西安电子科技大学

构件与中间件技术

第二部分 Java EE与EJB

§2-2 Java Web工程

徐悦甡

ysxu@xidian.edu.cn

http://web.xidian.edu.cn/ysxu/

西安电子科技大学

目录

Java EE与Web开发

- Java EE制定了一套协议规范，但并不是真正的实现，指导其他厂商应当如何实现企业服务
- Java EE可分为5大类，13个技术规范

分布式系统相关

- **JNDI**：Java Naming and Directory Interface，命名目录接口，把资源从程序中剥离,使得程序与资源耦合降低
- **IDL**： Interface Definition Language，不同系统平台不同语言之间的系统集成协议
- **EJB**：Enterprise Jave Bean, 可以进行远程调用的JavaBean协议
- **RMI**：Remote Method Invocation, Java对象远程调用协议
- **JMS**：Java Message Service, Java消息中间件协议。ActiveMQ是其中一个实现,后发展出AMQP（Advanced Message Queue Protocal, 高级消息队列协议），实现包括RabbitMQ

Java EE与Web开发

docs.oracle.com/javase/tutorial/jndi/overview/index.html

ORACLE Java Documentation

The Java™ Tutorials

Hide TOC

« Previous • Trail • Next » Home Page

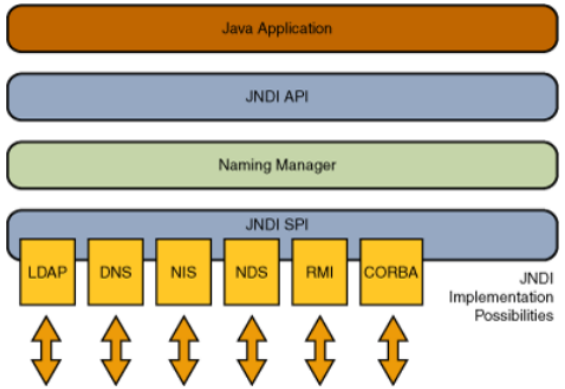
The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases and might use tec
See Java Language Changes for a summary of updated language features in Java SE 9 and subsequent releases.
See JDK Release Notes for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Lesson: Overview of JNDI

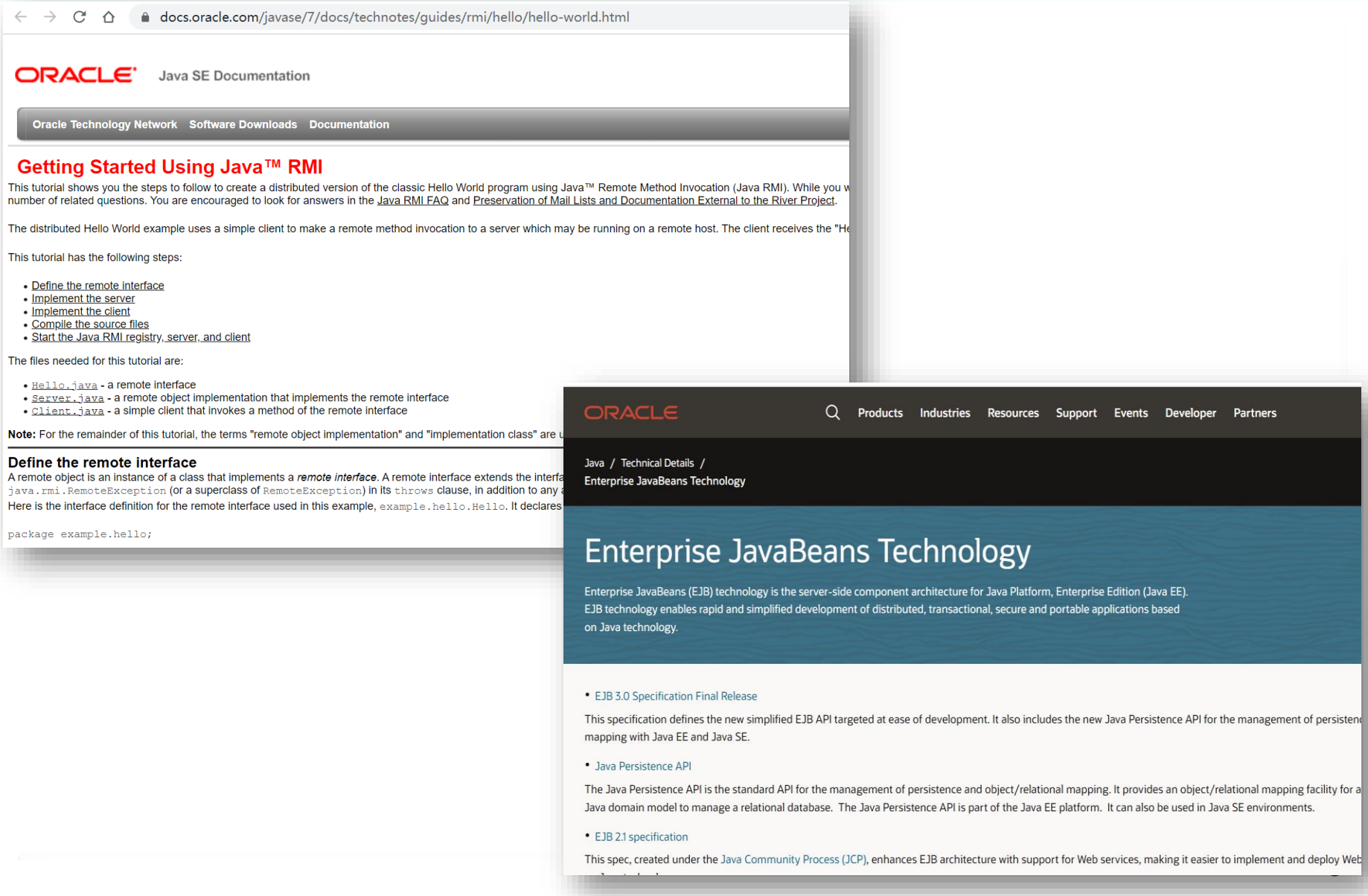
The Java Naming and Directory Interface™ (JNDI) is an application programming interface (API) that provides naming and directory functionality to applications written using the J to be independent of any specific directory service implementation. Thus a variety of directories -new, emerging, and already deployed can be accessed in a common way.

Architecture

The JNDI architecture consists of an API and a service provider interface (SPI). Java applications use the JNDI API to access a variety of naming and directory services. The SPI e services to be plugged in transparently, thereby allowing the Java application using the JNDI API to access their services. See the following figure:



Java EE与Web开发



Java EE与Web开发

ORACLE Java Documentation

Java Software Java SE Downloads Java SE 8 Documentation Search

Getting Started with Java IDL

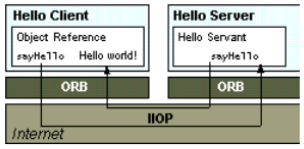
Java IDL is a technology for distributed objects – that is, objects interacting on different platforms across a network. Java IDL enables objects to interact regardless of whether they're written in the Java programming language or another language such as C, C++, COBOL, or others. This is possible because Java IDL is based on the Common Object Request Brokerage Architecture (CORBA), an industry-standard distributed object model. A key feature of CORBA is IDL, a language-neutral Interface Definition Language. Each language that supports CORBA has its own IDL mapping--and as its name implies, Java IDL supports the mapping for Java. To learn more about the IDL-to-Java language mapping, see IDL-to-Java Language Mapping.

To support interaction between objects in separate programs, Java IDL provides an Object Request Broker, or ORB. The ORB is a class library that enables low-level communication between Java IDL applications and other CORBA-compliant applications.

This tutorial teaches the basic tasks needed to build a CORBA distributed application using Java IDL. You will build the classic "Hello World" program as a distributed application. The Hello World program has a single operation that returns a string to be printed.

Any relationship between distributed objects has two sides: the client and the server. The server provides a remote interface, and the client calls a remote interface. These relationships are common to most distributed object standards, including Java Remote Method Invocation (RMI, RMI-IIOP) and CORBA. Note that in this context, the terms client and server define object-level rather than application-level interaction--any application could be a server for some objects and a client of others. In fact, a single object could be the client of an interface provided by a remote object and at the same time implement an interface to be called remotely by other objects.

This figure shows how a one-method distributed object is shared between a CORBA client and server to implement the classic "Hello World" application.



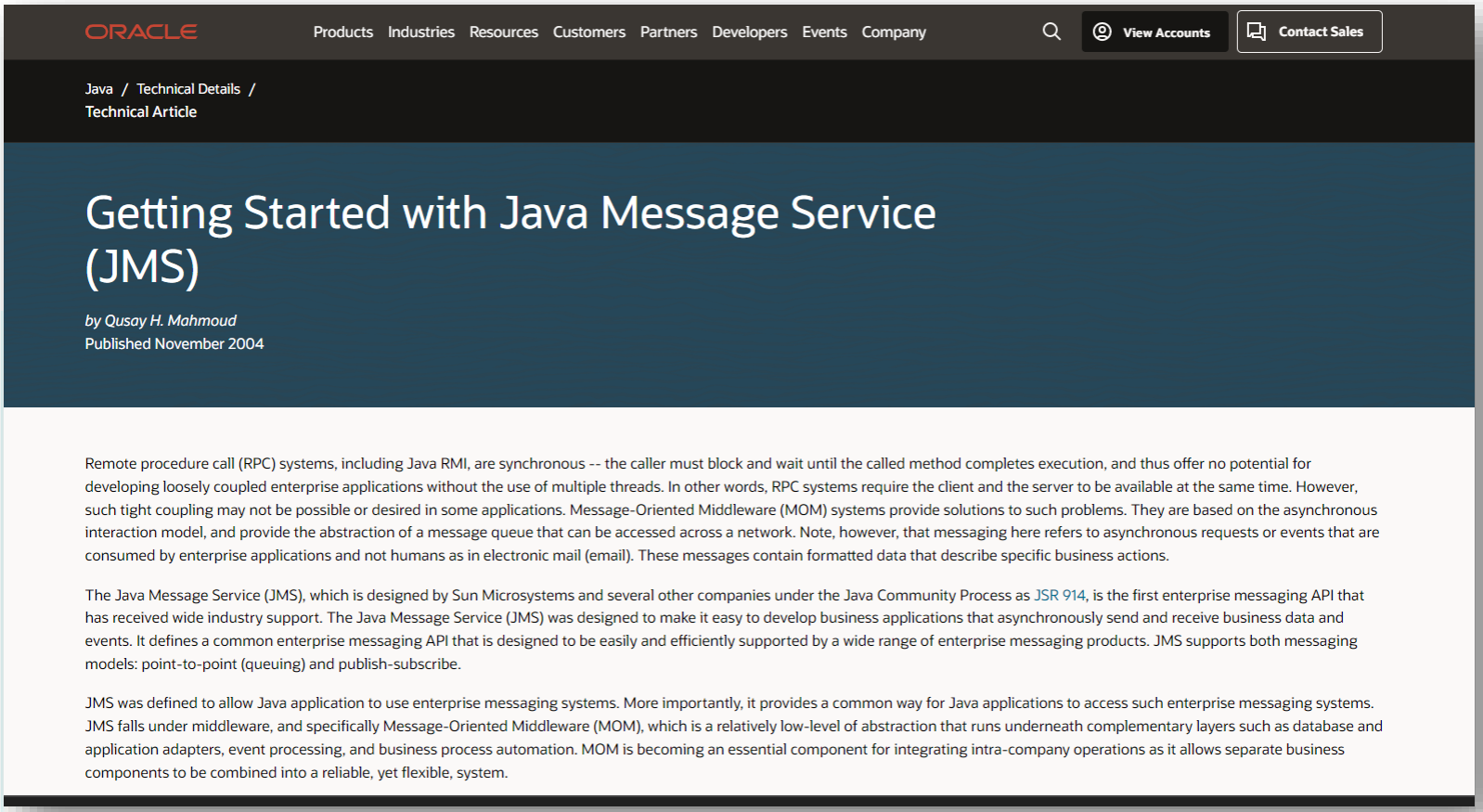
A one-method distributed object shared between a CORBA client and server.

