第 3 章 Java Web 基础

Java Web 是指所有应用于 Web 开发的 Java 技术的总称。XML 是由万维网联盟(W3C)创建的标记语言,用于定义、编码人类和机器可以读取的文档的语法。它非常适合万维网传输,提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。HTTP 是专门用于定义浏览器与服务器之间交互数据的过程以及数据本身的格式。对于 Web 开发人员来说,只有深入理解 HTTP,才能更好地开发、维护、管理 Web 应用。

通过本章的学习,您可以:

(1) 了解 XML 的概念;

(2) 掌握 XML 语法,学会定义 XML;

(3) 了解 HTTP 消息;

(4) 熟悉 HTTP 请求行和请求头字段的含义;

(5) 熟悉 HTTP 响应行和响应头字段的含义;

(6) 掌握在 Eclipse 中配置 Tomcat 服务器的方法。


视频讲解

3.1 XML 基 础

XML 是目前比较流行的、应用于不同应用程序之间的数据交换的一项技术。由于这种数据交换不以预先定义的一组数据结构为前提,因此,具有较强的可扩展性。

3.1.1 XML 文档简介

1. 什么是 XML

XML(eXtensible Makup Language,可扩展标记语言)中的可扩展指的是用户可以按照 XML 规则自定义标记。XML 文件就是存储该语言的文件。

在现实生活中,很多事物之间都存在着一定的关联关系。例如,一个班级有两名学生:学生 1,张强,男,20 岁;学生 2,李萌,女,20 岁,可以用层次结构图来表示。班级学生信息层次结构示意如图 3-1 所示。

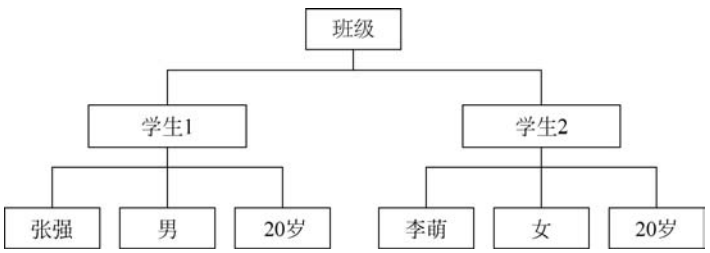


图 3-1 班级学生信息层次结构示意

图 3-1 直观地描述了班级与学生之间的层次结构关系；但是使用程序解析图片内容非常困难，这时，采用 XML 文件这种具有树状结构的数据格式就是最好的选择。用 XML 文档描述图 3-1 所示的关系，如文件 3-1 所示。

文件 3-1　student. xml

```
1   <?xml version = "1.0" encoding = "gb2312"?>
                                        <! -- 说明是 XML 文档,并指定 XML 文档的版本和编码 -- >
2      < class >                        <! -- 定义 XML 文档的根元素 -- >
3         < stu id = "001">             <! -- 定义 XML 文档的元素 -- >
4                   < name >张强</name >
5                   < sex >男</sex >
6                   < age > 20 </age >
7         </stu >                       <! -- 定义 XML 文档元素的结束标记 -- >
8         < stu id = "002">
9                 < name >李萌</name >
10                < sex >女</sex >
11                < age > 20 </age >
12        </stu >
13     </class >                        <! -- 定义 XML 文档根元素的结束标记 -- >
```

在文件 3-1 中，第 1 行代码是 XML 的文档声明，下面的< class >< stu >< name >< sex >< age >都是用户自己创建的标记。在 XML 中，它们被称为元素，其中 class 被称为整个文档的根元素，它有一个子元素 stu，在这个子元素中有一个属性 id，元素 stu 又有三个子元素 name、sex、age。元素必须成对出现，即包括开始标记和结束标记。例如，class 元素的开始标记是< class >，结束标记是</class >。

在 XML 文档中，通过元素的嵌套关系可以很准确地描述具有树状层次结构的复杂信息。因此，越来越多的应用程序都采用 XML 格式来存放相关的配置信息。在本书中，将介绍如何使用 web. xml 文件存放 Servlet 的配置信息，以及如何访问其他 XML 文件中的相关信息。

2. XML 与 HTML 的比较

XML 是一种与 HTML 相似的标记语言。它们都是标记文本，在结构上大致相同，都是以标记的形式来描述信息，但实际上它们又有着本质的区别。下面从 5 方面进行比较。

(1) 作用。HTML 的作用是用来显示数据；而 XML 的作用是为了传输和存储数据，其实现了数据的平台无关性。

(2) 是否区分大小写。HTML 不区分大小写；而 XML 是严格区分大小写的。

(3) 根元素的个数。HTML 可以有多个根元素；而格式良好的 XML 有且只能有一个根元素。

(4) 空格处理。HTML 中的空格是自动过滤的；而 XML 中的空格则不会自动删除。

(5) 标记的定义。HTML 中的标记是预定义的；而 XML 中的标记可以根据需要自己定义，并且可扩展。

3.1.2　XML 语法

一个基本的 XML 文档通常由文档声明和文档元素两部分组成。

1. 文档声明

在一个文本的 XML 文档中，必须包括一个 XML 的文档声明，用于说明这是一个 XML 的文档。XML 声明的语法格式如下：

```
1   <?xml version = "version" [encoding = "value"]  [standalone = "value"]?>
```

由以上格式可以看出,该声明包含在符号"<?"和"?>"之间,其中,xml 表明这是一个 XML 文档声明。

(1) version:用于指定遵循 XML 规范的版本号,最常用的版本是 1.0。在 XML 声明中必须包含 version 属性,该属性必须放在 XML 声明中的其他属性之前。

(2) encoding:关于指定 XML 文档中字符使用的编码集。常用的编码集为 GBK 或 GB2312(简体中文)、BIG5(繁体中文)、ISO88591(西欧字符)和 UTF-8(通用的国际编码)。

(3) standalone:英语指定该 XML 文档是否和一个外部文档嵌套使用。取值为 yes 或 no,若设置属性值为 yes,则说明是一个独立的 XML 文档与外部文件无关联;若设置属性值为 no,则说明 XML 文档不独立;默认取值为 no。

注意:

(1) 在声明中括号括起来的内容表示是可选的;

(2) "?"和"xml"之间不能留空格;

(3) 如果在 XML 文档中没有指定编码集,那么该 XML 文档将不支持中文。

2. 文档元素

XML 文档的主体部分是由元素组成的。XML 文档中的元素以树状分层结构排列,一个元素可以嵌套在另一个元素中。XML 文档中有且只有一个顶层元素,也称为根元素。其他所有元素都嵌套在根元素中。

XML 文档元素由开始标记、元素内容和结束标记 3 部分组成。定义 XML 文档元素的语法格式如下:

```
1    <tagName>content</tagName>
```

(1) <tagName>:XML 文档元素的开始标记。tagName 是元素的名称。

(2) content:元素内容,可以包含其他的元素、字符数据等。

(3) </tagName>:XML 文档元素的结束标记。tagName 是元素的名称,该名称必须与开始标志中指定的元素名称相同。

注意:

(1) XML 文档元素区分大小写;

(2) XML 文档有且只能有一个根元素;

(3) XML 文档元素必须同时拥有开始标志、结束标志,结束标志不能省略,这点与 HTML 不同;

(4) 空元素,没有内容的元素称为空元素。空元素可以不使用结束标志,但必须在开始标志的">"前加一个"/",例如:<img></img>可以简写成<img/>。

【例 3-1】 分析文件 3-1 中元素之间的关系。

解析: class、stu、name、sex、age 都是元素。其中,class 是根元素,stu 是 class 的子元素。name、sex、age 是 stu 的子元素。

3. 属性定义

在 XML 文档中,可以为元素定义属性,属性定义在元素的开始标记中,其值用单引号或双引号括起来。

【例 3-2】 在文件 3-1 中，为元素 stu 定义了属性 id 用于说明学生的 ID 号，代码如下：

```
1   <stu id = "001">
```

注意：在同一个元素的开始标记中属性名不能相同。

4. 注释

注释是为了便于阅读和理解，而在 XML 文档中添加的附加信息，比如作者姓名、地址、电话等信息，或者想暂时屏蔽某些 XML 语句。注释是对文档结构或内容的解释，不属于 XML 文档的内容，所以 XML 解析器不会处理注释内容。XML 文档的注释以“<!--”开始，以“-->”结束，与 HTML 写法基本一致。

如在文件 3-1 中的第 1～3 行都添加了注释。

3.1.3 XML 的应用

XML 应用于 Web 开发的许多方面，常用于简化数据的存储和共享。

1. XML 把数据从 HTML 文档分离

如果需要在 HTML 文档中显示动态数据，那么在数据改变时，将花费大量的时间来编辑 HTML。

通过 XML，数据能够存储在独立的 XML 文件中。这样程序员就可以专注于使用 HTML 进行布局和显示，并确保修改底层数据不再需要对 HTML 进行任何的改变。

