第2章补充内容2-- WEB application

2.1 HTTP简介

2.2 WEB application

2.3 WEB Modules

2.4 WEB Application Life Cycle

2.5 WEB module 的部署

2.6 case: Library web application

2.1 HTTP简介

HTTP(Hyper Text Transfer Protocol)是WEB应用的标准通信协议

2.1.1 HTTP的特点

HTTP协议采用“**请求/响应**”模型：

1）客户端（如浏览器）向服务器发请求(request)，索取特定资源（任何类型：HTML文档、动态内容）

2）服务器端若能满足要求，则返回一个包含正确内容的Response(响应)，否则客户端收到的响应里面将包含一段错误信息。

HTTP协议特点：

1）**无状态协议**(Stateless Protocol)

服务器每次响应完一次请求后，不会留下客户端的任何信息。

2）无法判断请求来源

点击链接、单击按钮、用户组件等

解决办法：

1）无状态：Servlet

2）请求来源：Servlet、JSP以及客户端脚本(javascript)

2.1.2 URL

http://www.ecust.edu.cn/index.htm

协议：//主机地址：端口/资源地址

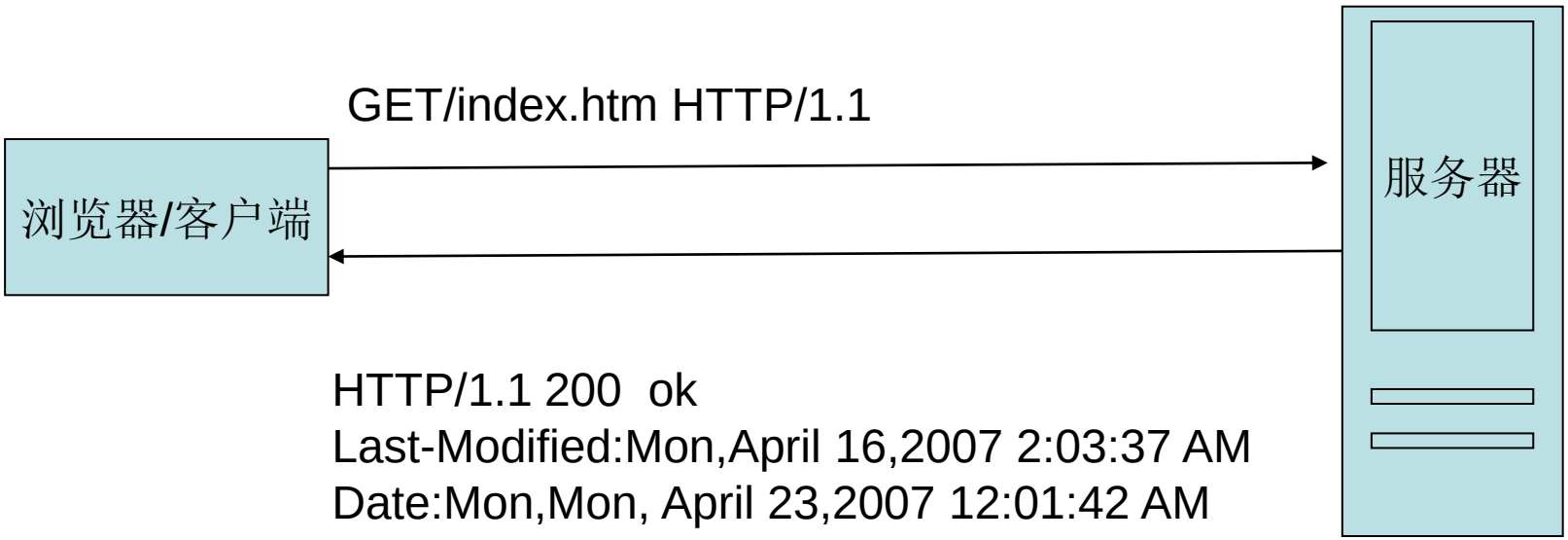
2.1.3 WEB浏览器和WEB服务器的交互

WEB浏览器或者客户机采用如下的方式来向WEB服务器请求文档：

打开一个TCP/IP 套接字(socket)，写入请求消息以及可能有关请求的数据，然后从套接字的输入流读取所请求文档的内容；

请求和响应都是用ASCII文本来编写的。

工作流程如下图所示：



HTTP/1.1 200 ok
Last-Modified:Mon,April 16,2007 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:text/html
Content-Length:186

```
<html>
  <head>
     <title> JSP Page</title>
</head>
<body>
   <h1> JSP PAGE</h1>
</body>
</html>
```