On the client side, the application includes a reference for the remote object. The object reference has a stub method, which is a stand-in for the method being called remotely. The stub is actually wired into the ORB, so that calling it invokes the ORB's connection capabilities, which forwards the invocation to the server.

On the server side, the ORB uses skeleton code to translate the remote invocation into a method call on the local object. The skeleton translates the call and any parameters to their implementation-specific format and calls the method being invoked. When the method returns, the skeleton code transforms results or errors, and sends them back to the client via the ORBs.

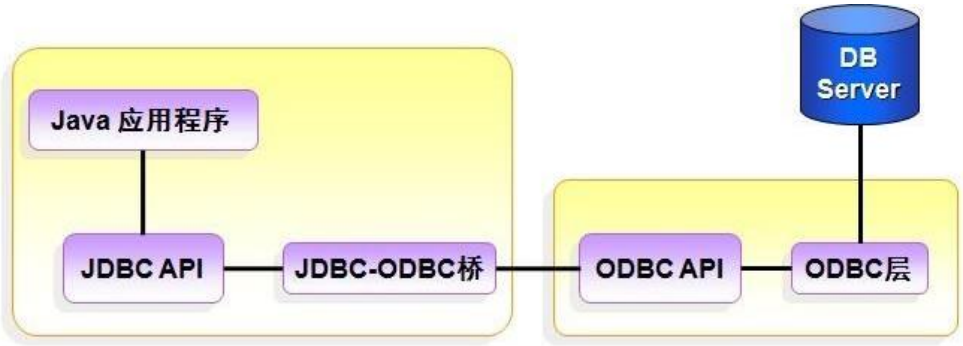
Between the ORBs, communication proceeds by means of a shared protocol, IIOP--the Internet Inter-ORB Protocol. IIOP, which is based on the standard TCP/IP internet protocol, defines how CORBA-compliant ORBs pass information back and forth. Like CORBA and IDL, the IIOP standard is defined by OMG, the Object Management Group.

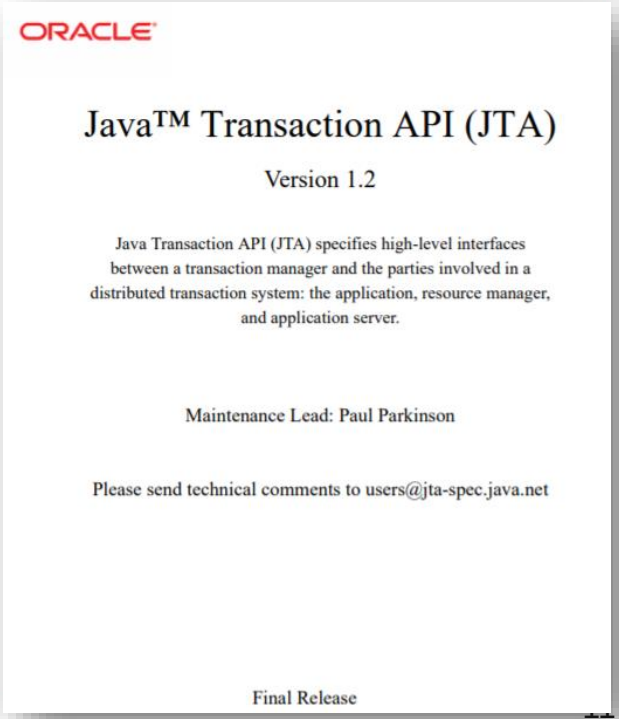
Java EE与Web开发



Java EE与Web开发

- **数据库开发相关：**
  - **JDBC**：数据库连接协议
  - **JTA**（Java Transaction API）：事务管理体系结构协议，主要规范进行事务管理必须的角色以及之间的关系
  - **JTS**：事务管理服务协议，为JTA更具体化的协议，规范各种角色之间的具体交互方法)




ORACLE

Java™ Transaction API (JTA)

Version 1.2

Java Transaction API (JTA) specifies high-level interfaces between a transaction manager and the parties involved in a distributed transaction system: the application, resource manager, and application server.

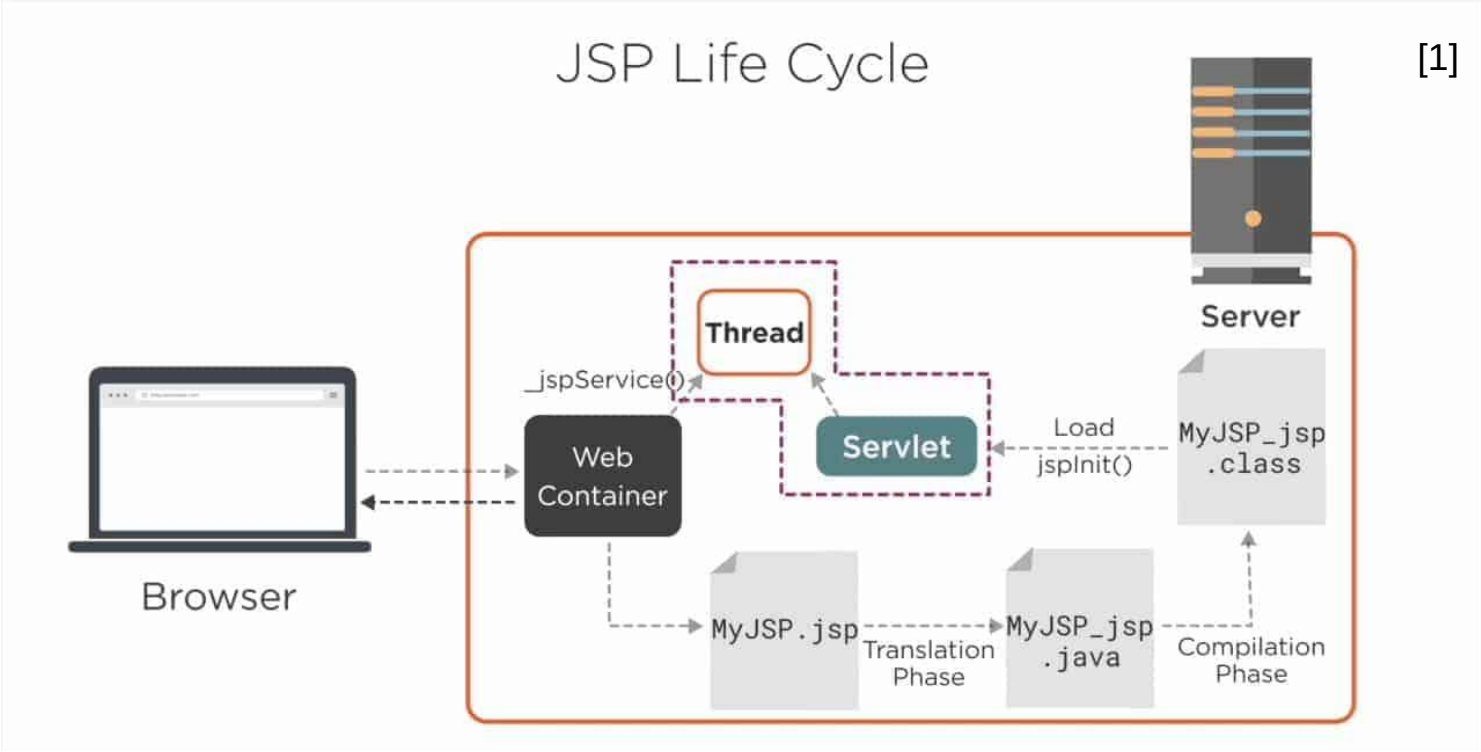
Maintenance Lead: Paul Parkinson

Please send technical comments to users@jta-spec.java.net

Final Release

Java EE与Web开发

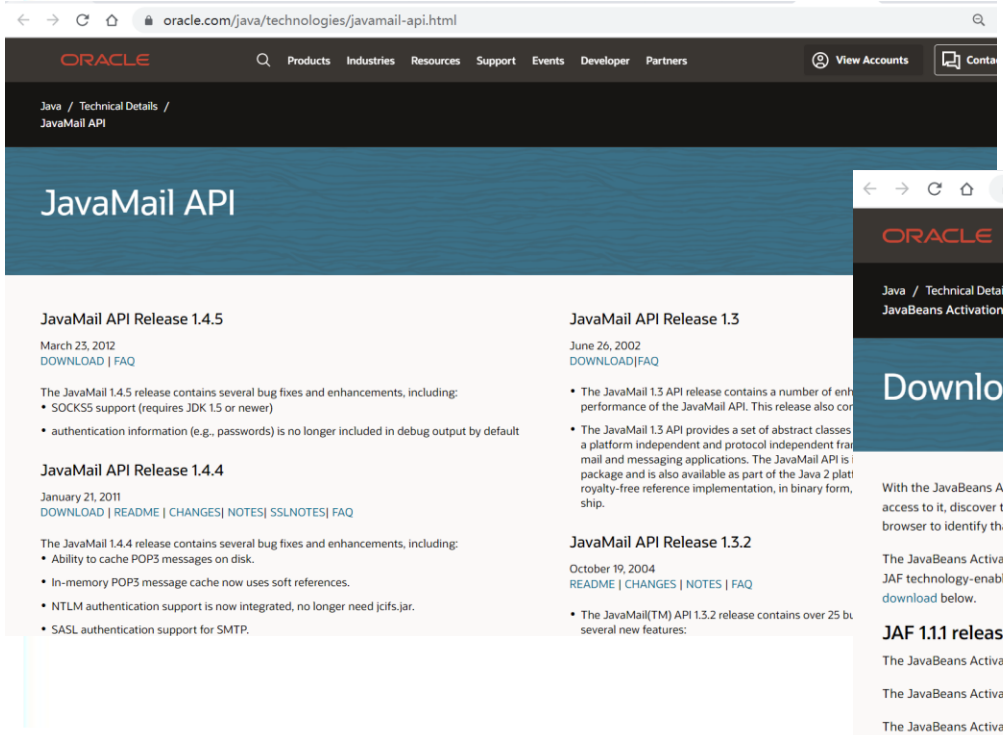
- **Web开发相关：**
  - **JSP**，Java Server Page，可在HTML中嵌入Java的协议
  - **Servlet**，Server Applet，服务端应用