通过使用几行 JavaScript，就可以读取一个外部 XML 文件，然后更新 HTML 中的数据内容。

2. XML 简化数据共享

在真实的世界中，计算机系统和数据使用不兼容的格式来存储数据。

XML 数据以纯文本格式进行存储，因此提供了一种独立于软件和硬件的数据存储方法。这让创建不同应用程序可以共享的数据变得更加容易。

3. XML 简化数据传输

通过 XML，可以在不兼容的系统之间轻松地交换数据。

对开发人员来说，其中一项最费时的挑战是在互联网上的不兼容系统之间交换数据。由于可以通过各种不兼容的应用程序来读取数据，以 XML 来交换数据便降低了其复杂性。

4. XML 简化平台的变更

升级到新的系统（硬件或软件平台）总是非常费时的。若要转换大量的数据，不兼容的数据就会丢失。

XML 数据以文本格式存储。这使得 XML 在不损失数据的情况下，更容易扩展或升级到新的操作系统、新应用程序或新的浏览器。

3.2 HTTP 协 议

3.2.1 HTTP 概述

1. HTTP 简介

HTTP（Hyper Text Transfer Protocol，超文本传输协议）是用于从万维网（World Wide Web，WWW）服务器传输超文本到本地浏览器的传输协议。

HTTP 工作于客户端—服务器端架构之上。浏览器作为 HTTP 客户端通过 URL 向 HTTP 服务器端即 Web 服务器发送所有请求；Web 服务器根据接收到的请求，向客户端回送响应信息，如图 3-2 所示。

图 3-2　客户端与服务器端的交互过程

HTTP 有如下特点。

(1) 支持 B/S 及 C/S 结构。

(2) 简单快速：客户端向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有 GET、HEAD、POST 三种。每种方法规定了客户与服务器联系的类型不同。由于 HTTP 简单，使得 HTTP 服务器的程序规模小，因而通信速度较快。

(3) 灵活：HTTP 允许传输任意类型的数据对象。正在传输的类型由 Content-Type 加以标记。

(4) 无状态：HTTP 是无状态协议。无状态是指协议对于事务处理没有记忆能力，如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。

(5) 无连接：无连接的含义是限制每次连接只处理一个请求。服务器端处理完客户端的请求，并收到客户端的应答后，即断开连接。采用这种方式可以节省传输时间。

2. HTTP 发展阶段

HTTP 于 1990 年提出，经过几年的使用与发展，得到不断的完善和扩展，最初的版本是 0.9，目前已经不使用了；1996 年 5 月，HTTP 1.0 版本发布；1997 年 1 月，HTTP 1.1 版本发布；HTTP/2 标准于 2015 年 5 月以 RFC 7540 正式发布，取代 HTTP 1.1 成为 HTTP 的实现标准。下面着重介绍目前流行的 HTTP 1.0 和 HTTP 1.1 版本。

(1) HTTP 1.0。1996 年 5 月，HTTP 1.0 版本发布，任何格式的内容都可以发送。这使得互联网不仅可以传输文字，还能传输图像、视频、二进制文件。这为互联网的发展奠定了基础。

HTTP 1.0 请求/响应的交互过程如图 3-3 所示。

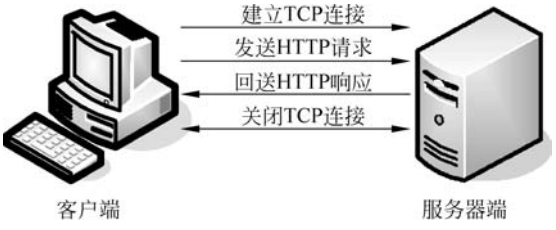


图 3-3　HTTP 1.0 请求/响应的交互过程

由图 3-3 可以看出，HTTP 1.0 版的每个 TCP 连接都只能发送一个请求。发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接。

TCP 连接的新建成本很高，随着网页加载的外部资源越来越多，这个问题就越发突出了。

【例 3-3】 分析 HTML 代码的请求响应交互过程。代码如文件 3-2 所示。

文件 3-2　http1.html

```
1   <html>
2      <body>
3            <img src="/image01.jpg">
4            <img src="/image02.jpg">
```

```
5              < img src = "/image03.jpg">
6      </body >
7   </html >
```

文件 3-2 中包含 3 个< img >标记,< img >标记的 src 属性是图片的 URL 地址。因此,当客户端访问这些图片时,需要发送 3 次请求,每次请求都要与服务器端建立连接。如此一来,必然导致交互延时,影响访问速度。

(2) HTTP 1.1。1997 年 1 月,HTTP 1.1 版本发布,只比 1.0 版本晚了半年。它进一步完善了 HTTP 协议,一直沿用至今,目前还是最流行的版本。

HTTP 1.1 版的最大变化,就是引入了持久连接(persistent connection),即 TCP 连接默认不关闭,可以被多个请求复用,但是所有的数据通信都是按次序完成的,服务器只有处理完一个响应,才会处理下一个响应。HTTP 1.1 的交互过程如图 3-4 所示。

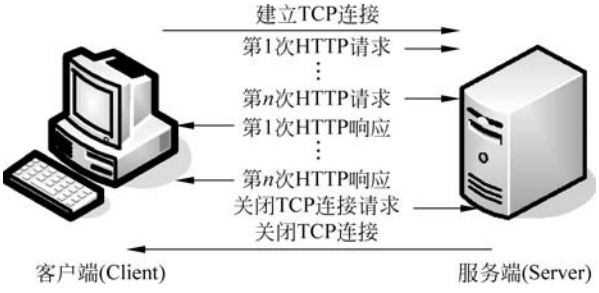


图 3-4　HTTP 1.1 的交互过程

HTTP 1.1 不仅继承了 HTTP 1.0 的优点,而且还有效解决了 HTTP 1.0 的性能问题,显著地减少了客户端与服务器端交互所需的时间。

3. 统一资源标识符

统一资源标识符(uniform resource identifier,URI)用来唯一地标识一个资源。Web 上可用的每种资源如 HTML 文档、图像、视频片段、程序等都是用一个 URI 来定位的。URI 有两个子集 URL 和 URN。

统一资源命名(uniform resource name,URN)是通过名字来标识资源。不指定访问方式。

统一资源定位器(uniform resource locator,URL)是一种具体的 URI,即 URL 可以用来标识一个资源,而且还指明了如何定位这个资源。

URI 是以一种抽象的、高层次概念定义统一资源标识,而 URL 和 URN 则是具体的资源标识的方式。URL 和 URN 都是一种 URI。URL、URN、URI 关系如图 3-5 所示。

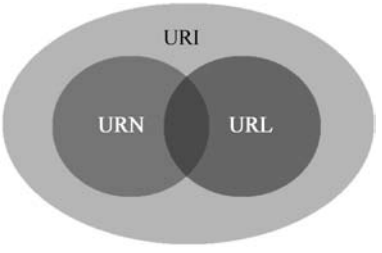


图 3-5　URL、URN、URI 关系图

【例 3-4】 举例说明 URI、URL 与 URN 的区别。

现实生活中通常用 ISBN 标识一本书,即 URI,有如下两种表示方式。

(1) ISBN: 9787302528999。它指定了一本书的 ISBN,可以唯一标识这本书,但是没有指定到哪里能找到这本书,这就相当于 URN。

(2) ISBN: 9787302528999,清华大学出版社。它指定了一本书的 ISBN,同时指明了出版社,这样读者就知道去哪里购买这本书了,这就相当于 URL。

在目前的互联网中,URN 用得非常少,几乎所有的 URI 都是 URL,一般的网页链接既可

以称为 URL,也可以称为 URI。

URL 语法格式如下:

```
scheme://hostname[:port]/website/[path/][file][?query][#fragment]
```

参数说明

(1) scheme:指定因特网服务的类型。最流行的类型是 HTTP,HTTPS。

(2) hostname:指定服务器的域名系统(DNS)主机名或 IP 地址。

(3) port:指定主机的端口号。端口号通常可以被省略,HTTP 服务器的默认端口号是 80。

(4) website:网站名称,可以是 Web 应用程序上下文、虚拟目录名、网站根目录等。

(5) path:指定远程服务器上的路径,该路径也可以被省略,省略该路径则默认被定位到网站的根目录。

(6) file:指定远程文档的名称。如果省略该文件名,通常会定位到 index.html、index.htm 等文件,或定位到 Web 服务器设置的其他文件。

(7) query:查询参数,以"?"开始,允许有多个参数,参数与参数之间用"&"作为分隔符,每个参数都以"参数名=参数值"的形式给出。

(8) fragment:信息片段,以"#"开始,是一种网页锚点。

【例 3-5】 分析 URL 的构成。

浏览某平台"Java Web 应用开发"课程的视频 URL 如下:

```
https://mooc1-1.chaoxing.com/nodedetailcontroller/visitnodedetail?courseId=204353013&
knowledgeId=168133932
```

该 URL 的协议部分为"https:",域名部分为"mooc1-1.chaoxing.com",虚拟目录是"/nodedetailcontroller/",文件名是"visitnodedetail",参数部分"courseId=204353013&knowledgeId=168133932"有两个参数 courseId 和 knowledgeId。