例: 当用户在超链接单击或者在浏览器中打开一个url:
http://www.ecust.edu.cn/index.html

浏览器做如下的事情（1—6步骤）：

（1）分析这个URL以获得它的各个成分：

协议：http

主机名称：www.ecust.edu.cn

文件名称：/index.html

（2）使用默认的http端口80来打开到www.ecust.edu.cn的一个套接字连接

（3）为所请求的文档编写一个http GET请求：

GET/index.htm HTTP/1.1

（4） 读取HTTP响应头和文档的内容：

```
HTTP/1.1 200  ok
Last-Modified:Mon,April 16,2007 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:text/html
Content-Length:186

<html>
  <head>
     <title> JSP Page</title>
</head>
<body>
   <h1> JSP PAGE</h1>
</body>
</html>
```

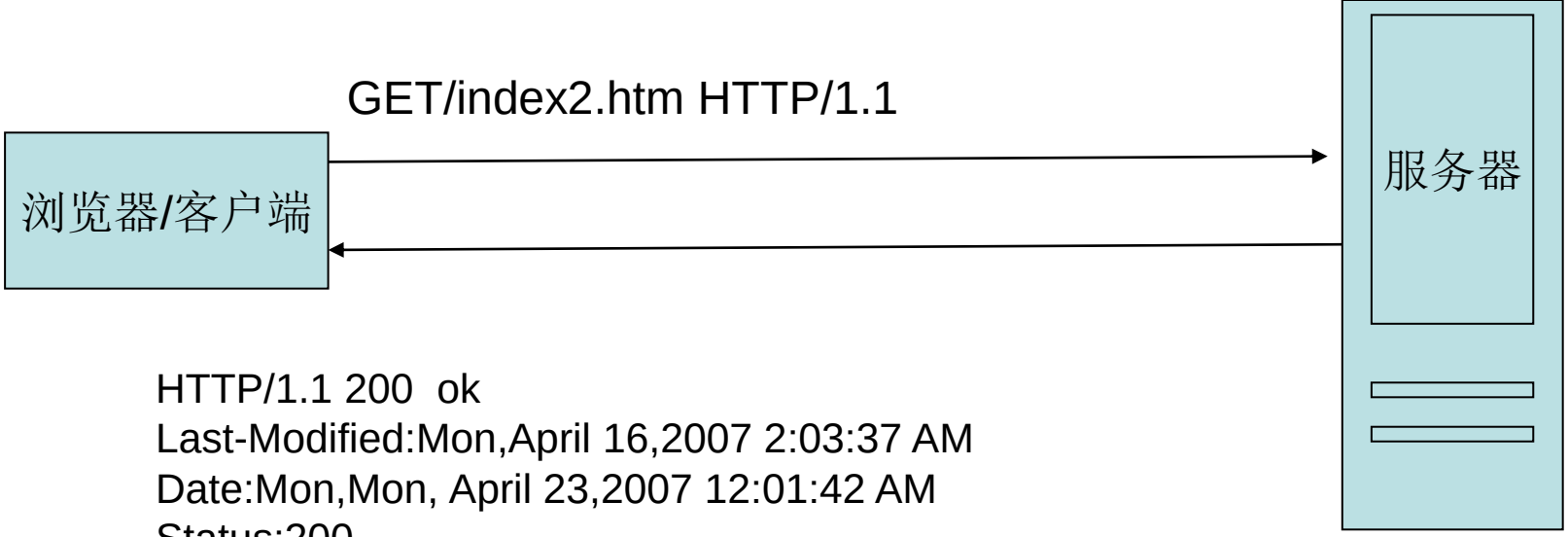
（5） 在浏览器的窗口中把所得到的文档格式化并显示出来 。

（6） 关闭套接字连接

组合请求：（HTML文档不仅是文本和链接，还有GIF和JPEG图像）

浏览器不是把文本和二进制图像数据组合到一起填入到单个请求中，而是发出多个请求，一个请求文本，其余各个请求所引用的任何图像。

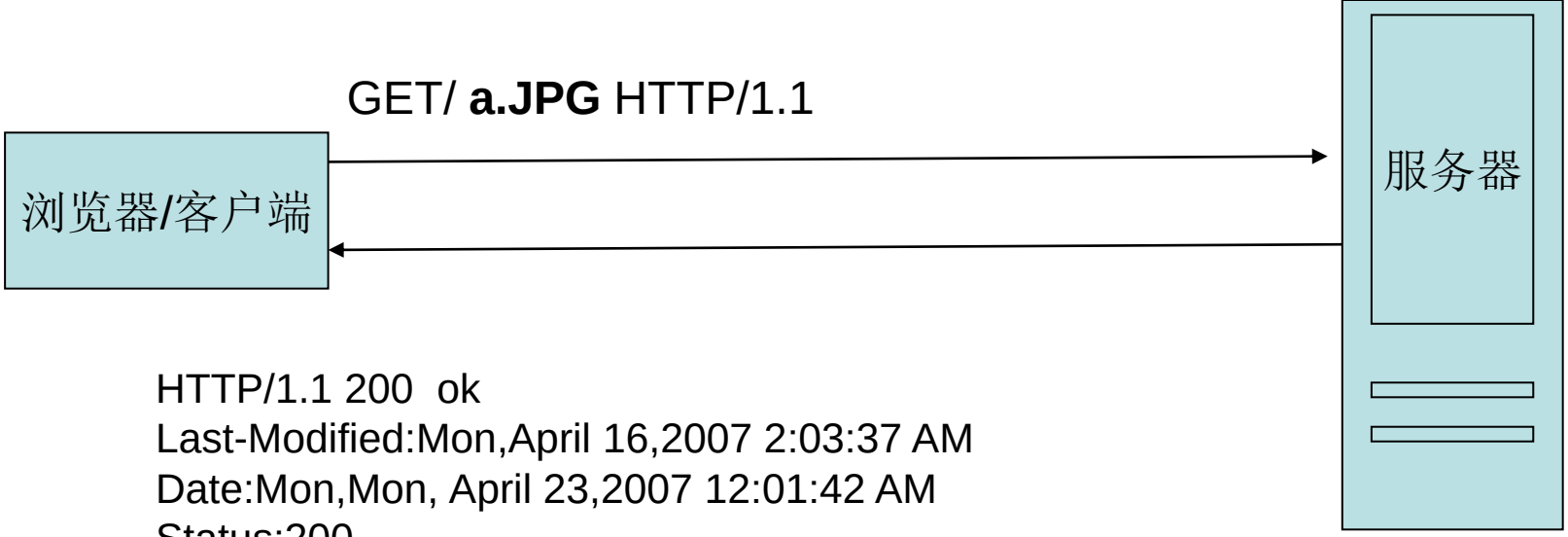
例:当用户在超链接单击或者在浏览器中打开一个url:
http://www.ecust.edu.cn/index2.html



HTTP/1.1 200 ok
Last-Modified:Mon,April 16,2007 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:text/html
Content-Length:186

```
<html>
  <head>
     <title> JSP Page</title>
</head>
<body>
   <h1> JSP PAGE
     <img src="a.JPG" alt="image" width="640" height="400">