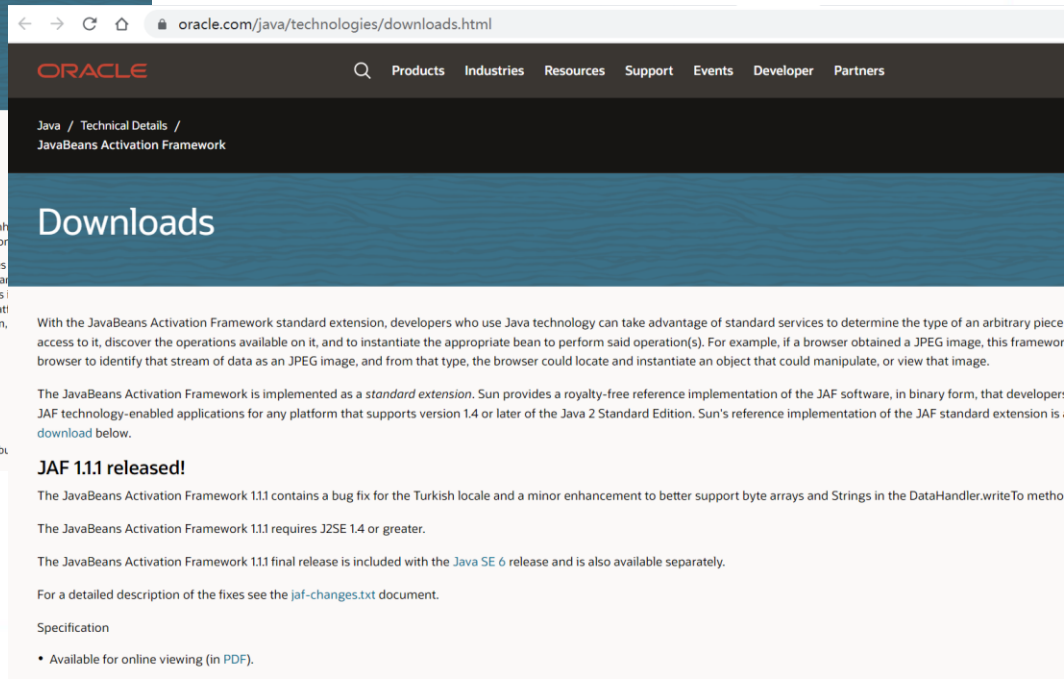


[1] https://leadsfac.com/diseno-paginas-web/

Java EE与Web开发

- **邮件相关：**
  - **Java Mail**：邮件服务协议
  - **JAF**(JavaBeans Activation Framework)： 邮件MIME（Multipurpose Internet Mail Extensions ）数据处理框架





Java EE与Web开发

- **公共相关：**
  - XML 可扩展标记语言，多用于配置文件

嵌套XML元素

- 示例

```
<教职工>
    <姓名>XXX</姓名>
    <职称>讲师</职称>
    <工资，货币=“人民币”>8000</工资>
    <学院>数学与统计学院</学院>
</教职工>
```

目录

Web与Web应用

Web

- Web（World Wide Web）即万维网，是基于超文本和HTTP的、全球性的、动态交互的、跨平台的分布式图形信息系统，为浏览者在Internet上查找和浏览信息提供了图形化的、易于访问的直观界面。
- 其中的文档及超级链接将Internet上的信息节点组织成一个互为关联的网状结构，故称为**万维网**。

Internet上供外界访问的Web资源可分为：

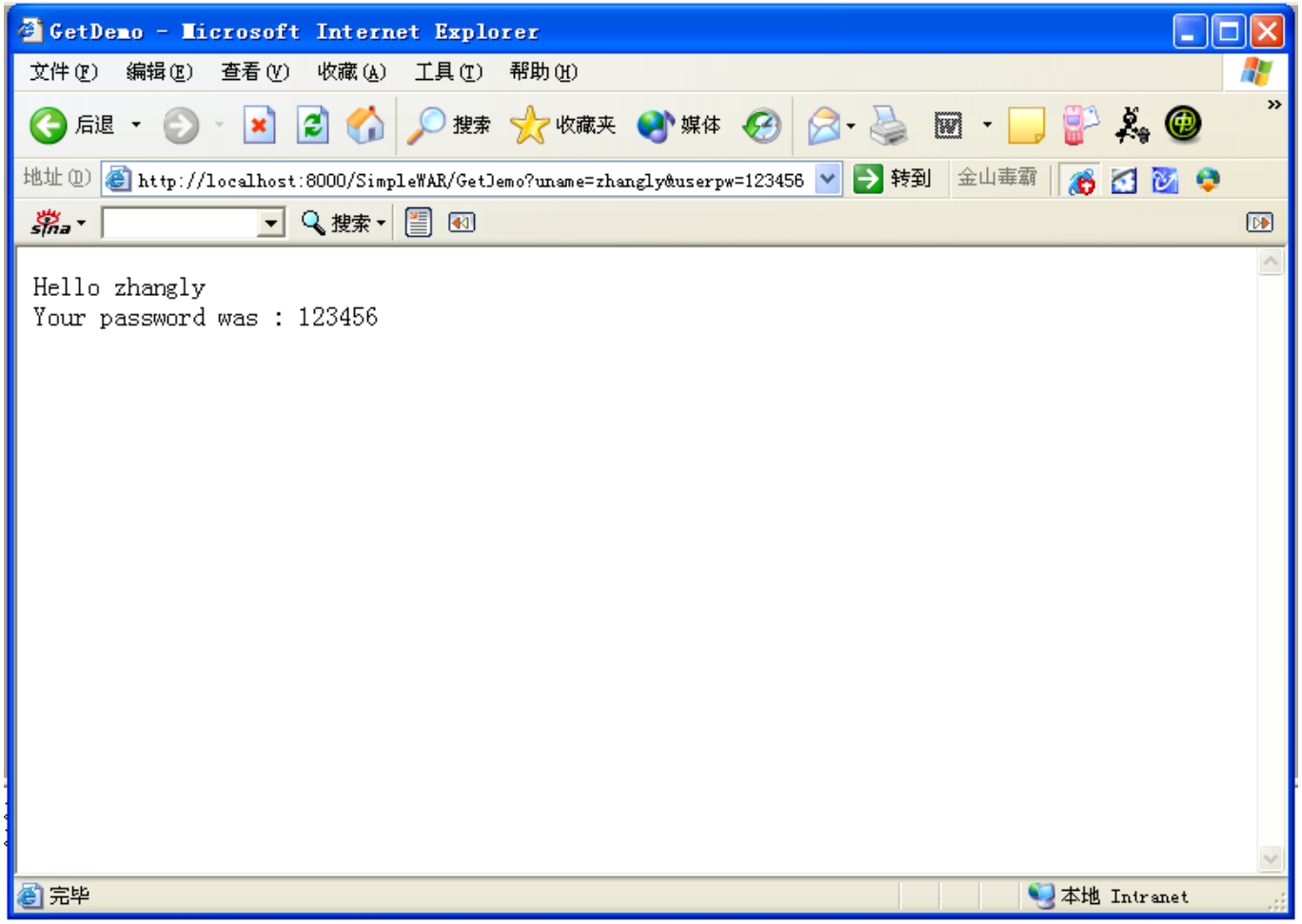
- **静态Web资源**（如HTML 页面）：指Web页面中供人们浏览的数据始终是不变。
- **动态Web资源**：指Web页面中供人们浏览的数据是由程序产生的，不同时间点访问Web页面看到的内容各不相同。

Web资源开发技术

- 静态Web资源开发技术：HTML
- 常用动态Web资源开发技术：JSP/Servlet、ASP、PHP等

Web与Web应用

- 静态网页

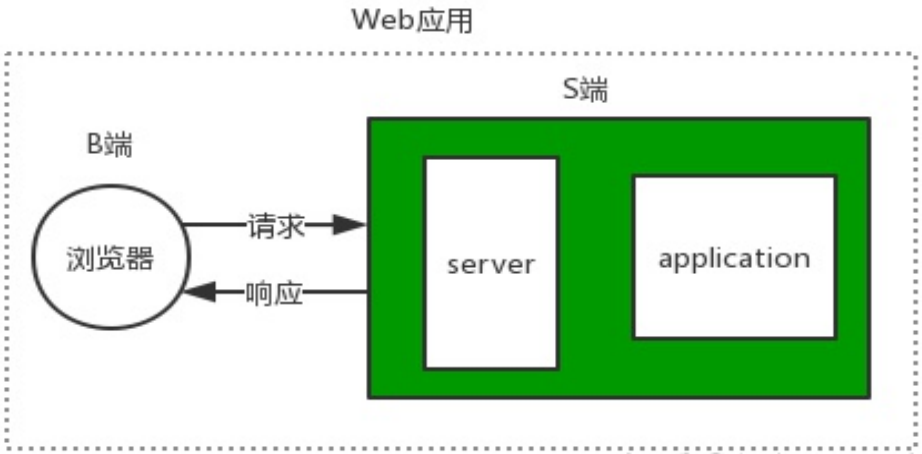


Web与Web应用

Web应用（程序）

- 供浏览器访问的程序，通常也简称为Web应用。
- 例如有a.html 、b.html.....多个Web资源，这多个Web资源用于对外提供服务，此时应把这多个Web资源放在一个目录中，以组成一个**Web应用**（程序）
- 一个Web应用由多个**静态Web资源**和**动态Web资源**组成，如：html、css、js文件，JSP文件、Java程序、支持jar包、配置文件等等。

  - 举例：小说阅读器，Web视频播放器，Web邮箱，网络地图，管理信息系统，网络游戏，学校的教务管理系统；
  - 说明：Web应用不是指一个或者几个网页，而是资源的集合。



Web与Web应用

- **Web Application**: A specific functionality-oriented component that utilizes Web technologies to deliver information and services to users or other applications/information systems.
- **Web (Based) Information System**: An information system that utilizes Web technologies to deliver information and services to users or other information systems/applications.
- **Web (World Wide Web):** The hypermedia infrastructure based on the Internet to deliver information and services to globally distributed users and systems.