4. HTTP 消息

当用户在浏览器中访问某个 URL 地址、单击网页的某个超链接或者提交网页上的 form 表单时,浏览器都会向服务器发送请求数据,即 HTTP 请求消息。服务器端接收到请求数据后,会将处理后的数据返回给客户端,即 HTTP 响应消息。HTTP 请求消息和 HTTP 响应消息,统称为 HTTP 消息。

在 HTTP 消息中,除了服务器端的响应实体内容(如 HTML 网页、图片等)外,其他消息对用户都是不可见的,想要观察这些隐藏的消息,需要借助浏览器的开发者工具。

这里,使用 Chrome 浏览器自带的开发者工具查看 HTTP 头信息,步骤如下所述。

(1) 打开 Chrome 浏览器,在网页任意地方右击选择"检查"或者按下 Shift+Ctrl+I 组合键或者按 F12 键,打开 Chrome 自带的调试工具,会在浏览器的右部或底部显示开发者工具窗口。Chrome 浏览器的开发者工具窗口如图 3-6 所示。

(2) 选择 Network 标签,刷新网页(在打开调试工具的情况下刷新),会弹出当前访问的所有资源信息列表面板。资源信息列表面板如图 3-7 所示。

(3) 在资源信息列表面板的 Name 列中选择资源 URI,如百度的 URI:www.baidu.com,单击该 URI,在窗口右侧显示选中资源的详情面板,在该详情面板中选择 Headers 标签,就可以查看当前资源的 HTTP 头信息。当 HTTP 请求为 GET 方式时,在 Headers 标签页中有 General(基本信息)、Response Headers(响应头信息)、Request Headers(请求头信息)三部分。Chrome 浏览器中 HTTP 头信息(GET)如图 3-8 所示。

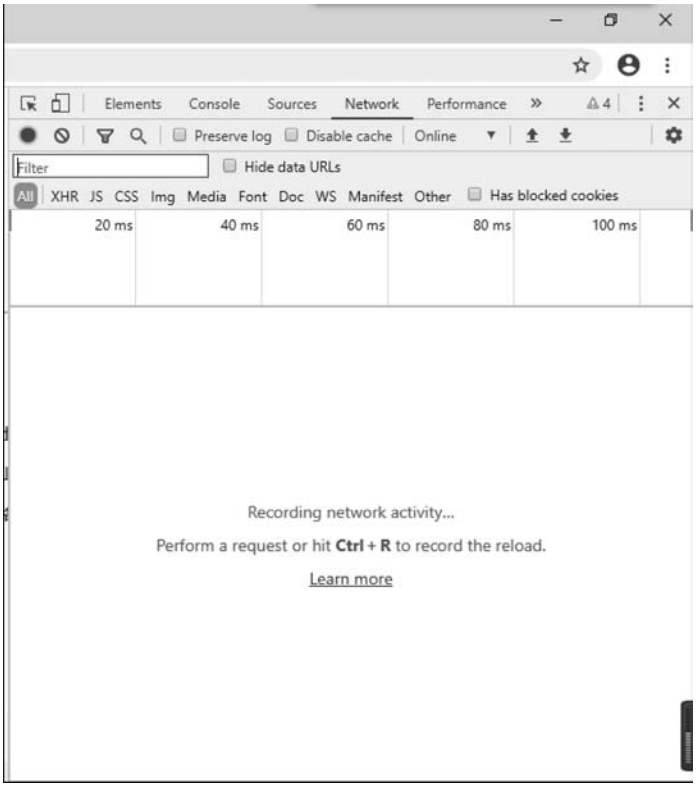


图 3-6　Chrome 浏览器的开发者工具窗口

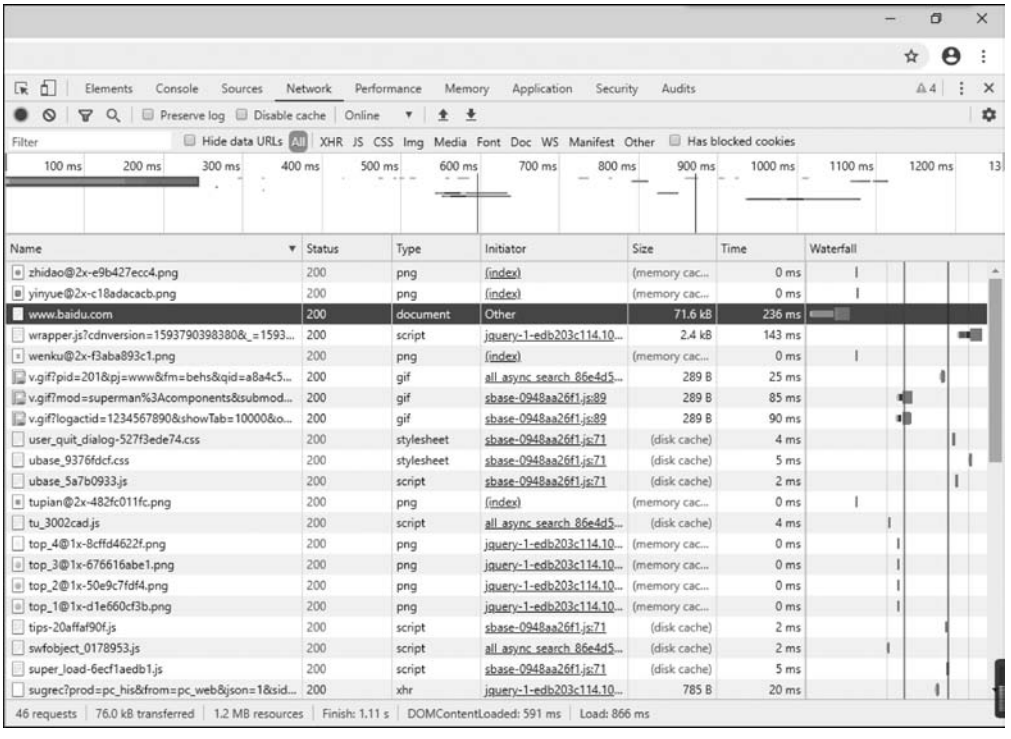


图 3-7　资源信息列表面板

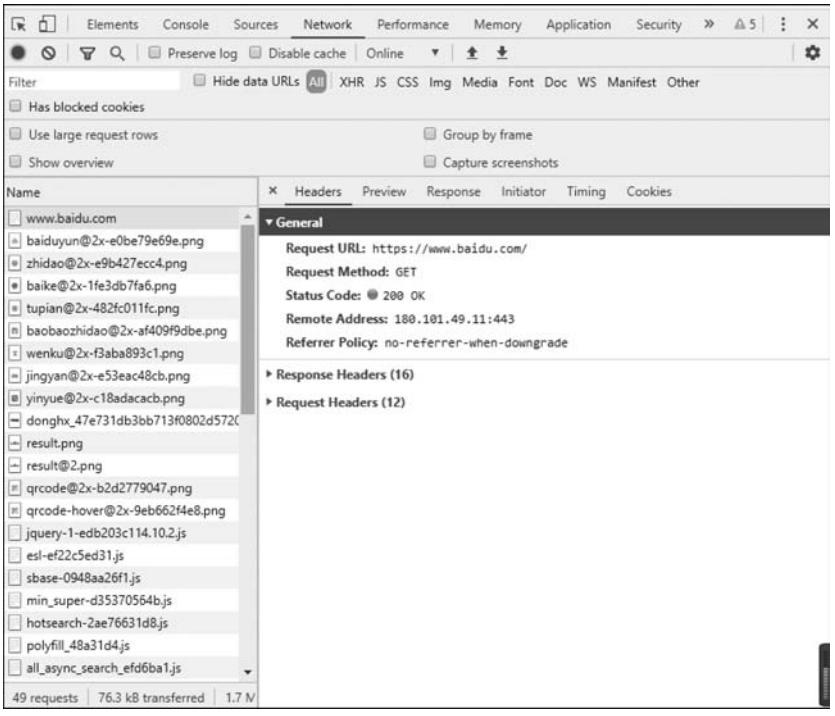


图 3-8　Chrome 浏览器中 HTTP 头信息(GET)

当 HTTP 请求为 POST 方式时,会增加 Form Data(表单数据)部分;当 HTTP 请求带参数时,会增加 Query String Parameters(请求参数)部分。单击前面的黑色三角可以折叠和展开每个部分的详情。Chrome 浏览器中 HTTP 头信息(POST)如图 3-9 所示。

