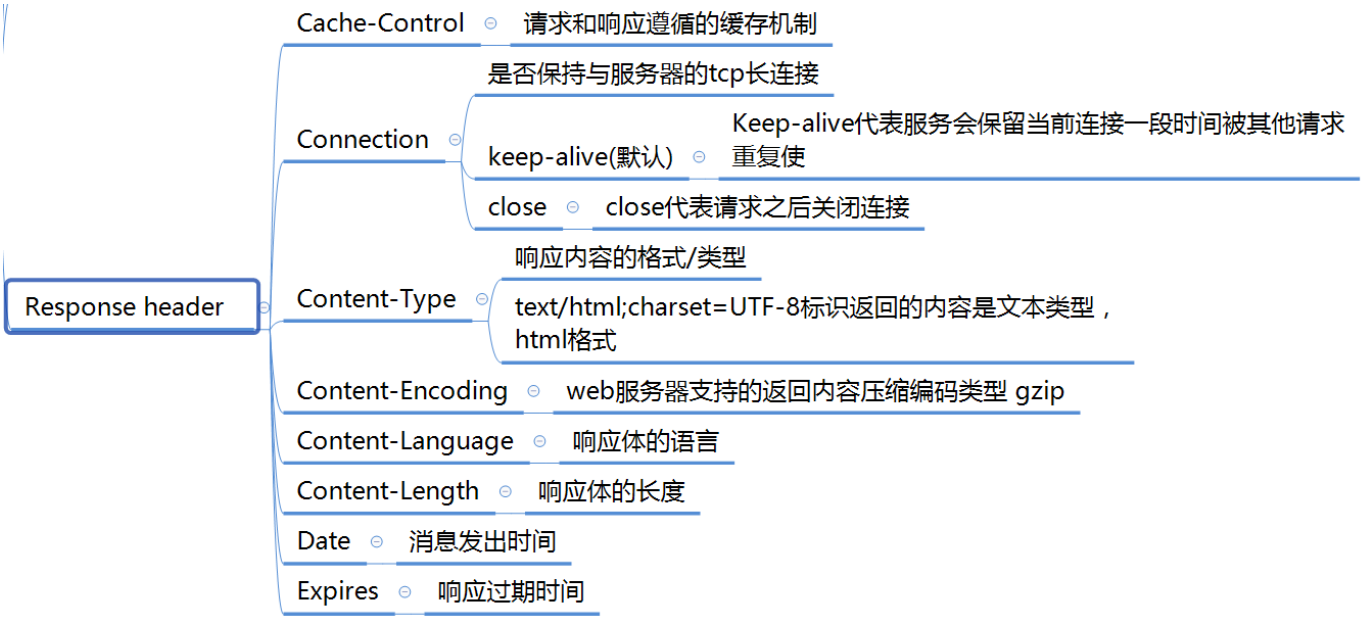
* General



* Request Header



* Response Header



* 能够理解Servlet的概述

* servlet 是运行在 Web 服务器中的小型 Java 程序（即：服务器端的小应用程序）。servlet 通常通过 HTTP（超文本传输协议）接收和响应来自 Web 客户端的请求。

* 能够掌握Servlet的HelloWorld开发

```
* 代码
public class TestServlet implements Servlet {
    @Override
```

```
    public void init(ServletConfig servletConfig) throws ServletException {

    }

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

    @Override
    public void service(ServletRequest servletRequest, ServletResponse servletR
        servletResponse.getWriter().write("HelloWorld");
    }

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

    @Override
    public void destroy() {

    }
}

* 在web.xml 配置
 <servlet>
        <servlet-name>test1</servlet-name>
        <servlet-class>com.lg.servlet.TestServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>test1</servlet-name>
        <url-pattern>/test1</url-pattern>
    </servlet-mapping>
```

* 能够掌握Servlet的执行流程

* 访问：http://localhost:8080/mweb/test1

* 通过/test1在web.xml中找到对应Servlet

```
    <servlet>
        <servlet-name>test1</servlet-name>
        <servlet-class>com.lg.servlet.TestServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>test1</servlet-name>
        <url-pattern>/test1</url-pattern>
    </servlet-mapping>
```

* Tomcat创建反射创建了TestServlet对象
* Tomcat创建完对象之后，调用了init方法
* 每次浏览器访问会调用service方法
* 当Servlet销毁的时候，调用destroy方法