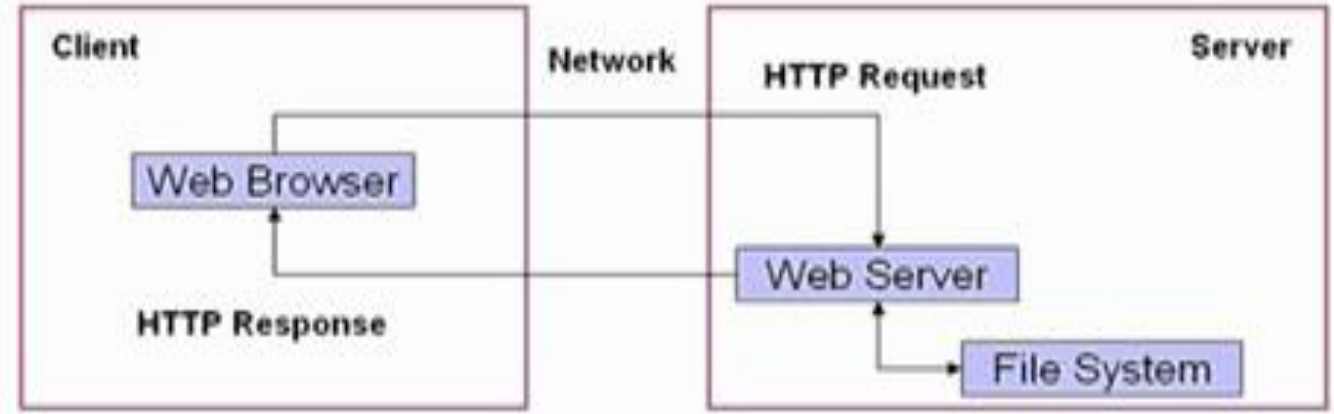
参照自Prof. Guangzhi Zheng (Georgia State University, USA)于200年编写的章节
A Historical Perspective of Web Engineering

https://www.igi-global.com/chapter/historical-perspective-web-engineering/17673

Web与Web应用

静态Web

- 在静态Web程序中，客户端使用**Web浏览器**（Chrome、Edge、Safari等）经过计算机网络（Computer Network）连接到服务器上，使用HTTP协议发起一个请求（Request），告诉Web服务器需要得到哪个页面，Web服务器根据接收到的请求，从文件系统（存放了所有静态页面的磁盘）取出内容。
- Web服务器返回给客户端，客户端接收到内容之后经过浏览器渲染解析，得到显示的效果。



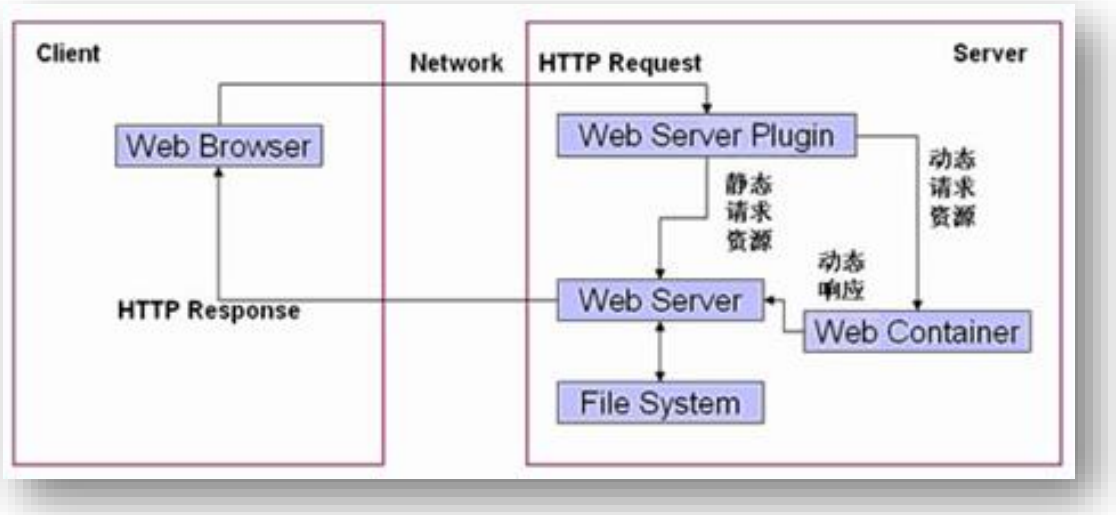
Web与Web应用

- 为了可以让静态的Web具有动态性，可以使用JavaScript等完成页面上的动态显示效果；
- 但是这些效果均借助于浏览器展现给用户，服务器上本身并没有变化。
  - JavaScript
  - VBScript

Web与Web应用

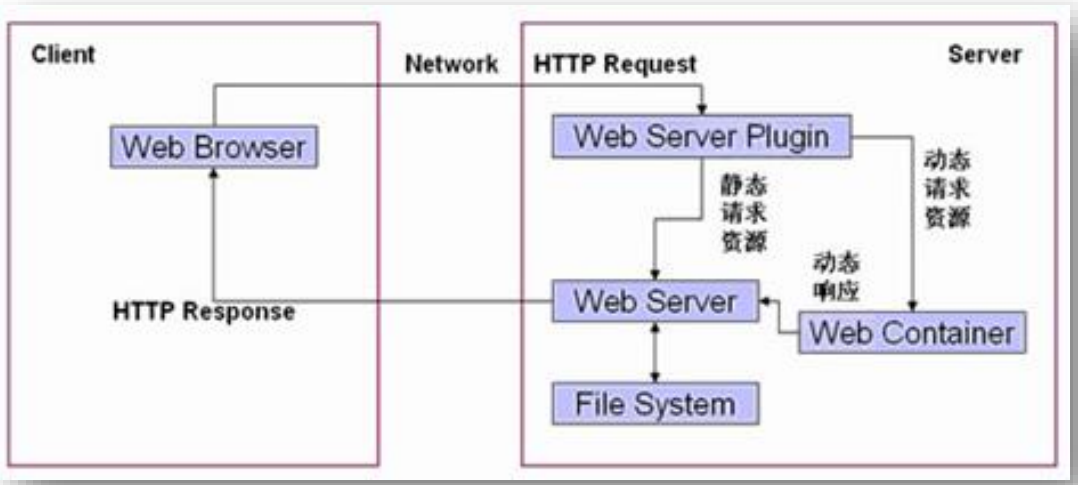
动态Web

- Web的页面的内容可以动态更新
- 所有请求都先经过一个Web Server Plugin（服务器插件）来处理，此插件用于区分是请求的是静态资源(*.htm, *html)还是动态资源。



Web与Web应用

- 若所请求为静态资源(*.htm，*html)，则将请求直接转交给Web服务器，Web服务器从文件系统中取出内容，发送回客户端浏览器进行解析执行。
- 若所请求为动态资源（*.jsp、*.asp/*.aspx、*.php），则先将请求转交给**Web Container**（Web容器），在Web Container中连接数据库，从数据库中取出数据等一系列操作后动态组装页面的展示内容，并交给Web服务器，
- Web服务器将内容发送回客户端浏览器进行解析执行。

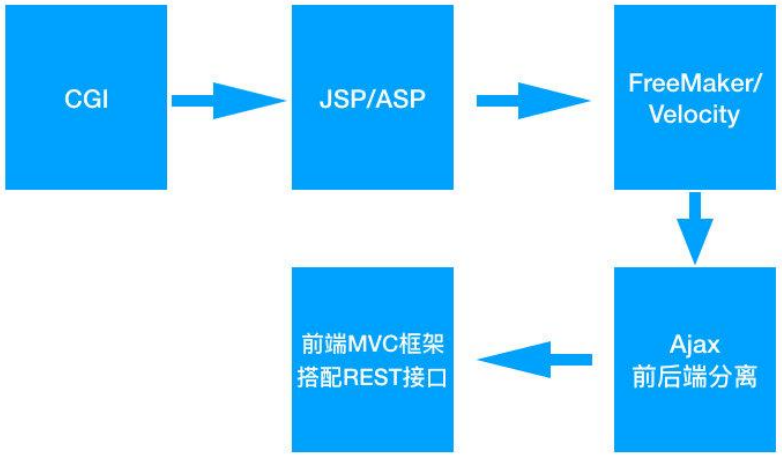


目录

Web开发技术

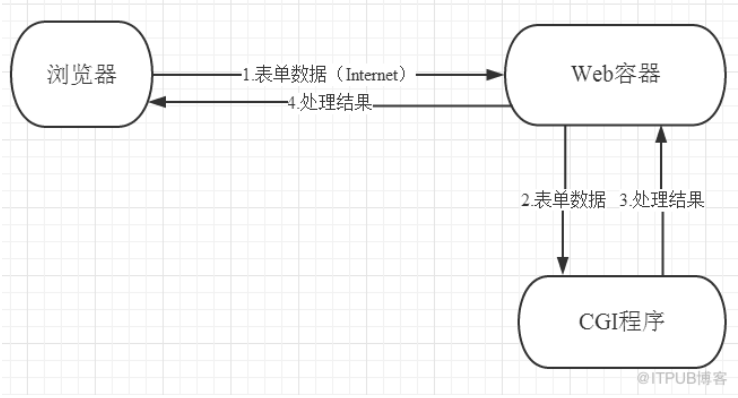
动态Web应用的实现技术

- CGI（Common Gateway Interface，公共网关接口）
- Microsoft ASP、ASP.NET
- PHP
- JAVA Servlet/JSP
- Ajax
- 前端MVC框架(Spring)+RESTful API



Web开发技术

- CGI（公共网关接口）
  - 外部应用程序（CGI程序）与Web服务器之间的接口标准，是在CGI程序和Web服务器之间传递信息的标准
  - 独立于任何语言与操作系统
  - 大部分CGI用来解释和处理来自表单的输入信息，并在服务器上做相应的处理，或将相应的信息反馈给浏览器
- CGI工作过程
  - 通过Internet把用户请求送到Web服务器；
  - Web服务器接收用户请求并交给CGI程序处理；
  - CGI程序把处理结果传递给Web服务器；
  - Web服务器把结果送回用户。