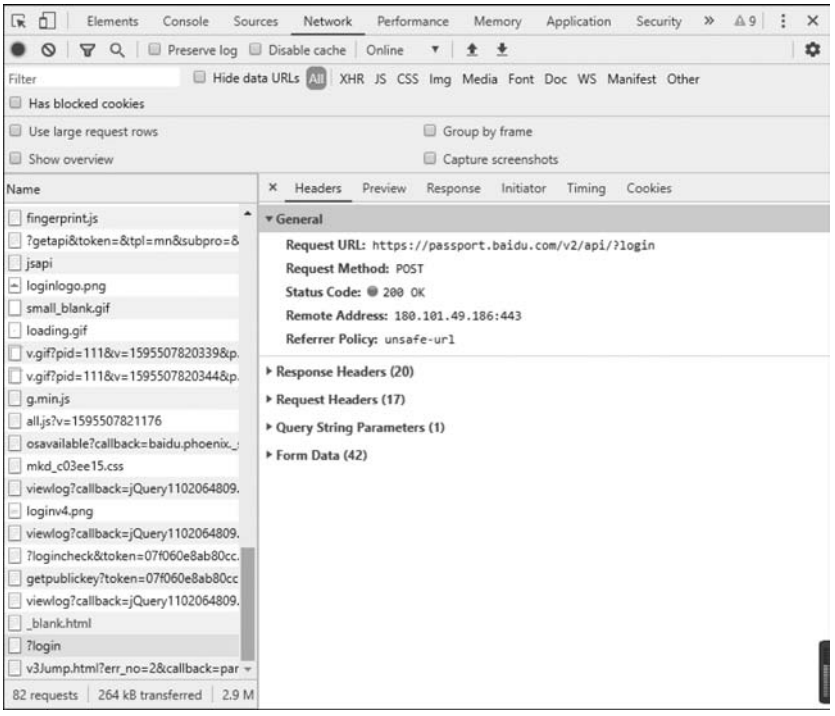


图 3-9　Chrome 浏览器中 HTTP 头信息(POST)

(4) 展开 Response Headers 详情,单击右侧的 view source,就可以查看原始的 HTTP 响应消息。其代码如下:

```
1   HTTP/1.1 200 OK
2   Bdpagetype: 1
3   Bdqid: 0xb121005d0018c9f4
4   Cache - Control: private
5   Connection: keep - alive
6   Content - Encoding: gzip
7   Content - Type: text/html;charset = UTF - 8
8   Date: Fri, 03 Jul 2020 15:31:46 GMT
9   Expires: Fri, 03 Jul 2020 15:30:46 GMT
10  Server: BWS/1.1
```

上述行代码中,第 1 行为响应状态行,其他行为响应头信息。

(5) 展开 Request Headers 详情,单击右侧的 view source,就可以查看原始的 HTTP 请求消息。其代码如下:

```
1   GET / HTTP/1.1
2   Host: www.baidu.com
3   Connection: keep - alive
4   Cache - Control: max - age = 0 Upgrade - In
5   secure - Requests: 1
6   User - Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/1.0.4044.138 Safari/537.36
7   Accept: text/html,application/xhtml + xml,application/xml
```

上述代码中,第 1 行为请求行,其他行为请求消息头。

关于 HTTP 请求和响应消息的其他内容在后续章节中再进行详细介绍。

大部分浏览器都自带有开发者工具,查看 HTTP 消息头的操作方法类似 Chrome 浏览器。

3.2.2 HTTP 请求消息

HTTP 请求消息由请求行、请求消息头和请求实体(请求数据)三部分构成,请求消息头与请求实体中间有一行空行,是告诉服务器请求消息头到此结束,接下来是请求正文。HTTP 请求消息结构如图 3-10 所示。

图 3-10 HTTP 请求消息结构

在 Chrome 浏览器的 Headers 标签中 Request Headers 部分,包含 HTTP 请求消息的请求行和请求消息头,而 Form Data(表单数据)部分则对应请求实体,是可选的。如图 3-8 所示的 HTTP 头信息没有请求实体,而图 3-9 中有请求实体。

1. HTTP 请求行

请求行由请求方法、URL 和协议版本 3 部分组成,各部分之间以空格分隔。

请求方法告诉服务器,你的具体操作是什么。URL 为请求资源对应的 URL 地址,它和消息头的 Host 属性组成完整的请求 URL,资源名以"/"开头,相对于连接的主机名。协议版本为协议名称及版本号。

【例 3-6】 分析如下代码所示请求行的请求方法,URL 及协议版本号分别是什么?

```
1   GET / HTTP/1.1
2   POST /v2/api/?login HTTP/1.1
```

第 1 行请求代码中,请求方法为 GET,URL 为"/"表示网站的根目录,访问默认页面如 index.html、index.jsp 等,协议版本号为 HTTP/1.1。

第 2 行请求代码中,请求方法为 POST,URL 为"/v2/api/?login", 协议版本号为 HTTP/1.1。

在 HTTP 的请求消息中,请求方法有 GET、POST、HEAD、OPTIONS、DELETE、TRACE、PUT 和 CONNECT 共 8 种,每种方法都指明了操作服务器中指定 URI 资源的方式。其中,GET 和 POST 是最常见的 HTTP 请求方法。请求方法及其含义如表 3-1 所示。

表 3-1 请求方法及其含义

请 求 方 法	含　　义
GET	请求获取请求行的 URI 所标识的资源
POST	向指定资源提交数据,请求服务器进行处理(如提交表单或者上传文件)
HEAD	请求获取由 URI 所标识资源的响应消息头
PUT	将网页放置到指定 URL 位置(上传/移动)
DELETE	请求服务器删除 URI 所标识的资源
TRACE	请求服务器回送收到的请求信息,主要用于测试或诊断
CONNECT	保留将来使用
OPTIONS	请求查询服务器的性能,或者查询与资源相关的选项和需求

(1) GET 请求方法与 POST 请求方法。

① GET 请求方法发送一个请求来获取服务器上的资源,资源通过 HTTP 响应头和数据(如 HTML 文档、图片、样式、视频等)返回给客户端(如浏览器)。GET 请求如果带参数,就在 URL 中附带查询参数,如 test.html?id=1。

② POST 请求方法用于向服务器提交数据,请求的参数要在请求实体中发送,可用于表单的提交和异步提交(如 Ajax)。理论上,POST 传递的数据量没有限制。

(2) GET 和 POST 发送 HTTP 请求。

在客户端如果发生下面的事件,浏览器就向 Web 服务器发送一个 HTTP 请求。

① 用户在浏览器的地址栏中输入 URL 并按回车键。

② 用户单击了 HTML 页面中的超链接。

③ 用户在 HTML 页面中填写一个表单并提交。

上述①和②向 Web 服务器发送的都是 GET 请求。如果使用 HTML 表单发送请求,那么可以通过 method 属性指定使用 GET 请求或 POST 请求。在默认情况下,使用表单发送的请求也是 GET 请求;如果发送 POST 请求,就需要将 method 属性值指定为 POST。

用户登录页面 login.jsp 的表单部分代码如下:

```
1       <form action = "user - login" method = "POST">
2          用户名:< input type = "text" name = "username" />
3          密码:< input type = "password" name = "password" />
4                < input type = "submit"   value = "登录">
5   </form >
```

在上述代码中,如若 method 属性的值为 POST,则使用 POST 方法提交表单信息;若 method 属性的值为 GET 或省略,则使用 GET 方法提交表单信息。

(3) GET 请求方法与 POST 请求方法的比较。

可通过它们请求的资源类型、发送数据类型、发送数据量、参数的可见性、数据是否缓存等方面做比较。GET 请求方法与 POST 请求方法的比较如表 3-2 所示。

表 3-2　GET 请求方法与 POST 请求方法的比较

特征	GET 请求方法	POST 请求方法
资源类型	静态的或动态的	动态的
数据类型	文本	文本或二进制数据
数据量	一般不超过 255 个字符	没有限制
可见性	数据是 URL 的一部分,在浏览器的地址栏中对用户可见	数据不是 URL 的一部分而是作为请求的消息体发送,在浏览器的地址栏中对用户不可见
数据缓存	数据可在浏览器的 URL 历史中缓存	数据不能在浏览器的 URL 历史中缓存

在实际开发中,通常都会使用 POST 方法发送请求,其原因主要有两个。

① POST 传输数据大小无限制。

由于 GET 请求方法是通过请求参数传递数据的,因此最多可传递 1KB 的数据。而 POST 请求方法是通过实体内容传递数据的,因此对传递数据的大小没有限制。

② POST 比 GET 请求方法更安全。

由于 GET 请求方法的参数信息都会在 URL 地址栏明文显示,而 POST 请求方法传递的参数隐藏在实体内容中,用户是看不到的,因此,POST 请求方法比 GET 请求方法更安全。

2. HTTP 请求消息头

在 HTTP 请求消息中,请求行之后,便是若干请求消息头。请求消息头主要用于向服务器端传递附加消息,例如,客户端可以接收的数据类型、压缩方法、语言以及发送请求的超链接所属页面的 URL 地址等信息。HTTP 消息头,以明文的字符串格式传送,是以冒号分隔的键/值对,如 Accept-Charset: UTF-8,每一个消息头的最后都以回车符(CR)和换行符(LF)结尾。HTTP 请求消息头代码如下:

```
1   Host: www.baidu.com
2   Connection: keep - alive
3   Cache - Control: max - age = 0 Upgrade - In
4   secure - Requests: 1
5   User - Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/81.0.4044.138 Safari/537.36
6   Accept: text/html,application/xhtml + xml,application/xml
```