* 能够掌握Servlet的生命周期
  * Serlvet的生命周期
    * 实例化--->init--->service--->destroy

```java
public class HelloServlet implements Servlet{

    public HelloServlet() {
        System.out.println("HelloServlet");
    }
    @Override
    public void init(ServletConfig config) throws ServletException {
        System.out.println("init");
    }

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

    @Override
```

```
17      public void service(ServletRequest req, ServletResponse res) throws Servlet
18          System.out.println("service");
19          //响应文本到浏览器
20          res.getWriter().write("hello Servlet");
21      }
22
23      @Override
24      public String getServletInfo() {
25          return null;
26      }
27
28      @Override
29      public void destroy() {
30          System.out.println("destroy");
31      }
32
33  }
34
```

```
信息: Server startup in 20841 ms
HelloServlet
init
service
```

* 停止tomcat的时候会调用destroy的方法

```
信息: ContextListener: contextDestroyed()
destroy
三月 29, 2018 10:02:29 上午 org.apache.coyote.AbstractProto
```

总结：第一次方法的时候，会调用Serlvet构造器--->init--->service

接下来方法的时候，只调用--->service

```
HelloServlet
init
service
service
service
service
service
service
service
service
service
service
service
service
service
service
service
```

* 可以看出：tomcat创建一个实例，单实例，放在内存

* 能够掌握Servlet创建方式

* 实现Servlet（参考TestServlet）

* 继承GenericServlet

* 模拟写一个适配器Servlet

```java
public abstract class MServlet implements Servlet{
    // 重写这个方法，只有一个不重写（service）
    @Override
    public void init(ServletConfig config) throws ServletException {

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

```
    @Override
    public void destroy() {

    }
}
public class HelloServlet1 extends MServlet{

    @Override
    public void service(ServletRequest req, ServletResponse res) throws Servlet
        res.getWriter().write("hello Servlet1");
    }

}
public class HelloServlet1 extends GenericServlet{

    @Override
    public void service(ServletRequest req, ServletResponse res) throws Servlet
        res.getWriter().write("hello Servlet1");
    }

}
```

配置文件：

```
<servlet>
        <servlet-name>hello1</servlet-name>
        <servlet-class>com.lg.servlet.HelloServlet1</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>hello1</servlet-name>
        <url-pattern>/hello1</url-pattern>
</servlet-mapping>
```

* 继承HttpServlet

* 分析HttpServlet的service方法：

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

    String method = req.getMethod();

    if (method.equals(METHOD_GET)) {
        long lastModified = getLastModified(req);
        if (lastModified == -1) {
            // servlet doesn't support if-modified-since, no reason
            // to go through further expensive logic
            doGet(req, resp);
        } else {
            long ifModifiedSince;
            try {
                ifModifiedSince = req.getDateHeader(HEADER_IFMODSINCE);
            } catch (IllegalArgumentException iae) {
                // Invalid date header - proceed as if none was set
                ifModifiedSince = -1;
            }
            if (ifModifiedSince < (lastModified / 1000 * 1000)) {
                // If the servlet mod time is later, call doGet()
                // Round down to the nearest second for a proper compare
```

假如是get形式，调用doGet的方法

```
} else if (method.equals(METHOD_HEAD)) {
    long lastModified = getLastModified(req);
    maybeSetLastModified(resp, lastModified);
    doHead(req, resp);

} else if (method.equals(METHOD_POST)) {
    doPost(req, resp);

} else if (method.equals(METHOD_PUT)) {
    doPut(req, resp);

} else if (method.equals(METHOD_DELETE)) {
    doDelete(req, resp);

} else if (method.equals(METHOD_OPTIONS)) {
    doOptions(req,resp);

} else if (method.equals(METHOD_TRACE)) {
    doTrace(req,resp);
```

假如是post，就调用doPost方法

```
1 public class HelloServlet2 extends HttpServlet{
2
3     @Override
```

```
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) thro
        super.doGet(req, resp);
        System.out.println("doGet");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) thr
        super.doPost(req, resp);
        System.out.println("doPost");
    }
}
配置:
<servlet>
        <servlet-name>hello2</servlet-name>
        <servlet-class>com.etc.servlet.HelloServlet2</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>hello2</servlet-name>
        <url-pattern>/hello2</url-pattern>
</servlet-mapping>

测试:
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="/5HelloServlet/hello2" method="post">
        <input type="submit" value="提交">
    </form>
</body>
</html>

* 温馨提醒：可以通过idea快速的创建Servlet
```