CGI

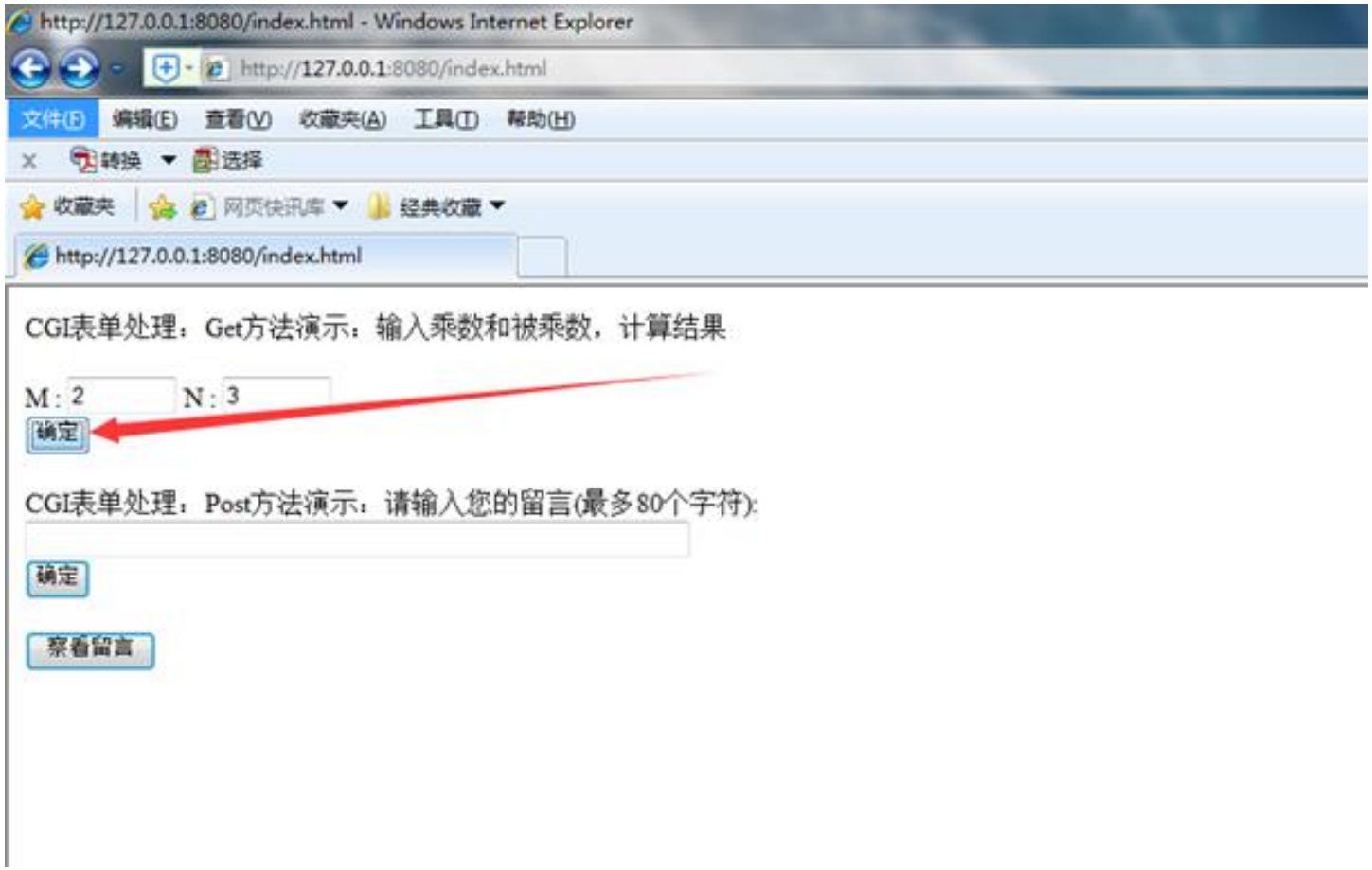
Web开发技术

- CGI实例
  - 使用C语言编写CGI程序，index.html
  - HTML表单

```
<head>
  <meta http-equiv="Content-Type"
content="text/html;charset=utf-8" />
  <body>
    <form action="/cgi-bin/mult.cgi">
       <p>CGI表单处理：Get方法演示：输入乘数和被乘数，计算
结果</p>
          M : <input name="m" size="5">
          N : <input name="n" size="5">
       <br><input type="submit" value="确定
"></input></br>
    </form>

    <form action="/cgi-bin/collect.cgi" method="POST" >
       <p>CGI表单处理：Post方法演示：请输入您的留言(最多80
个字符):
       <br><input name="data" size="60"
maxlength="80"></br>
       <input type="SUBMIT" value="确定">
    </form>
```

```
<form action="/cgi-bin/viewdata.cgi">
       <p><input type="SUBMIT" value="察看
留言">
    </form>
  </body>
</head>
```

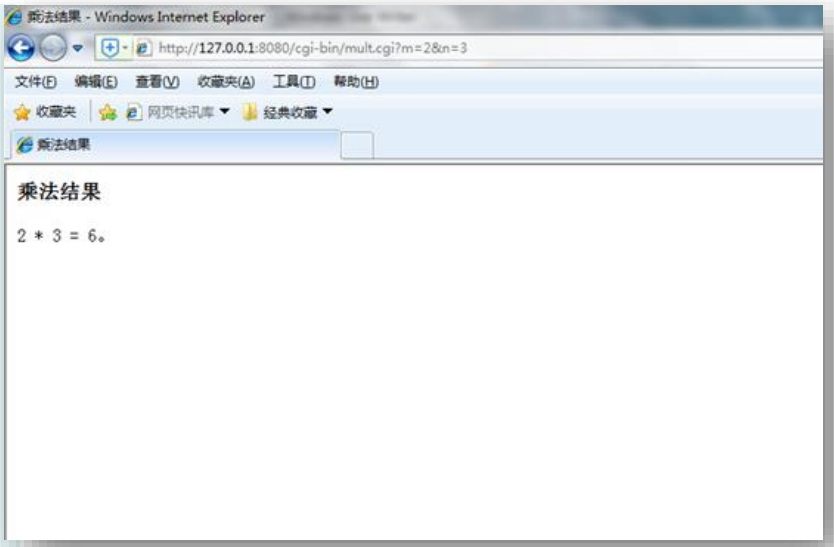
Web开发技术


http://127.0.0.1:8080/index.html - Windows Internet Explorer
http://127.0.0.1:8080/index.html
文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)
转换 选择
收藏夹 网页快讯库 经典收藏
http://127.0.0.1:8080/index.html
CGI表单处理：Get方法演示：输入乘数和被乘数，计算结果
M : 2 N : 3
确定
CGI表单处理：Post方法演示：请输入您的留言(最多80个字符):
确定
察看留言

Web开发技术

- CGI实例
  - 使用C语言编写CGI程序
  - mult.cgi

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    char *data;
    long m, n;
    printf("%s", "Content-Type:text/html\n\n");
    printf("<html>");
    printf("<head><title>乘法结果</title>");
    printf("<h3>乘法结果</h3> ");
    printf("</head><body>");
    data = getenv("QUERY_STRING");
    if (data == NULL)
        printf("<p>错误!数据没有被输入或者数据传输有问题");
    else if (sscanf(data, "m=%ld&n=%ld", &m, &n) != 2)
        printf("<p>错误!输入数据非法。表单中输入的必须是数字。");
    else
        printf("<p>%ld * %ld = %ld。", m, n, m * n);
    printf("</body></html>");
    return 0;
}
```



Web开发技术

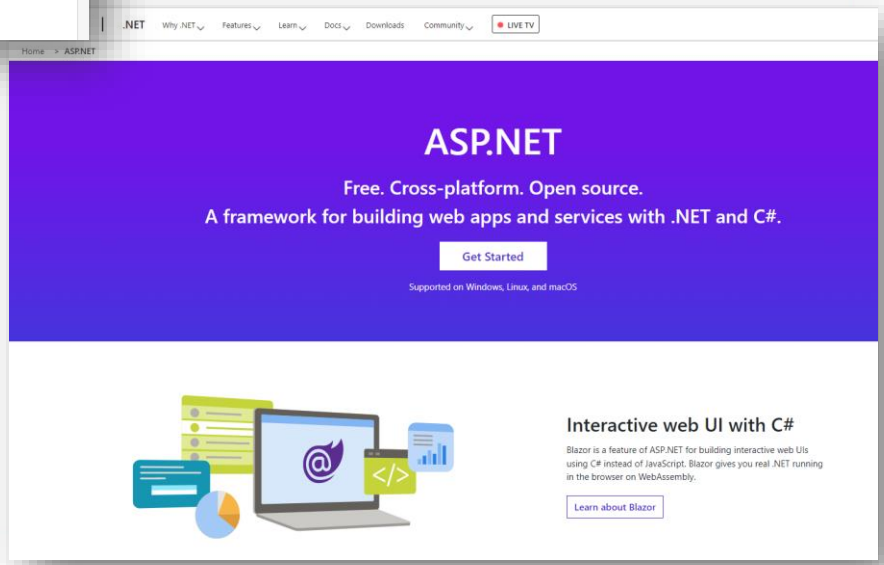
- 动态Web应用的实现技术
  - Microsoft ASP、ASP.NET
    - ASP(Active Server Pages)：HTML+ VBScript，ASP只能运行在Windows操作系统上，经典开发平台：Windows+IIS+SQL Server/Access
    - ASP.NET在性能有了很大的改善，而且开发迅速，但是依然受限于平台，C#语言
      - ASP.NET extends the .NET developer platform with tools and libraries specifically for building Web apps.

Web开发技术

源代码:

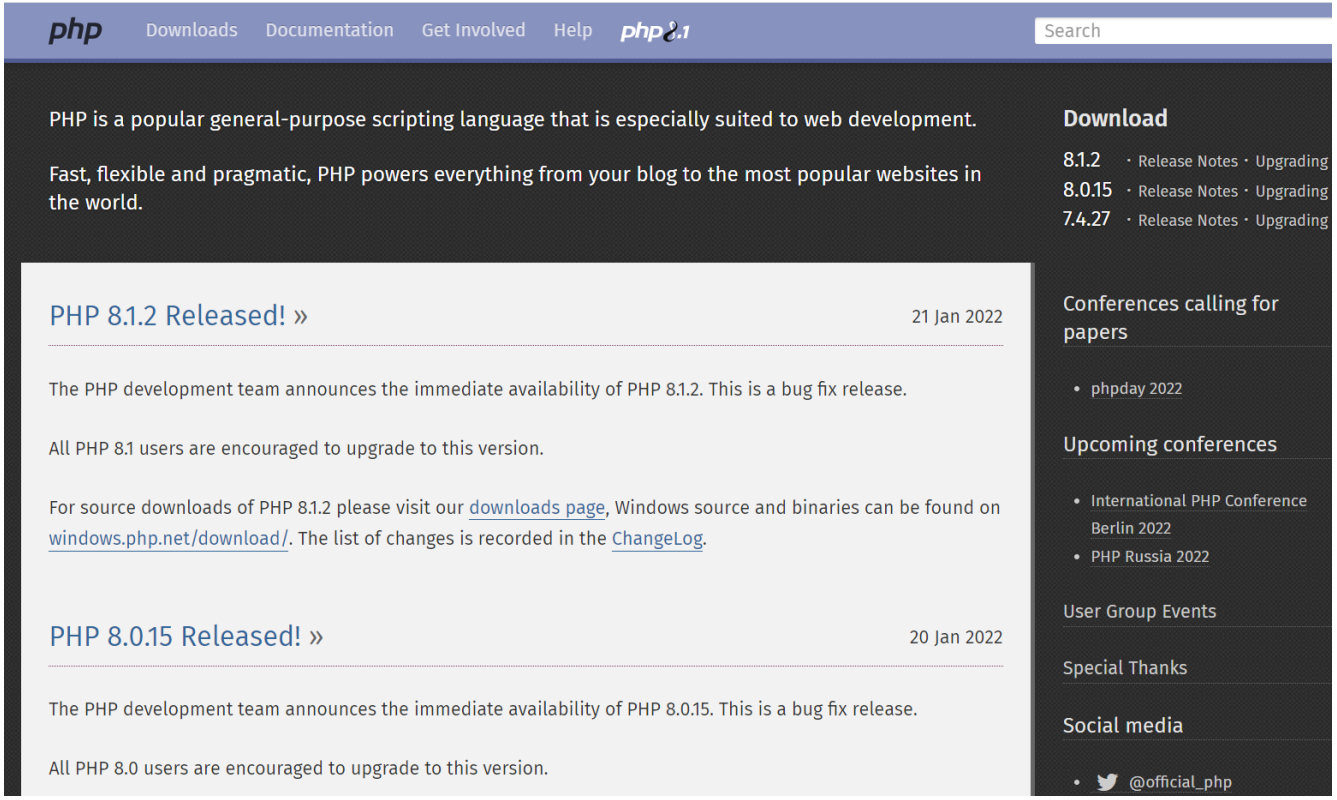
点击运行 》