   </h1>
</body>
</html>
```

①



HTTP/1.1 200 ok
Last-Modified:Mon,April 16,2007 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:image/jpeg
Content-Length:15667

(字节数据）

②

浏览器做如下的事情（1—6步骤）：

（1）分析这个URL以获得它的各个成分：

协议：http

主机名称：www.ecust.edu.cn

文件名称：/index2.html

（2）使用默认的http端口80来打开到www.ecust.edu.cn的一个套接字连接

（3）为所请求的文档编写一个http GET请求：

GET/index2.htm HTTP/1.1

（4） 读取HTTP响应头和文档的内容：

HTTP/1.1 200 ok
Last-Modified:Mon,April 16,2007 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:text/html
Content-Length:186

```
<html>
  <head>
     <title> JSP Page</title>
</head>
<body>
   <h1> JSP PAGE
      <img src="a.JPG" alt="image" width="640" height="400">
   </h1>
</body>
</html>
```

（5） 关闭套接字连接

（6） 扫描文本看是否有任何图像的URL(例如：a.jpeg)，对于所找到的每一个请求图像的URL，就使用图像的URL来重复步骤1-5。

（7） 在浏览器的窗口中把所得到的文档格式化并显示出来 。

JSP Page - Windows Internet Explorer

D:\save\html\index2.htm

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

收藏夹 JSP Page

JSP PAGE

2.1.4 HTTP请求(Request)

一个典型的HTTP请求包括：

一个请求命令行(Request Line)

一段请求报头（Request Header)

一个空行

可选的请求主体（Request Body)