每个请求消息头单独占一行,头字段名称不区分大小写,一般首字母大写。常用的请求消息头字段如表 3-3 所示。

表 3-3 常用的请求消息头字段

协 议 头	说 明
Accept	可接受的响应内容类型(Content-Types)
Accept-Charset	客户端可接受的字符集
Accept-Encoding	客户端可接受的响应内容的编码方式
Accept-Language	可接受的响应内容语言列表
Authorization	用于表示 HTTP 中需要认证资源的认证信息
Cache-Control	用来指定当前的请求/回复中是否使用缓存机制
Connection	客户端(浏览器)想要优先使用的连接类型
Cookie	由之前服务器通过 Set-Cookie(见下文)设置的一个 HTTP 协议 Cookie
Content-Length	以八进制表示的请求体的长度
Content-Type	请求体的 MIME 类型 (用于 POST 和 PUT 请求中)
Date	发送该消息的日期和时间(以 RFC 7231 中定义的"HTTP 日期"格式来发送)
Expect	表示客户端要求服务器做出特定的行为
Host	表示服务器的域名以及服务器所监听的端口号。如果所请求的端口是对应的服务的标准端口(80),则端口号可以省略
If-Match	仅当客户端提供的实体与服务器上对应的实体相匹配时,才进行对应的操作。主要用于像 PUT 这样的方法中,仅当从用户上次更新某个资源后该资源未被修改的情况下,才更新该资源
If-Modified-Since	允许在对应的资源未被修改的情况下返回 304 未修改
If-None-Match	允许在对应的内容未被修改的情况下返回 304 未修改(304 Not Modified),参考超文本传输协议的实体标记
If-Range	如果该实体未被修改过,则返回所缺少的那一个或多个部分;否则,返回整个新的实体
Referer	表示浏览器所访问的前一个页面,可以认为是之前访问页面的链接将浏览器带到了当前页面
User-Agent	浏览器的身份标识字符串

(1) Accept 头字段用于指出客户端程序(通常是浏览器)能够处理的 MIME(Multipurpose Internet Mail Extensions,多用途互联网邮件扩展)类型。例如,如果浏览器和服务器同时支持 png 类型的图片,则浏览器可以发送包含 image/png 的 Accept 头字段,服务器检查到 Accept 头中包含 image/png 这种 MIME 类型,可能在网页中的 img 元素中使用 png 类型的文件。MIME 类型有很多种,例如,下面的这些 MIME 类型都可以作为 Accept 头字段的值。

```
Accept:text/html,表明客户端希望接受 HTML 文本
Accept:image/gif,表明客户端希望接受 GIF 图像格式的资源
Acept:image/*,表明客户端可以接受所有 image 格式的子类型
Accept:*/*,表明客户端接受所有格式的类型
```

(2) Accept-Charset 头字段用于告知服务器端客户端所使用的字符集,具体示例如下:

```
Accept-Charset:UTF-8
```

在上面的请求头中,指出客户端服务器使用 UTF-8 字符集。如果想指定多种字符集,则可以在 Accept-Charset 头字段中将指定的多个字符集以逗号分隔,具体示例如下:

```
Accept-Charset:UTF-8,ISO-8859-1
```

需要注意的是，如果 Accept-Charset 头字段没有在请求头中出现，则说明客户端能接受使用任何字符集的数据。如果 Accept-Charset 头出现在请求消息里，但是服务器不能发送采用客户端期望字符集编码的文档，那么服务器将发送一个 406 错误状态响应，406 是一个响应状态码，表示服务器返回内容使用的字符集与 Accept-Charset 头字段指定的值不兼容。

(3) Accept-Encoding 头字段用于指定客户端能够进行解码的数据编码方式，这里的编码方式通常指的是某种压缩方式。在 Accept-Encoding 头字段中，可以指定多个数据编码方式，它们之间以逗号(,)分隔，具体示例如下：

```
1    Accept-Encoding: gzip, compress
```

在上面的消息头中，gzip 和 compress 这两种格式是最常见的数据编码方式。在传输较大的实体内容之前，对其进行压缩编码，可以节省网络带宽和传输时间。

需要注意的是，Accept-Encoding 和 Accept 头字段不同，Accept 头字段指定的 MIME 类型是指解压后的实体内容类型，而 Accept-Encoding 头字段则指定的是实体内容压缩的方式。

(4) Host 头字段用于指定资源所在的主机名和端口号，格式与资源的完整 URL 中的主机名和端口号部分相同，具体示例如下：

```
1      Host: www.baidu.com:80
2      Host: www.baidu.com
```

在上述示例中，由于浏览器连接服务器时默认使用的端口号为 80，所以“www.baidu.com”后面的端口号信息“:80”可以省略；第 1 行与第 2 行指定的 Host 头字段作用一致。

需要注意的是，在 HTTP 1.1 中，浏览器和其他客户端发送的每个请求消息中都必须包含 Host 请求头字段，以便 Web 服务器能够根据 Host 头字段中的主机名来区分客户端所要访问的虚拟 Web 站点。当浏览器访问 Web 站点时，会根据地址栏中的 URL 地址自动生成相应的 Host 请求头。

(5) If-Modified-Since 头字段的作用与 If-Mach 类似，只不过它的值为 GMT 格式的时间。If-Modified-Since 头字段被视作一个请求条件，只有服务器中文档的修改时间比 If-Modified-Since 头字段指定的时间新，服务器才会返回文档内容；否则，服务器将返回一个 304(Not Modified)状态码来表示浏览器缓存的文档是最新的，而不向浏览器返回文档内容，这时，浏览器仍然使用以前缓存的文档。通过这种方式，可以在一定程度上减少浏览器与服务器之间的通信数据量，从而提高通信效率。

(6) Referer 头字段标识发出请求的超链接所在网页的 URL。例如，本地 Tomcat 服务器的 ch3_demo 项目中有一个 Html 文件 get.html，get.html 中包含一个指向远程服务器的超链接，当单击这个超链接向服务器发送 GET 请求时，浏览器会在发送的请求消息中包含 Referer 头字段。其代码如下：

```
Referer:http://get.htmllocalhost:8080/ ch3_demo
```

Referer 头字段非常有用，常被网站管理人员用来追踪网站的访问者是如何导航进入网站的，同时 Referer 头字段还可以用作网站的防盗链。

(7) User-Agent(用户代理，UA)用于指定浏览器或者其他客户端程序使用的操作系统及版本、浏览器及版本、浏览器渲染引擎、浏览器语言等，以便服务器针对不同类型的浏览器而返回不同的内容。例如，Windows 10 操作系统下 Chrome 浏览器生成的 User-Agent 请求消

息头代码如下：

```
User - Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)) Chrome/81.0/81.0.4044.138 Safari/537.36
```


在上面的请求消息头中，User-Agent 头字段列出了 Mozilla 版本、操作系统的版本(Windows NT 10.0)、浏览器的引擎名称(AppleWebKit/537.36)及浏览器版本(Chrome/81.0)信息。

注意：操作系统不同，浏览器的 User-Agent 字段值也不相同。

由于篇幅有限，其他头字段可以参考 HTTP 文档，此处不再赘述。

3.2.3 HTTP 响应消息

一个 HTTP 响应代表服务器向客户端回送的数据，由三部分组成：响应状态行、响应消息头和响应数据。响应数据也叫响应消息体，即客户端浏览器显示的内容。HTTP 响应消息结构如图 3-11 所示。

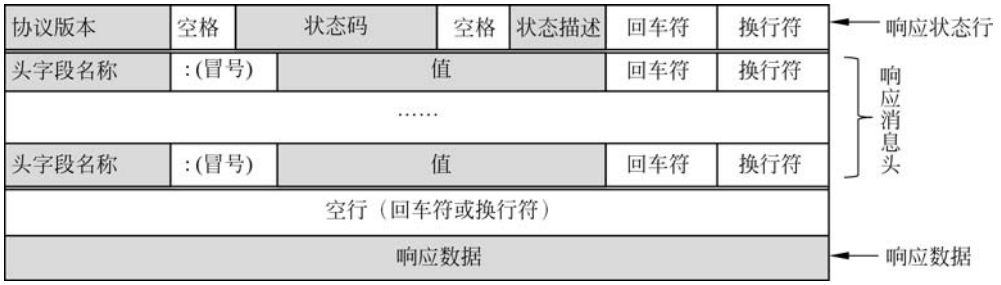


图 3-11 HTTP 响应消息结构

在 Chrome 浏览器中，Headers 标签中的 Response Header 部分包含了 HTPP 响应消息的状态行和响应消息头；而 Preview 标签或 Response 标签的内容则对应响应数据。

1. HTTP 响应状态行

HTTP 响应状态行由协议版本、响应状态码和状态描述三部分组成，各部分间以空格分隔。

响应状态码用于表示服务器对请求的处理结果，是一个三位的十进制数，分为五类。响应状态码及其含义如表 3-4 所示。

表 3-4 响应状态码及其含义

状态码	含　义	常用状态码
1xx	表示成功接收请求，要求客户端继续提交下一次请求才能完成整个处理过程	100 表示服务器同意处理客户的请求
2xx	表示成功接受请求并已完成整个处理过程	200 表示请求成功，204 表示内容不存在
3xx	为完成请求，客户需进一步细化请求	301 表示页面移走了，304 表示缓存的页面仍然有效
4xx	客户端的请求有错误	403 表示禁止的页面，404 表示页面没有找到
5xx	服务器端出现错误	500 表示服务器内部错误，503 表示以后再试