```
<!DOCTYPE html>
<html>
<body>

<script type="text/vbscript">
document.write("This is my first VBScript!")
</script>

</body>
</html>
```



Web开发技术

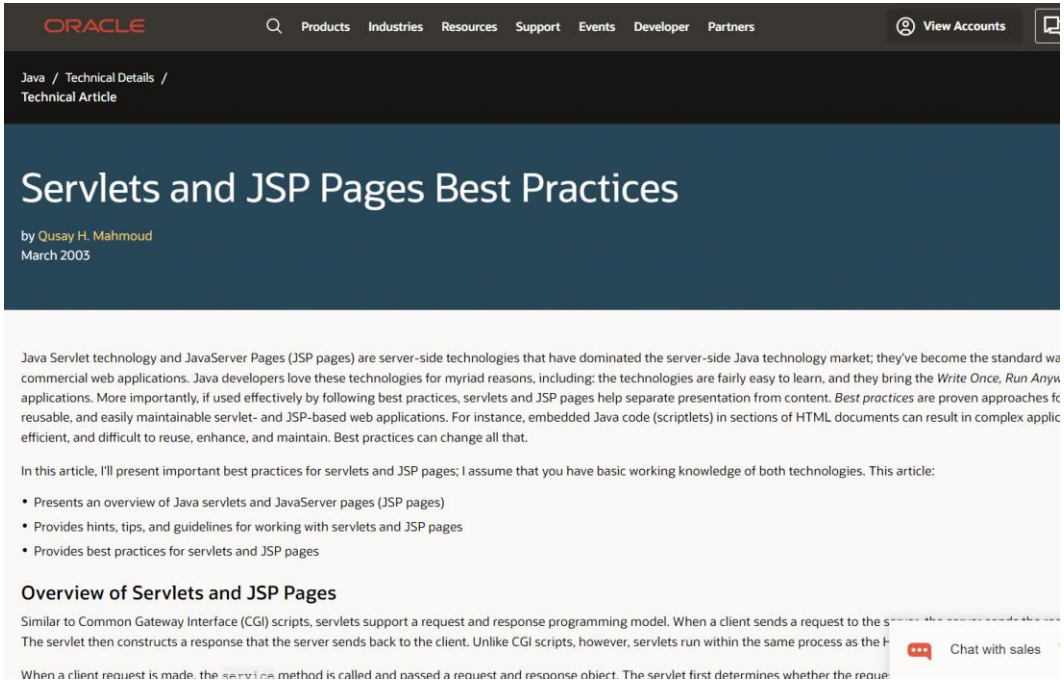
- 动态Web应用的实现技术
  - PHP(Hypertext Preprocessor)
    - 开发速度快，跨平台(操作系统)，代码简单



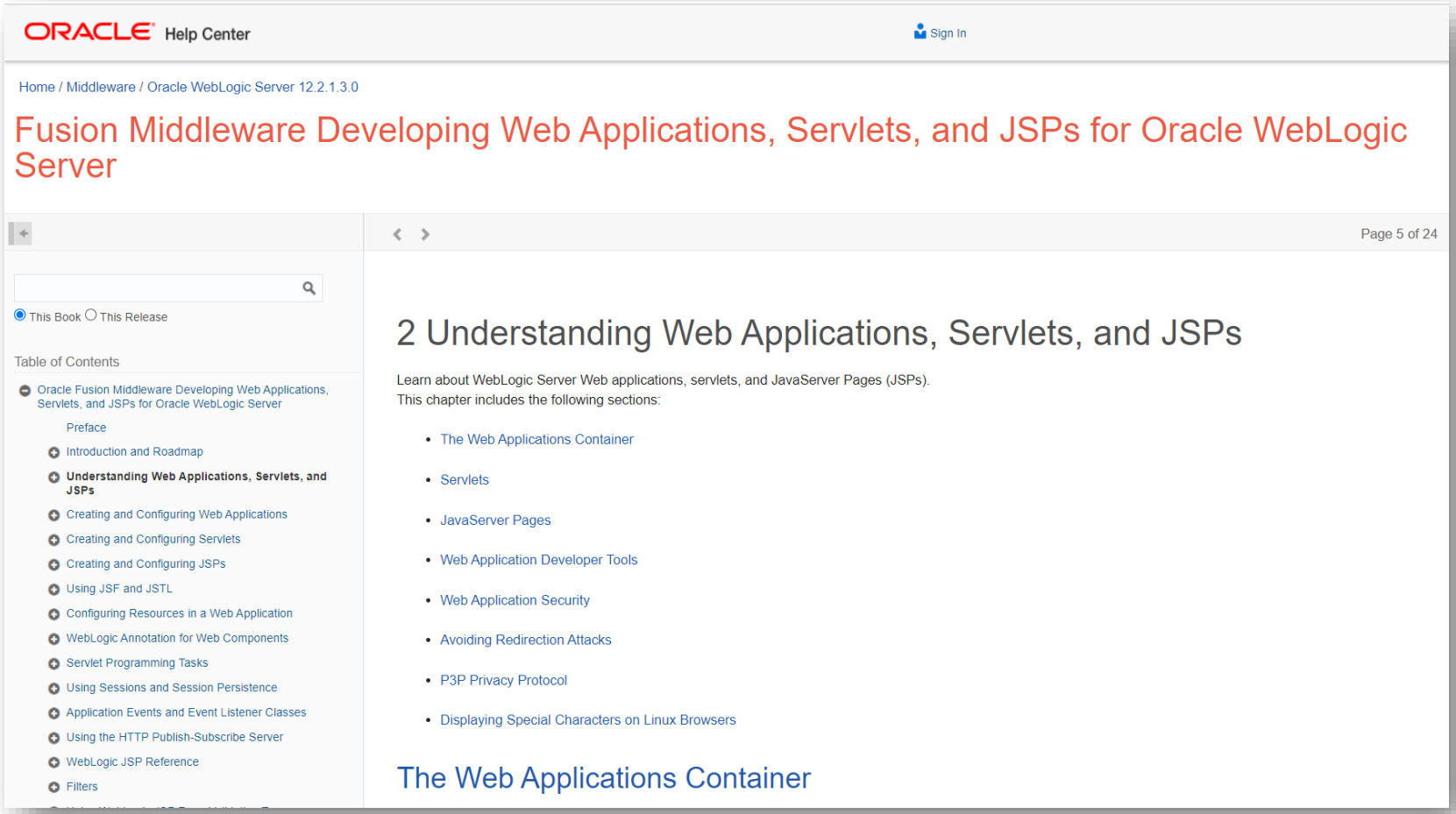
Web开发技术

Servlet/JSP

- SUN公司/Oracle公司推出的B/S架构的实现语言，基于JAVA语言
- 跨平台、多线程、性能非常高。
- SUN公司最早推出了Servlet程序，所有程序采用JAVA代码+HTML的方式编写的，即使用JAVA输出语句，一行一行输出HTML代码
- 之后，SUN公司受到了ASP的启发，发展出了JSP(Java Server Page)



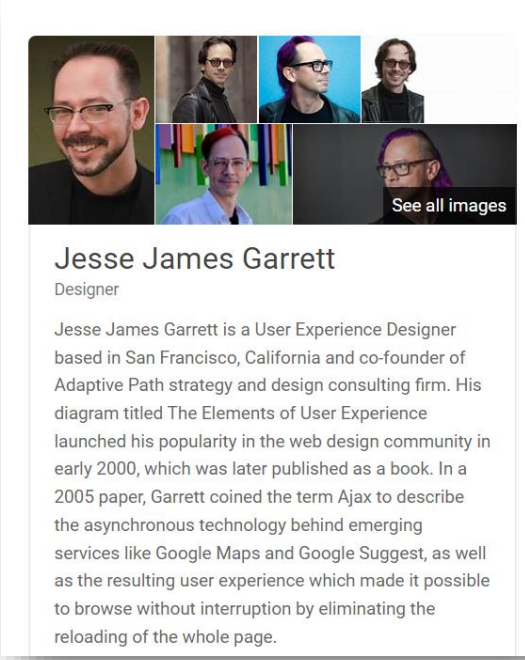
Web开发技术


ORACLE Help Center
Sign In
Home / Middleware / Oracle WebLogic Server 12.2.1.3.0
Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server
Page 5 of 24
This Book
This Release
Table of Contents
Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server
Preface
Introduction and Roadmap
Understanding Web Applications, Servlets, and JSPs
Creating and Configuring Web Applications
Creating and Configuring Servlets
Creating and Configuring JSPs
Using JSF and JSTL
Configuring Resources in a Web Application
WebLogic Annotation for Web Components
Servlet Programming Tasks
Using Sessions and Session Persistence
Application Events and Event Listener Classes