例如：

请求方法:**GET**、**POST**、HEAD、PUT、DELETE、OPTIONS、TRACE、CONNECT

GET/index.htm HTTP/1.1 ←——请求命令行

Host:www.ecust.edu.cn

Connection:close

Accept-Encoding:gzip

Accept:*/*

Accept-Language:en-us

Accept-Charset:iso-8859-1,*,utf-8

User-Agent:Mozilla/4.0(compatible;MSIE 7.0;Windwos NT 6.0;.NET 3.0.04506)

Referer:http://www.sun.com

请求报头

←—— 空行:通知服务器以下不再有报头。

请求主体只适用于某些请求方法，如使用POST传送数据。GET方法没有配合请求主体。

（1） POST和GET方法

除了请求报头之外，还可以通过请求参数向服务器传递更多额外的信息。

对应于不同的请求方法，主要有用于GET方法的查询字符串方式（Query String)和用于POST方法的请求主体方式。

例 :GET方法

http://www.library.edu.cn/servlet/Login?user=Liang&Password=123

使用POST方式，浏览器则会向浏览器发送如下的请求：

例如：

POST /servlet/Login HTTP/1.1
Host:www.library.edu.cn
Connection:close
Accept-Encoding:gzip
Accept:*/*
Accept-Language:en-us
Accept-Charset:iso-8859-1,*,utf-8
User-Agent:Mozilla/4.0(compatible;MSIE 7.0;Windwos NT 6.0;.NET 3.0.04506)

user=Liang&Password=123

GET与POST方法有以下区别：

（1） 在客户端，Get方式在通过URL提交数据，数据在URL中可以看到；POST方式，数据放置在HTML HEADER内提交。

（2） GET方式提交的数据最多只能有1024字节，而POST则没有此限制。

（3） 安全性问题。正如在（1）中提到，使用 Get 的时候，参数会显示在地址栏上，而 Post 不会。所以，如果这些数据是中文数据而且是非敏感数据，那么使用get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post为好。

2.1.5 HTTP响应(Response)

WEB服务器收到请求后，根据请求的URL、请求报头和请求参数等信息进行处理。

有可能读取一个静态的HTML页面、图片、服务器端组件等。

HTTP响应的格式：
状态行、响应报头、一个空行、响应主体组成。

例如：　状态码：成功

HTTP/1.1 200 ok ← 状态行

Last-Modified:Mon,April 16,2007 2:03:37 AM
Date:Mon,Mon, April 23,2007 12:01:42 AM
Status:200
Content-Type:text/html
Content-Length:186

} 响应报头

← 一个空行

```
<html>
  <head>
     <title> JSP Page</title>
</head>
<body>
   <h1> JSP PAGE</h1>
</body>
</html>
```

} 响应主体

2.2 web application

2.2.1 A web application is a dynamic extension of a web or application server.

2.2.2 类型:

1)Presentation-oriented web application :

产生可交互的WEB 页面 (包含各种标记语 言: HTML、XML等），以便动态地响应用户的请求。

2)Service-oriented web application :

A service-oriented web application implements the endpoint of a web service.

Presentation-oriented applications are often clients of service-oriented web applications.

2.2.3 web application 的构成：

web components,

static resource files （ images），

Helper classes，和

libraries.

2.2.4 web components ： provide the dynamic extension capabilities for a webserver.

Java servlets, JSP pages, web service endpoints

(1)Servlets是java语言编写的类,它可以动态地处理请求(requests)和创建响应(responses).

(2) JSP pages 是文本型文件

JSP pages execute as servlets

(3) JavaServer Faces技术 建立在Servlet 和JSP技术基础之上，为WEB 应用系统提供了用户接口组件框架。

特点：

（1）Servlets 适合 service-oriented applications (web service endpoints are implemented as servlets) 以及在presentation- oriented application中扮演控制流程角色