HTTP 响应状态行代码示例如下：

```
1  HTTP/1.1 404 Not Found,页面没有找到
2  HTTP/1.1 500 Internal Error,服务器内部错误
3  HTTP/1.1 200 OK,请求成功
```

2. HTTP 响应消息头

服务器端通过响应消息头向客户端传递附加信息，包括服务程序名、被请求资源需要的认证方式、客户端请求资源的最后修改时间、重定向地址等信息。HTTP 响应消息头的具体示例如下：

```
Bdpagetype: 1
Bdqid: 0xc625bd4d000d4ea7
Cache-Control: private
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html;charset=UTF-8
Date: Thu, 23 Jul 2020 17:07:00 GMT
Expires: Thu, 23 Jul 2020 17:07:00 GMT
Server: BWS/1.1
Set-Cookie: BDSVRTM=16; path=/
```

从上面的响应消息头可以看出，它们的格式和 HTTP 请求消息头的格式相同。当服务器向客户端回送响应消息时，根据情况的不同，发送的响应消息头也不相同。常用的响应消息头字段如表 3-5 所示。

表 3-5　HTTP 常用的响应消息头字段

响　应　头	说　　明
Accept-Ranges	服务器所支持的内容范围
Age	响应对象在代理缓存中存在的时间，以秒为单位
Cache-Control	通知从服务器到客户端内的所有缓存机制，表示它们是否可以缓存这个对象及缓存有效时间。其单位为秒
Content-Disposition	对已知 MIME 类型资源的描述，浏览器可以根据这个响应头决定对返回资源的动作，如将其下载或是打开
Content-Type	当前内容的 MIME 类型
Date	此条消息被发送时的日期和时间
ETag	对于某个资源的某个特定版本的一个标识符，通常是一个消息散列
Expires	指定一个日期/时间，超过该时间则认为此回应已经过期
Last-Modified	所请求的对象的最后修改日期
Location	用于在进行重定向或在创建了某个新资源时使用
Proxy-Authenticate	要求在访问代理时提供身份认证信息
Refresh	用于重定向或者当一个新的资源被创建时。默认会在 5 秒后刷新重定向
Server	服务器的名称
Set-Cookie	设置 HTTP Cookie
Vary	告知下游的代理服务器，应当如何对以后的请求协议头进行匹配，以决定是否可使用已缓存的响应内容而不是重新从原服务器请求新的内容
WWW-Authenticate	表示在请求获取这个实体时应当使用的认证模式

(1) Location 头字段用于通知客户端获取请求文档的新地址，其值为一个使用绝对路径的 URL 地址，具体示例如下：

```
Location:http://www.people.com.cn
```

Location 头字段和大多数 3xx 状态码配合使用，以便通知客户端自动重新连接到新的地

址请求文档。由于当前响应并没有直接返回内容给客户端,所以使用 Location 头的 HTTP 消息不应该有实体内容。由此可见,在 HTTP 消息头中不能同时出现 Location 和 Content-Type 两个头字段。

(2) Server 头字段用于指定服务器软件产品的名称,具体示例如下:

```
Server: Apache
```

(3) Refresh 头字段用于告诉浏览器自动刷新页面的时间,它的值是一个以秒为单位的时间数,具体示例如下:

```
Refresh:2
```

上面所示的 Refresh 头字段用于告诉浏览器在 2 秒后自动刷新此页面。

注意:在 Refresh 头字段的时间值后面还可以增加一个 URL 参数,时间值与 URL 之间用分号(;)分隔,用于告诉浏览器在指定的时间值后跳转到其他网页,例如告诉浏览器经过 2 秒跳转到 www.people.com.cn 网站,具体示例如下:

```
Refresh:2;url = http://www.people.com.cn
```

(4) Content-Disposition。

服务器向客户端浏览器发送文件时,如果是浏览器支持的文件类型(如 txt、jpg 等),一般会默认直接在浏览器中显示;如果是让用户选择将响应的实体内容保存到一个文件中,就需要使用 Content-Disposition 头字段。

Content-Disposition 属性是作为对下载文件的一个标识字段,Content-Disposition 属性有两种类型:inline 和 attachment。inline 是将文件内容直接显示在页面;attachment 是弹出对话框让用户下载。

attachment 后面还可以指定 filename 参数。filename 参数值是服务器建议浏览器保存实体内容的文件名称,浏览器应该忽略 filename 参数值中的目录部分,只取参数中的最后部分作为文件名。在设置 Content-Disposition 之前,一定要设置 Content-Type 头字段,具体示例如下:

```
1  Content - Type:application/octet - stream
2  Content - Disposition: attachment; filename = "filename.xls"
```

设置如上所示响应消息头后,浏览器会提示保存还是打开,即使选择打开,也会使用相关联的程序,比如使用记事本打开,而不是用浏览器直接打开。

3.3 开发环境配置

视频讲解

3.3.1 开发工具介绍

1. JDK

JDK(Java Development Kit,Java 开发工具包)由 Sun 公司提供。它为 Java 程序的开发提供了编译和运行环境,所有的 Java 程序的编写都依赖于它。JDK 有 J2SE、J2EE 和 J2ME 三个版本。其中,J2SE 为标准版,主要用于开发桌面应用程序;J2EE 为企业版,主要用于开发企业级应用程序,如电子商务网站和 ERP 系统等;J2ME 为微缩版,主要用于开发移动设备、嵌入式设备上的 Java 应用程序。JDK 的版本更新较快。但 JDK8 版本比较稳定,适合用于

Eclipse 等多种开发软件。本书中案例开发环境使用的是 JDK8 版本。

2. Tomcat 服务器

Tomcat 是 Apache 组织的 Jakarta 项目中的一个重要子项目，它是 Sun 公司推荐的运行 Servlet 和 JSP 的容器(引擎)，其源代码是完全公开的。Tomcat 不仅具有 Web 服务器的基本功能，还提供了数据库连接池等许多通用组件功能。

Tomcat 运行稳定、可靠、效率高，不仅可以和目前大部分主流的 Web 服务器(如 Apache、IIS 服务器)一起工作，还可以作为独立的 Web 服务器软件。因此，越来越多的软件公司和开发人员都使用它作为运行 Servlet 和 JSP 的平台。

Tomcat 的版本在不断地升级，功能也不断地完善与增强。目前最新版本为 Tomcat 9.0，本书中以 Tomcat v8.5 作为 Java Web 应用开发服务器。

3. Eclipse 开发平台

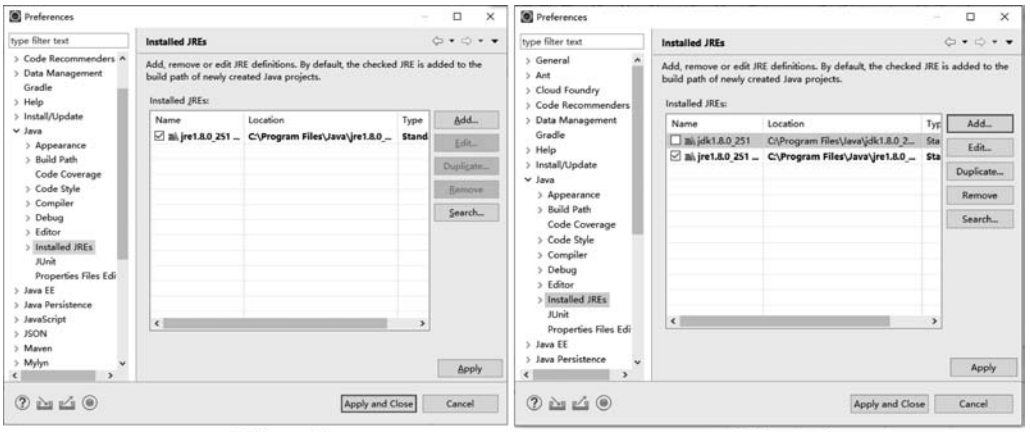
Eclipse 是一个基于 Java、开放源码并可扩展的应用开发平台，为开发人员提供了一流的 Java 集成开发环境。它是一个可以用于构建集成 Web 应用程序开发工具的平台，其本身并不提供大量的功能，而是通过插件来实现程序的快速开发。对于 Java 应用程序开发者来说，可下载普通的 J2SE 版本的 Eclipse；而对于 Java Web 应用程序开发者来说，需要使用 J2EE 版本的 Eclipse。

Eclipse 版本更新较快，本书中使用 photon 版，这可以从 Eclipse 的官方网站下载。在 Eclipse 下载完成后，将其解压到指定的文件夹下，就完成了 Eclipse 的安装。

3.3.2 在 Eclipse 中配置 JDK

在 Eclipse 中指定使用 jdk1.8.0_251 版本的 JRE，配置操作步骤如下：

(1) 单击菜单栏 Window→Preferences 命令，打开 Preferences 对话框，在左侧选择 Java→Installed JREs，右边窗口就出现了 JDK 的配置项，如图 3-12(a)所示。