Using the HTTP Publish-Subscribe Server
WebLogic JSP Reference
Filters
2 Understanding Web Applications, Servlets, and JSPs
Learn about WebLogic Server Web applications, servlets, and JavaServer Pages (JSPs).
This chapter includes the following sections:
The Web Applications Container
Servlets
JavaServer Pages
Web Application Developer Tools
Web Application Security
Avoiding Redirection Attacks
P3P Privacy Protocol
Displaying Special Characters on Linux Browsers
The Web Applications Container

Web开发技术

Ajax

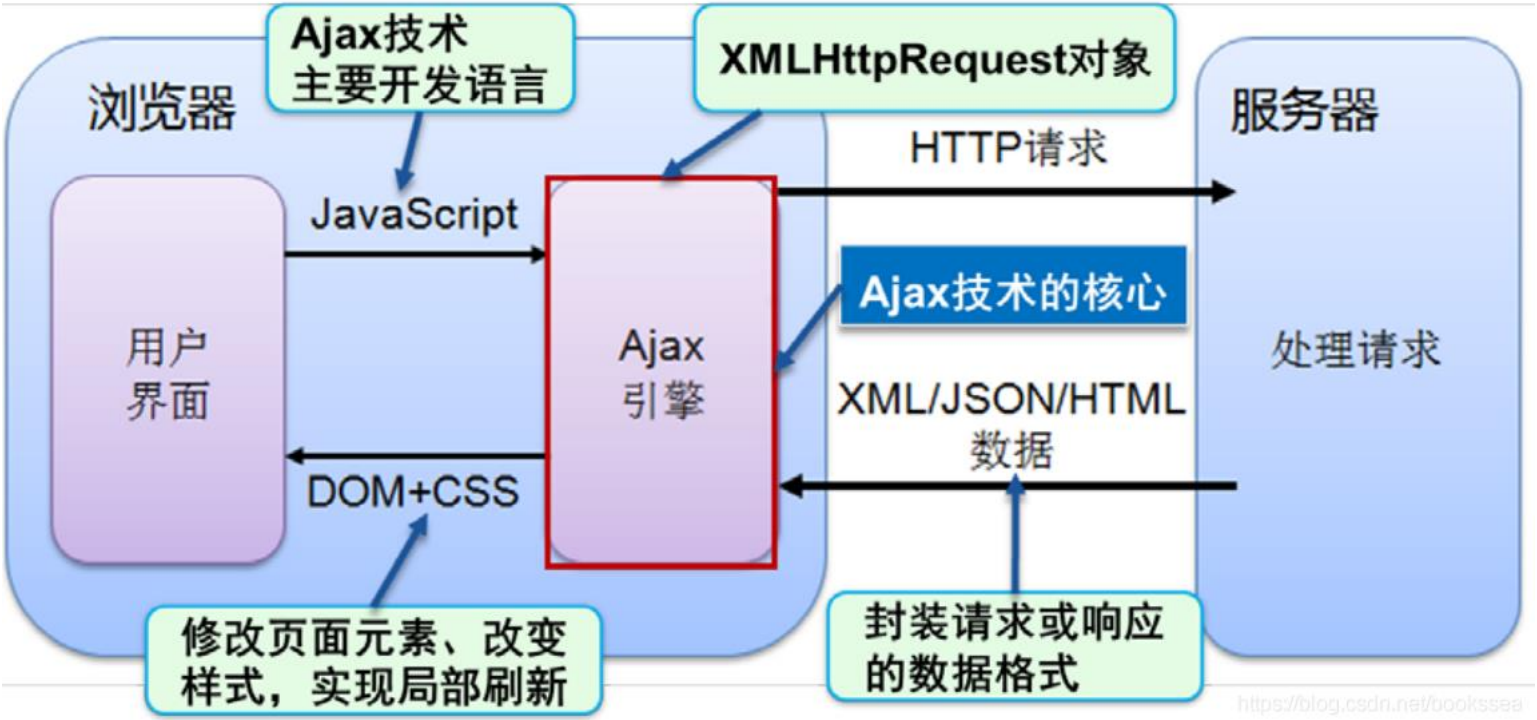
- Asynchronous Javascript And XML
- 1999年，微软公司发布 IE 浏览器5.0版，第一次引入新功能：允许 JavaScript 脚本向服务器发起 HTTP 请求。
- 2004年 Gmail 发布和2005年 Google Map 发布，成功应用这项功能
- 2005年，Jesse James Garrett提出Ajax
- Ajax 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的Web应用程序的技术
- Ajax 通过 JavaScript 的异步通信（XMLHttpRequest），从服务器获取 XML 文档从中提取数据，再更新当前网页的对应部分，而不用刷新整个网页。当前的数据格式已改变为JSON为主
- Ajax是一项独立于Web服务器的浏览器技术，与ASP.net、PHP、JSP不同。



Web开发技术

Ajax工作流程

Ajax 工作流程



Web开发技术

- 传统Web技术与Ajax对比

差异	方式	说　明
发送请求方式不同	传统Web	提交表单方式发送请求
	Ajax技术	异步引擎对象发送请求
服务器响应不同	传统Web	响应内容是一个完整页面
	Ajax技术	响应内容只是需要的数据
客户端处理方式不同	传统Web	需等待服务器响应完成并重新加载整个页面后用户才能进行操作