（2）JSP pages 适合于页面标记语言：HTML、XML等。

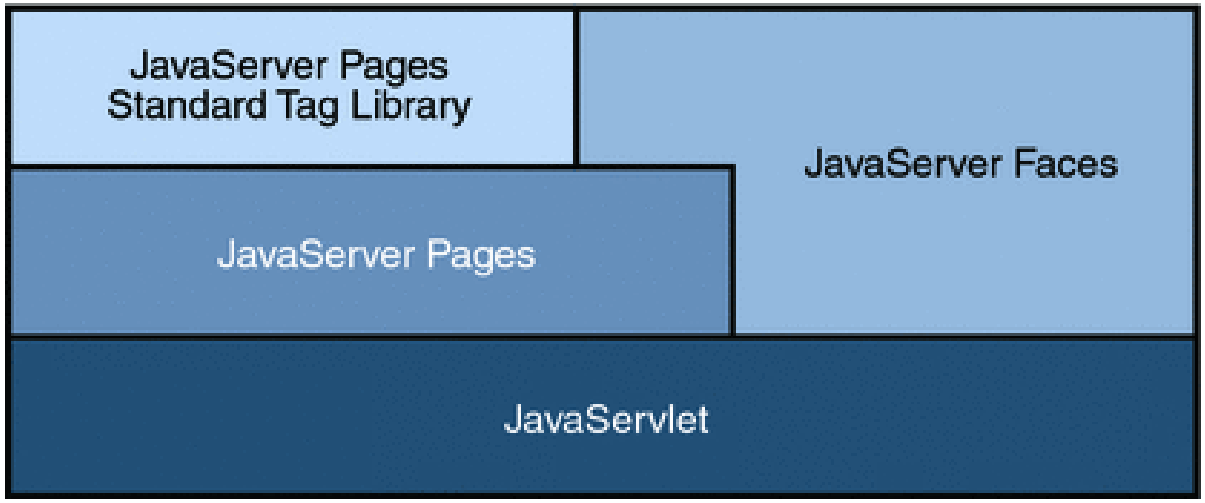


FIGURE 2–1 JavaWebApplication Technologies

说明：

1）Servlet技术是WEB应用系统的基础

2）其他技术是在Servlet技术基础上做了某些抽象，使WEB系统更好编写、维护、健壮。

3）大多数JavaEE Web clients 使用HTTP protocol，因此对HTTP 的支持是Web components的主要方面.

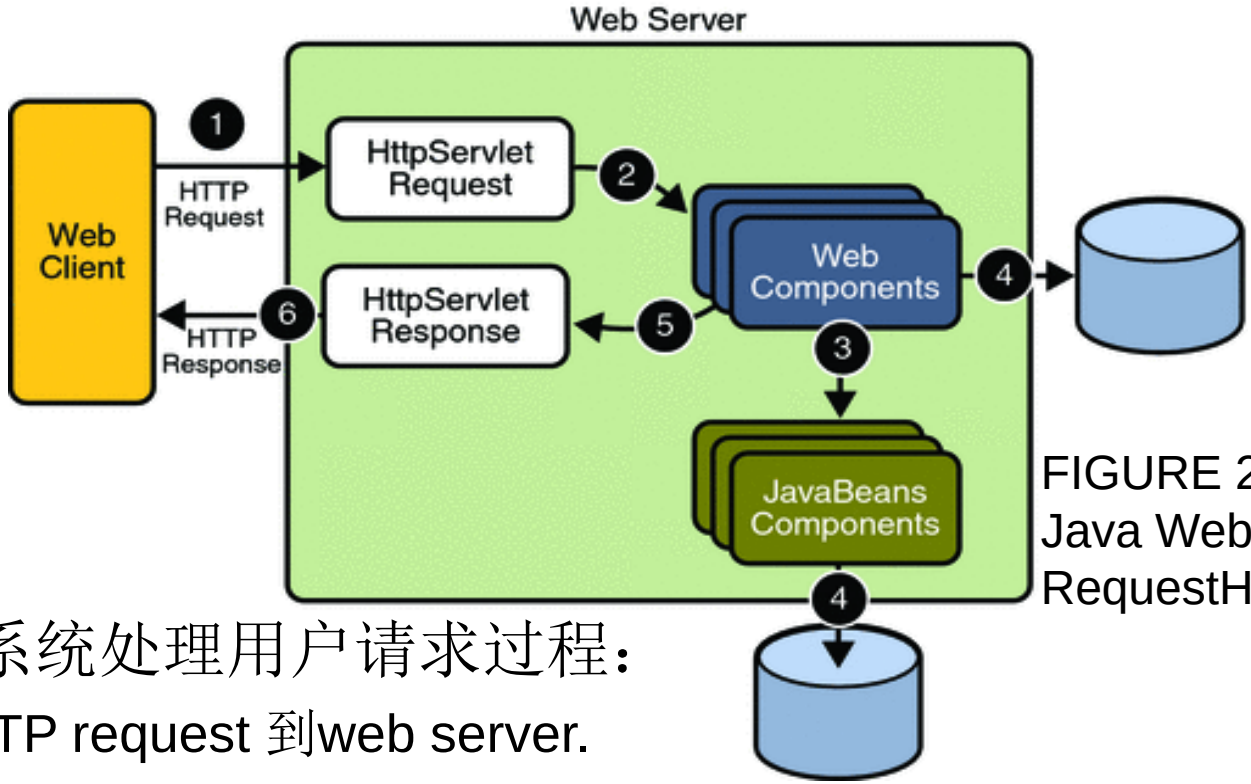


FIGURE 2–2
Java WebApplicati
RequestHandling

2.2.5 WEB应用系统处理用户请求过程：

1) 用户发送一个HTTP request 到web server.