(a) 添加JRE前　　(b) 添加JRE后

图 3-12　Preferences 对话框

(2) 单击 Add 按钮，弹出 Add JRE 对话框的类型选择窗口。这里会要求选中一个 JRE 版本添加到工作空间中，选择 Standard VM，单击 Next 按钮，进入 Add JRE 对话框的 JRE Definition 窗口。Add JRE 对话框如图 3-13 所示。

(3) 单击 Directory 按钮，选择 jdk1.8.0_251 的安装目录，就会在 JRE Name 文本输入框中自动添加名称 jdk1.8.0_251，单击 Finish 按钮，完成 JRE 的添加，返回 Preferences 对话框，如图 3-12(b)所示。在 JRE 列表中选择需要使用的 JRE 即完成了 Eclipse 中 JDK 的配置。

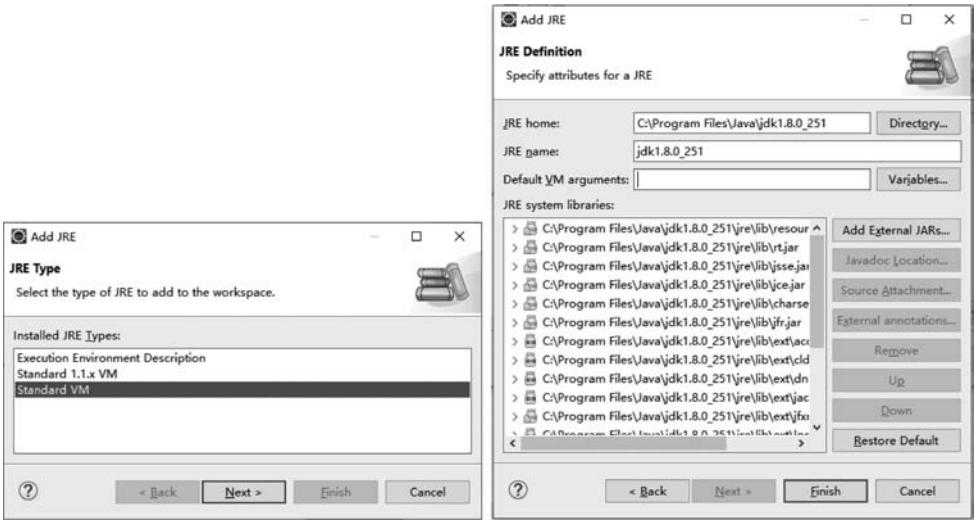


图 3-13　Add JRE 对话框

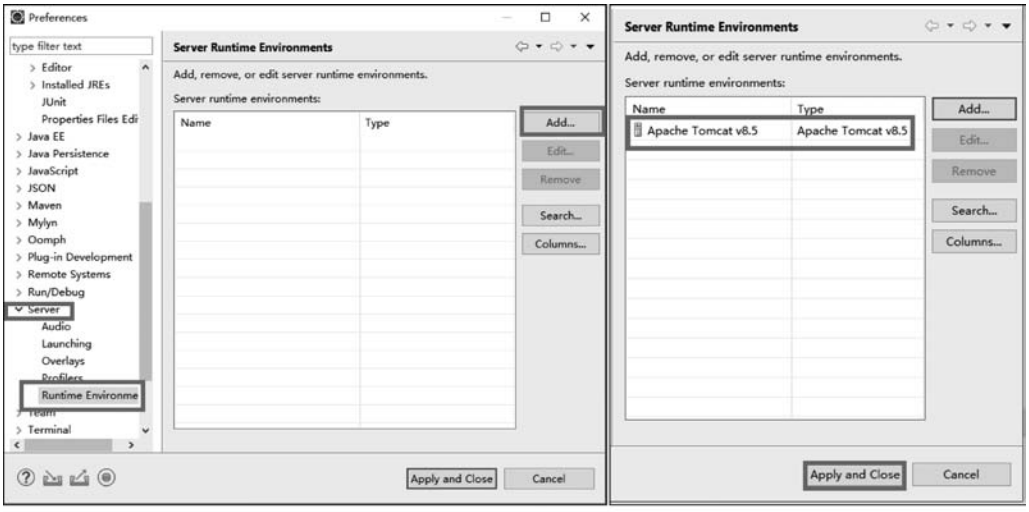
注意：在首次启动 Eclipse 时，默认会选择当前安装的 JRE。如果对 JDK 的版本没有特别要求，就不需要单独配置 JRE。

3.3.3　在 Eclipse 中配置 Tomcat

在 Eclipse 中指定使用 Tomcat v8.5，首先要配置服务器环境。其操作步骤如下：

1. 关联 Eclipse 和 Tomcat 服务器

(1) 选择菜单栏 Window→Preferences 选项，打开 Preferences 对话框，在其左侧选择 Server→Runtime Environments，右边窗口就出现了 Server 的配置项如图 3-14(a)所示。



(a)　　(b)

图 3-14　Server Runtime Environments 对话框

(2) 单击 Add 按钮，在弹出的选择 Tomcat 服务器对话框中选择 Apache Tomcat v8.5。选择 Tomcat 服务器对话框如图 3-15(a)所示。

(3) 单击 Next 按钮，在弹出的指定 Tomcat 路径对话框中单击 Browse 按钮，选择 Tomcat v8.5 的安装路径如图 3-15(b)所示。

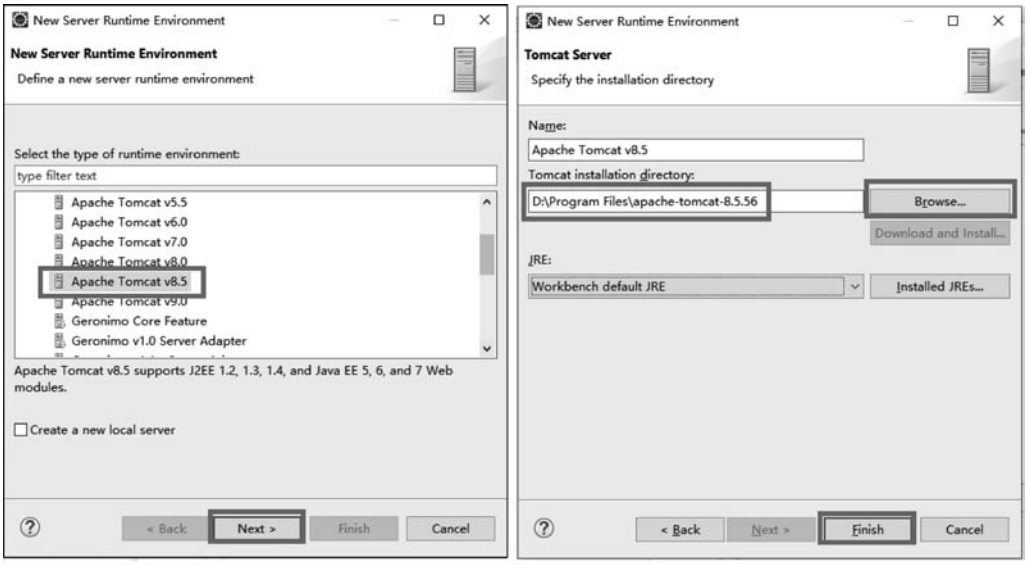


图 3-15　选择 Tomcat 服务器对话框

(4) 单击 Finish 按钮,添加完成,返回到 Server Runtime Environments 对话框,如图 3-14(b)所示。在此对话框中单击 Apply and Close 按钮,即完成 Eclipse 与 Tomcat 服务器的关联。

2. 在 Eclipse 中创建 Tomcat 服务器

单击 Eclipse 下侧窗口的 Servers 视图(如果没有这个视图,可以通过选择菜单 Window→Show View 选项打开),在选项卡中可以看到一个"No servers available. Define a new server from the new server wizard…"的链接,单击这个链接,会弹出 New Server 对话框,如图 3-16 所示。

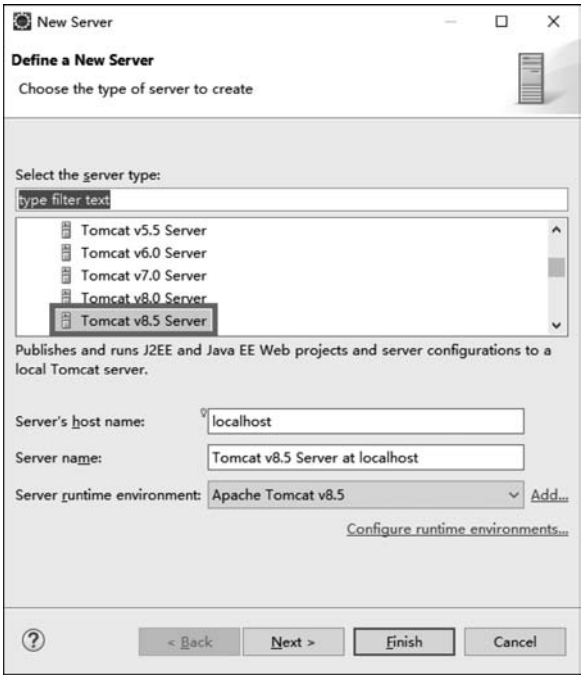


图 3-16　New Server 对话框

选中图 3-16 所示的 Tomcat v8.5 Server 选项,单击 Finish 按钮,完成 Tomcat 服务器的创建。这时,在 Servers 视图中会出现一个“Tomcat v8.5 Server at localhost[Stopped]”选项,具体如图 3-17 所示。