2)web server把request转换成一个 HTTPServletRequest object

3)HTTPServletRequest object被传递给WEB组件，从而WEB组件可以读取HTTPServletRequest object的内容，以获得用户的信息。

4) WEB组件可以直接访问数据库，或者通过JavaBEAN，以产生动态的响应内容； WEB组件可以调用另外的WEB组件，并且把HTTPServletRequest object 传递过去。

5)然后WEB组件会生成一个HTTPServletResponse object

6)最后web server把HTTPServletResponse object转换成一个HTTP response，传递给用户。

2.2.6影响web application运行时的行为

1)web container为Web components 提供运行时的服务：

request dispatching,
security,
concurrency,
life-cycle management，以及
访问API(naming,transactions, and email)

2)web application有一个web application 配置文件（web application deployment descriptor）：

使得web application 被部署到web container后，某些行为会有些差异。

2.3 Web Modules

2.3.1 Web Modules概述

web resources : web components 与 static web content files （如： images）。

web module： A web module is the smallest deployable and usable unit of web resources.

一个web module 与一个 web application相对应

组成：

1)web components

2)static web content files

3)Server-side utility classes (database beans, shopping carts等)：一般要符合JavaBeans 组件规范

4)Client-side classes (applets 、 utility classes).

说明:

1)一个WEB应用程序是由一组Servlet、JSP、HTML页面、java类以及其他的资源组成的运行在WEB服务器(web容器:web container)中的完整的应用程序。

2)WEB应用程序以一种结构化的有层次的目录形式存在。

3)一个WEB容器可以运行多个Web应用程序，每个Web应用程序都有一个唯一的上下文根，用户通过WEB应用程序的上下文根来访问WEB应用程序中的资源。

上下文根的确定是与**具体的WEB容器**、**如何部署Web应用程序**相关。

2.3.2 web module 的目录结构：

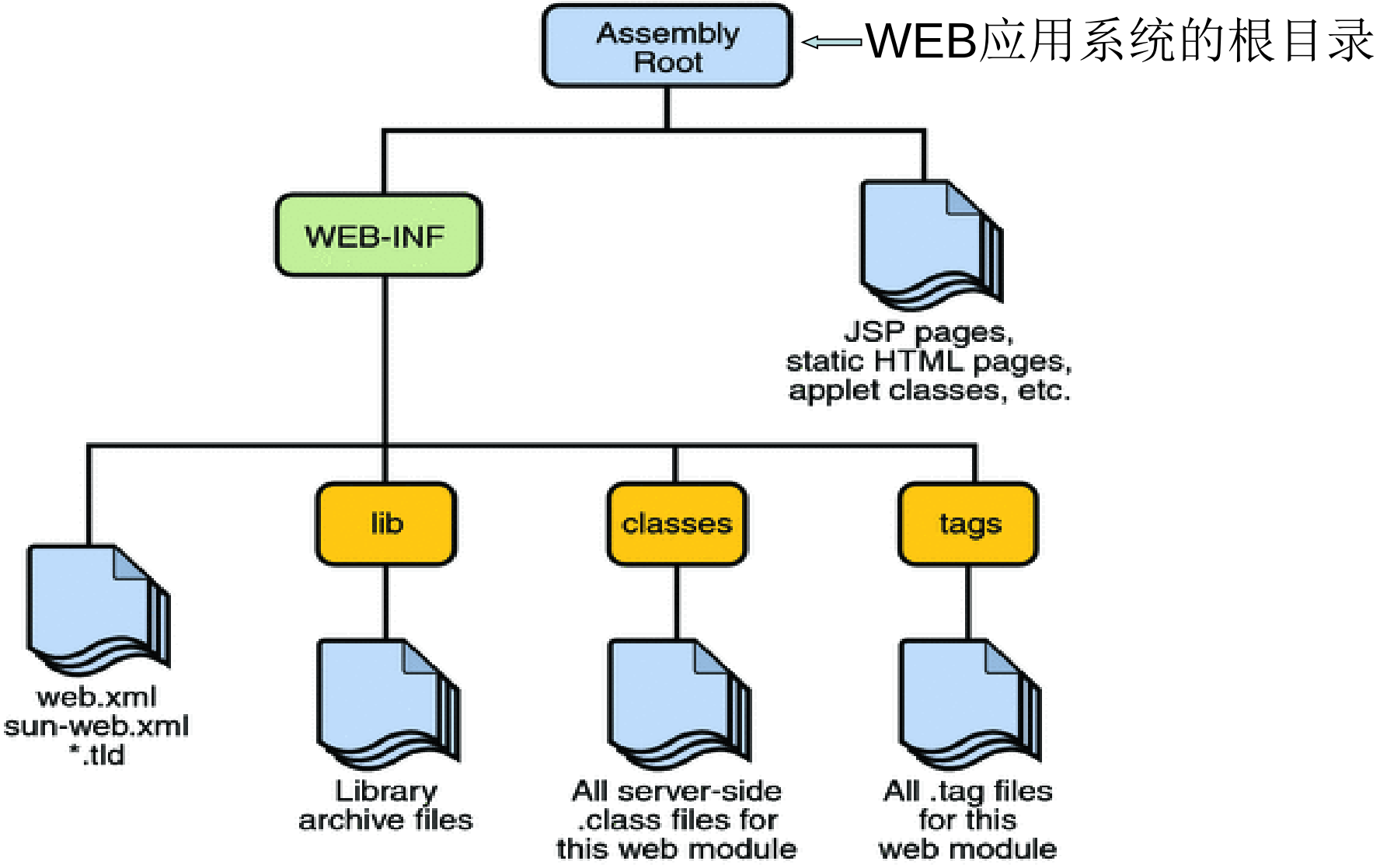


FIGURE 2–3 WebModule Structure

1）WEB-INF

(1)web.xml: WEB应用系统的部署文件

配置Servlet的映射路径等

(2)Tag library descriptor files：标记库描述文件

(3)classes:服务器端类（servlets, utility classes, and JavaBeans Components）

(4) tags:标记文件(标记库的实现）目录

(5)lib:服务器端类引用的类库（JAR)

说明：

(1)web module 如果没有包含servlets, filter, or listener components ,那就不需要WEB应用系统部署文件。(只有JSP 、images、HTML等文件）

(2)在WEB容器运行时，WEB应用程序的类加载器将首先加载classes目录下的，其次才是lib目录下的类。如有同名类，起作用的是classes目录下的类。

(3) WEB-INF目录并不属于WEB应用程序可以访问的上下文路径的一部分，对客户端来说，这个目录是不可见的。

例如：在客户端，无法访问在WEB-INF中的index.htm

http://localhost:8080/Library/WEB-INF/index.htm

但WEB-INF目录下的内容 对于Servlet 是 可见的。

2.4 Web Application Life Cycle

创建、部署、执行**Web Application** 过程：

1)编码WEB组件

2) 编辑WEB应用系统部署文件(web.xml)

3) 编译WEB组件及其它引用类文件。

4)打包WEB应用系统成为一个可部署单元（deployable unit）(可选）

5)部署WEB应用系统到web container

6)通过URL访问WEB应用系统

2.5 web module 的部署

2.5.1 web module 的部署：任何符合Servlet规范的WEB容器中。

1)部署应用系统的整个目录结构

2)部署应用系统的打包文件(war)

(1) Packaging Web Modules

jar命令

(2) Deploying a WAR File

① copy

②部署工具

2.6 case: Library web application

2.6.1 应用系统目录结构 (针对JBoss4.0中间件)