图 3-17　在 Eclipse 中创建的 Tomcat 服务器

3. 更改 Tomcat 发布路径

在 Tomcat 服务器创建完毕后就可以使用了。此时如果创建了项目,并使用 Eclipse 发布后,由于项目会发布到 Eclipse 的 metadata 文件夹中,故不便于发布者查找发布的项目。为了方便查找发布后的项目目录,可以将项目直接发布到 Tomcat 中,这就需要对 Server 进行配置。具体配置方法如下。

双击图 3-17 中创建好的 Tomcat 服务器,在打开的 Overview 对话框中,选择 Server Locations 选项中的“Use Tomcat installation”选项,并将 Deploy path 文本框中的内容改为 webapps,单击 Eclipse 工具栏上的 (保存)按钮进行保存。至此,就完成了 Tomcat 的所有配置。Overview 对话框如图 3-18 所示。

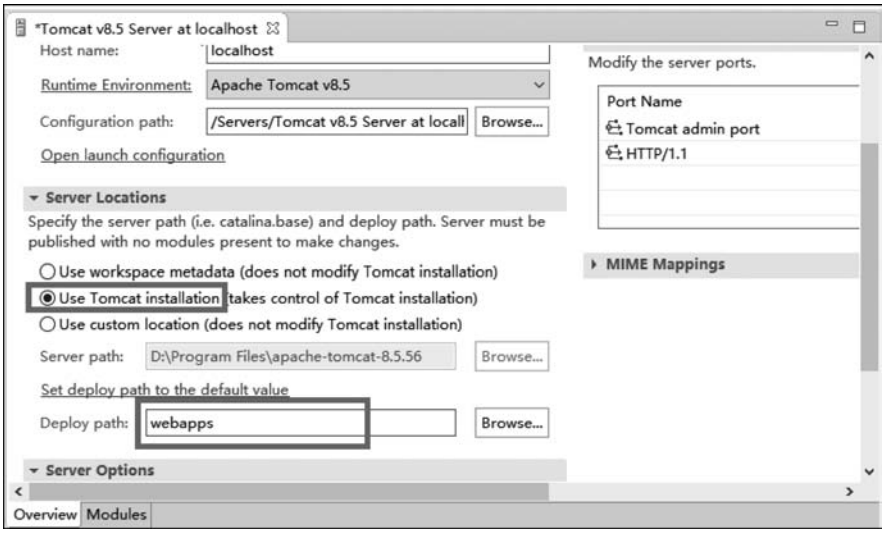


图 3-18　Overview 对话框

4. 测试 Tomcat 的配置

单击图 3-17 所示的 Servers 视图工具栏中的 按钮,即可启动 Tomcat 服务器。在浏览器地址栏中输入 http://localhost:8080/,如果能打开 Tomcat 服务器首页,说明 Tomcat 服务器启动成功。Tomcat 服务器首页如图 3-19 所示。


视频讲解

3.3.4　创建第一个 Java Web 项目

编写一个简单的 Java Web 网站,访问该网站时,页面上输出“维护网络空间生态环境,从我做起……”。

图 3-19　Tomcat 服务器首页

1. 创建项目

在 Eclipse 中动态网站的项目类型是 Dynamic Web Project，设定项目名称为 ch3_demo，其他选项取默认值。

2. 创建 JSP 文件

在项目 ch3_demo 中，右击 WebContent 文件夹，在 New 菜单中选择 JSP File 选项，打开 New JSP File 对话框，输入文件名 index.jsp，单击 Finish 按钮，文件创建完成。

index.jsp 文件代码如下：

```
1   <%@ page language="java" contentType="text/html; charset=GB18030"   pageEncoding="
    GB18030"%>
2   <!DOCTYPE html>
3   <html>
4       <head>
5           <meta charset="GB18030">
6           <title>第一个 Java Web 项目</title>
7       </head>
8       <body>
9           <center>维护网络空间生态环境，从我做起……</center>
10      </body>
11  </html>
```

单击工具栏上的保存按钮。至此，网站创建完成。

3. 部署动态网站

Java Web 项目创建完成后，即可将项目发布到 Tomcat 并运行该项目。下面介绍具体的操作方法。

(1) 在项目资源管理器中选择项目名称，在工具栏上单击 按钮中的倒三角，在弹出的菜单中选择 Run As(运行方式)→Run On Server(在服务器上运行)选项，打开 Run On Server 对话框，在该对话框中选中 Always use this server when running this project(将服务器设置为默认值)复选框，其他采用默认如图 3-20 所示。

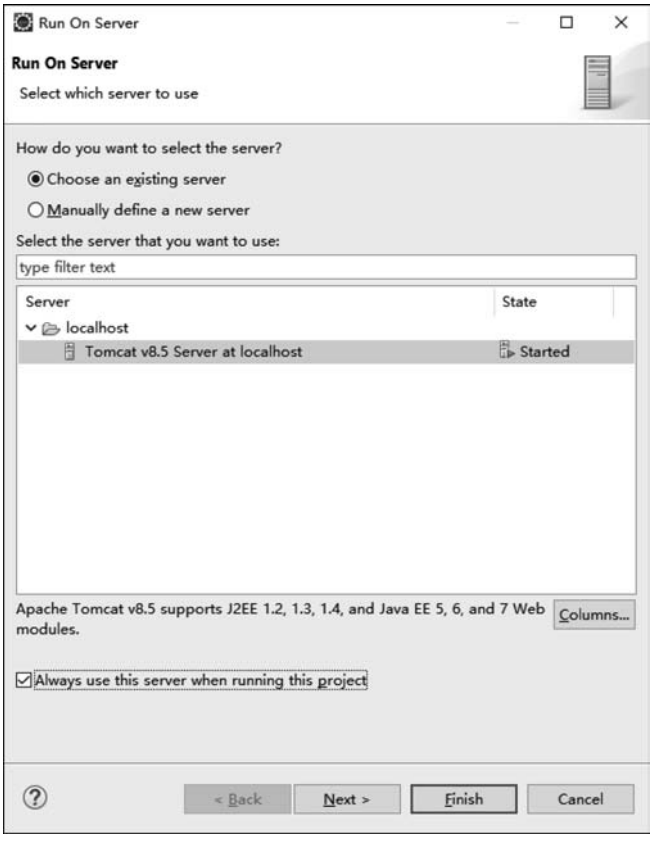


图 3-20　Run On Server 对话框

(2) 单击 Finish 按钮,即可通过 Tomcat 运行该项目,运行后的效果如图 3-21 所示。

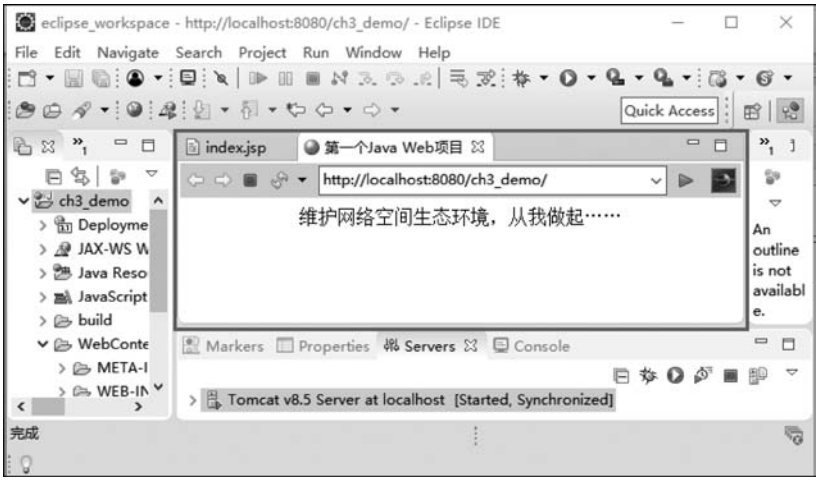


图 3-21　运行 ch3_demo 项目效果示意

注意:

(1) 如果服务器已经启动了,这时会弹出一个对话框询问是否要重启服务器。建议选择第一个选项 Restart Server(重启服务器)。

(2) 如果想在其他浏览器中查看该网站，可以将地址直接复制到浏览器的地址栏中，按回车键运行即可。

3.4 本章小结

本章主要介绍了与 Java Web 应用程序开发相关的概念和开发环境的配置。3.1 节介绍 XML 的概念，包括 XML 文档结构、XML 语法等，并举例说明 XML 文档的创建；3.2 节详细介绍了 HTTP 的概念及 HTTP 的请求和响应消息，并通过其开发者工具观察请求头字段与响应头字段的值，理解其含义；3.3 节介绍 Java Web 开发工具的下载及开发环境的配置，重点介绍 Tomcat 服务器的安装以及在 Eclipse 中配置服务器环境，并完成第一个动态网站的创建。