系统的根目录

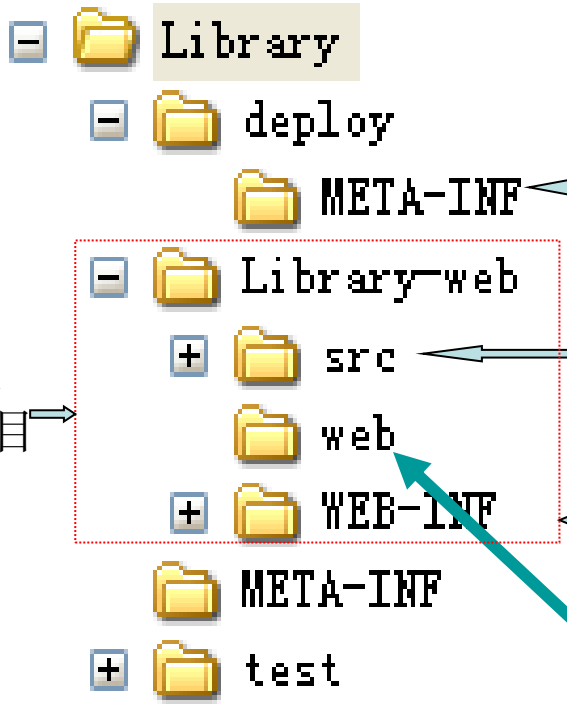
存放WEB部署文件及系统配置文件，由这些部署文件及配置文件产生应用系统的部署文件EAR

WEB应用程序的根目录(web应用程序 的上下文根Context)

系统配置文件application.xml的目录(名字固定)

测试WEB组件的客户端程序(运行在命令行方式）

2.6.2 WEB应用系统目录结构 (针对JBoss4.0中间件)



WEB应用程序目录结构

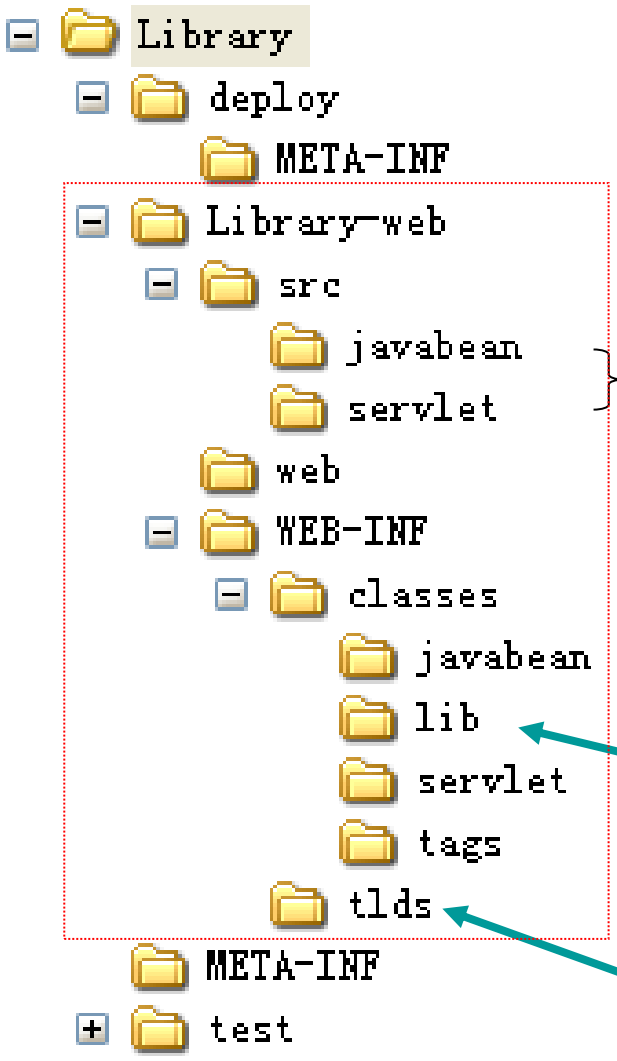
系统配置文件application.xml的目录
(与根目录Library中的META-INF子目录一样)

WEB组件对应的JavaBean、Servlet及其它公用文件的源文件的目录，编译后的class文件放置在WEB-INF\classes目录中

WEB组件对应的配置文件web.xml、jboss-web.xml的目录(名字固定)

WEB组件对应的jsp、html文件的目录

展开:



WEB应用程序目录结构

WEB组件对应的JavaBean、Servlet及其它公用文件的源文件的目录，编译后的class文件放置在WEB-INF\classes目录中

WEB组件对应的JavaBean、Servlet及其它公用文件的class文件的目录

JavaBean、Servlet、JSP等WEB组件 需要引用相关类对应的jar文件：
mssqlserver.jar、msbase.jar、msutil.jar、javaee.jar、servlet.jar、jbossall-client.jar

标签库目录

说明：

1）部署在不同的中间件环境中，应用系统的目录结构和主要配置文件都一样。

2）部署在不同的中间件环境中，个别配置文件会不一样（请参照相关中间件资料)。

3）本课件使用中间件（应用服务器）JBoss4.0