	Ajax技术	可以动态更新页面中的部分内容 用户不需要等待请求的响应

目录

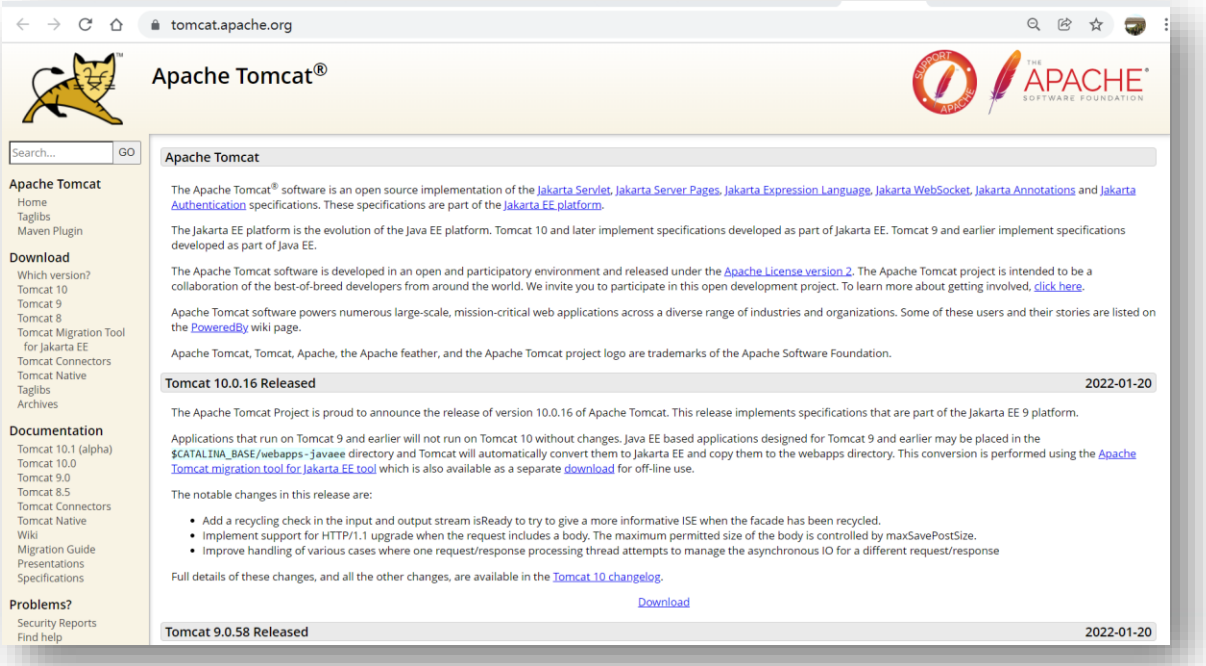
Web服务器与应用服务器

- 什么是**Web服务器**（Web server）？
  - 驻留于因特网上的一类程序，可以向发出请求的浏览器提供文档。
  - 可以接收浏览器等Web客户端的请求并返回响应，也可以放置网站文件以浏览，或放置数据文件以下载
- Web服务器如何工作?
  - 当Web浏览器（客户端）连到服务器上并请求文件时，服务器将处理该请求并将文件反馈到该浏览器上，附带的信息告诉浏览器如何查看该文件（即文件类型）。
  - 服务器通常使用HTTP（超文本传输协议）与客户机浏览器完成请求响应，故也称为HTTP服务器。
  - Web服务器是一种被动程序，只有当互联网上运行的、其他计算机中的浏览器发出请求时，Web服务器才会响应。

Web服务器与应用服务器

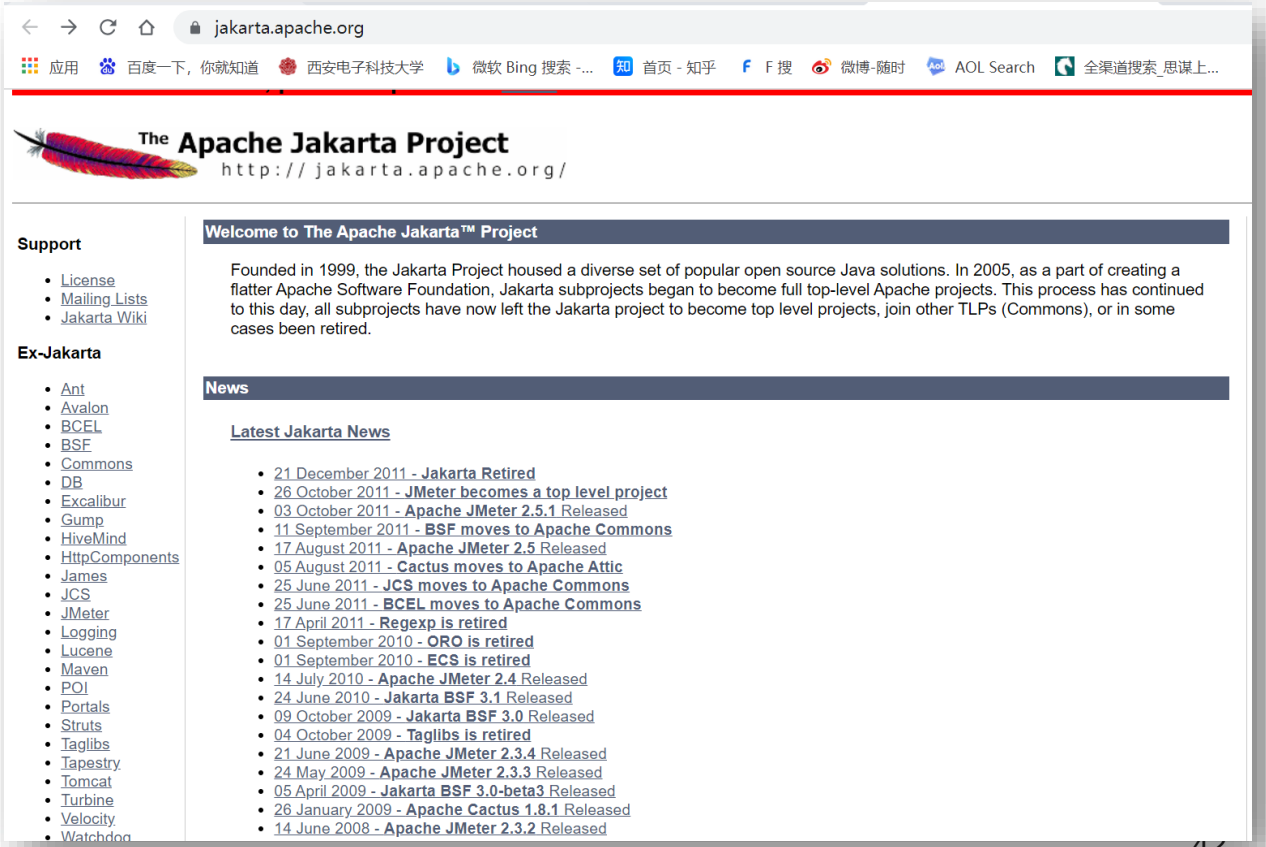
Tomcat

- Tomcat是Apache 软件基金会（Apache Software Foundation）的Jakarta项目中的一个核心项目，由Apache、Sun 和其他一些公司及社区共同开发而成。
- Tomcat 服务器是一个免费的开放源码的Web服务器，也可以视为轻量级应用服务器。
- Tomcat同时也是运行Servlet和JSP 的Web(应用)容器/Servlet容器



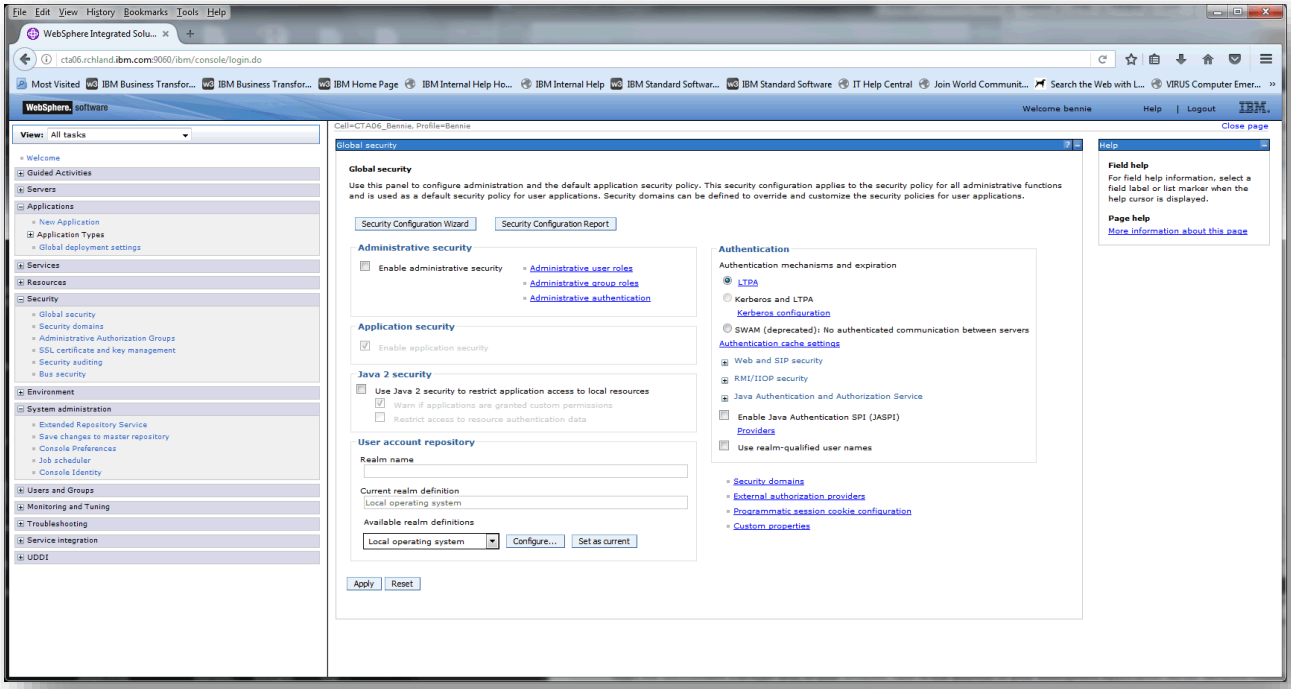
Web服务器与应用服务器

- 拓展：Apache Jakarta Project
  - 与Web开发关系紧密
  - 代表性开源项目：Maven, Tomcat, Lucene, Log4j
    - 目前各个项目均已独立，2011年Jakarta project was retired to the Attic



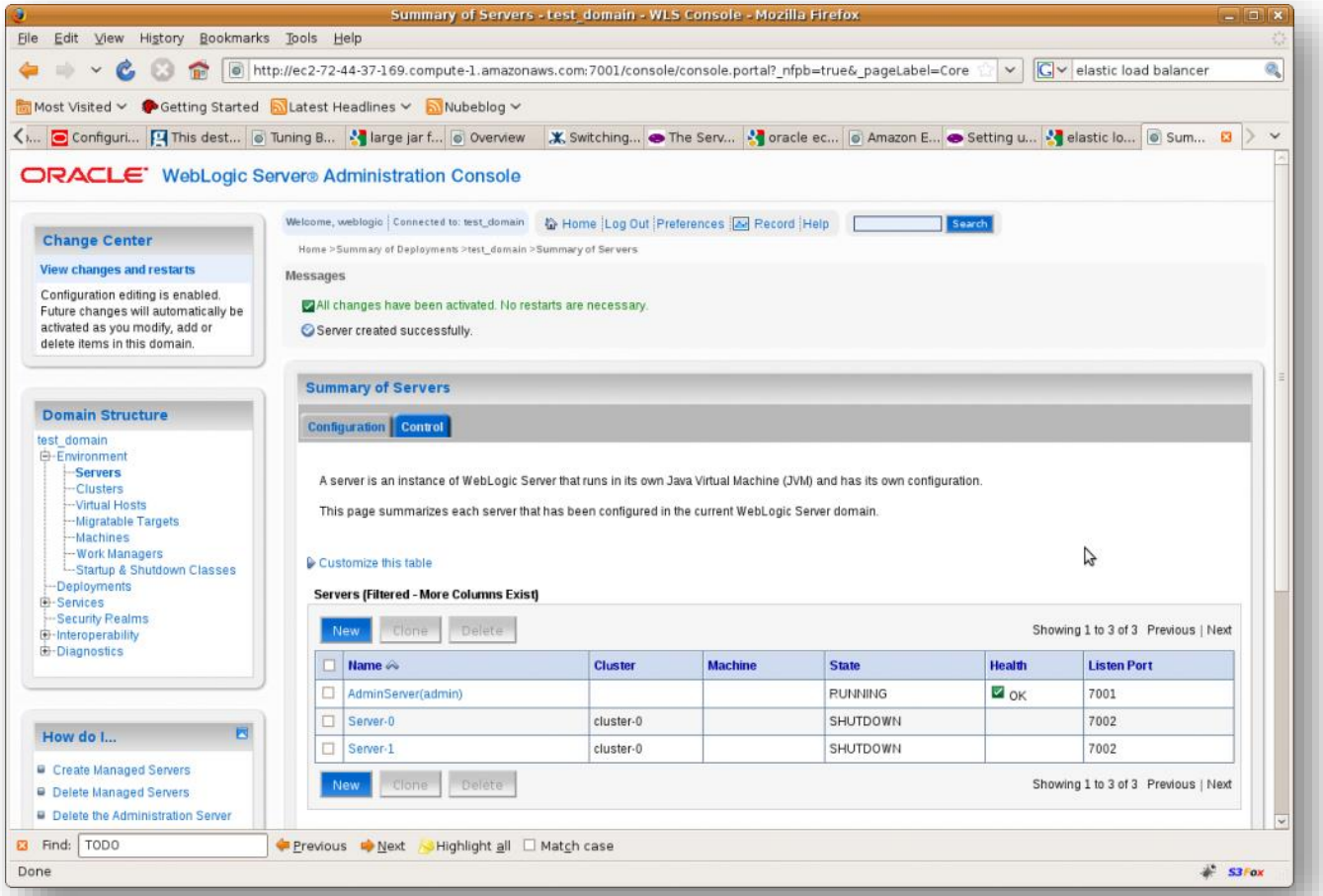
Web服务器与应用服务器

- **应用服务器**Application Server
  - Servlet容器+Java EE服务器（EJB容器、JNDI服务器、JMS服务器等）
  - Web服务器专门处理HTTP请求(request)，但是应用程序服务器是通过很多协议来为应用程序提供(serves)商业逻辑(business logic)
    - WebSphere
    - Jboss
    - WebLogic



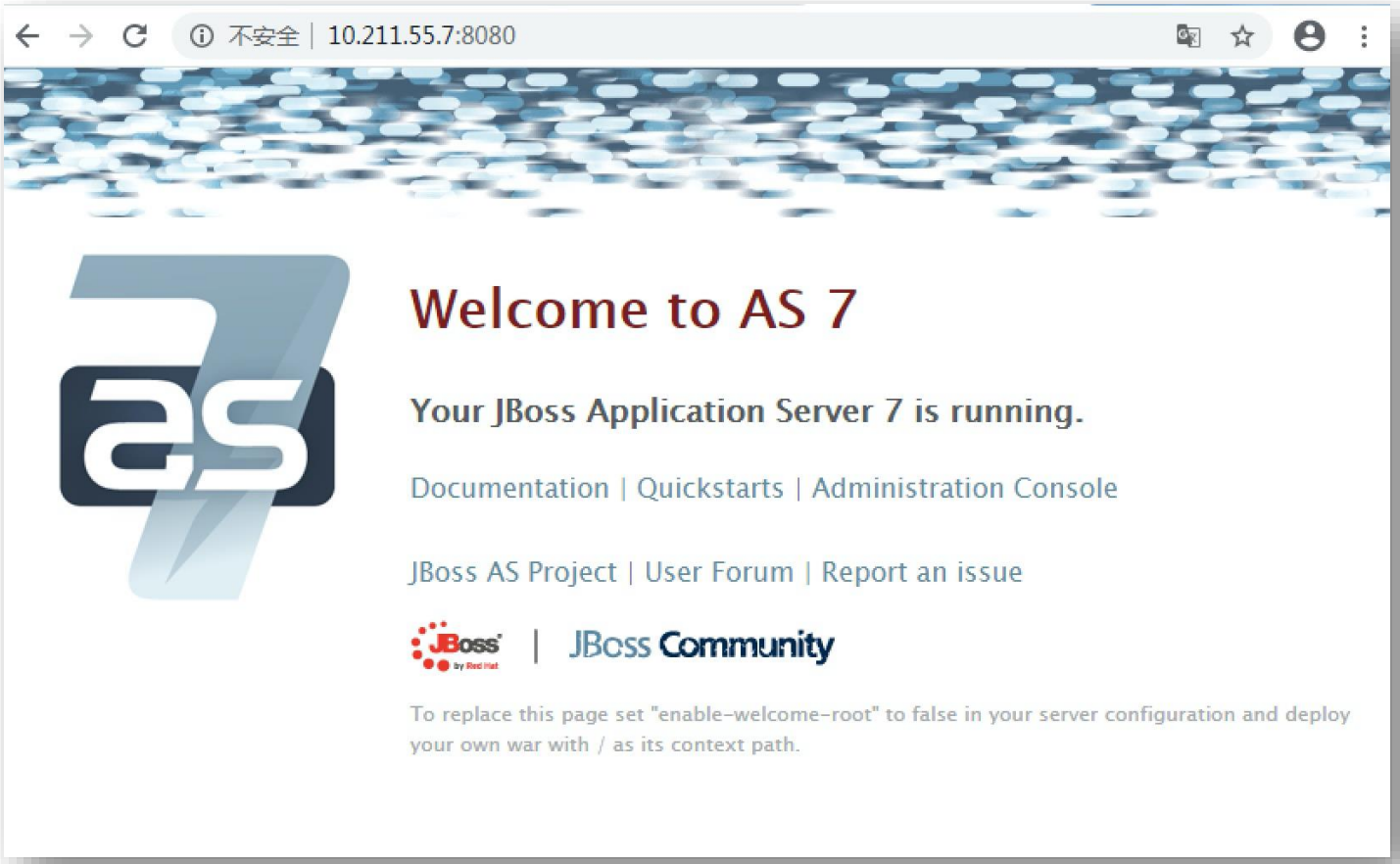
IBM WebSphere

Web服务器与应用服务器



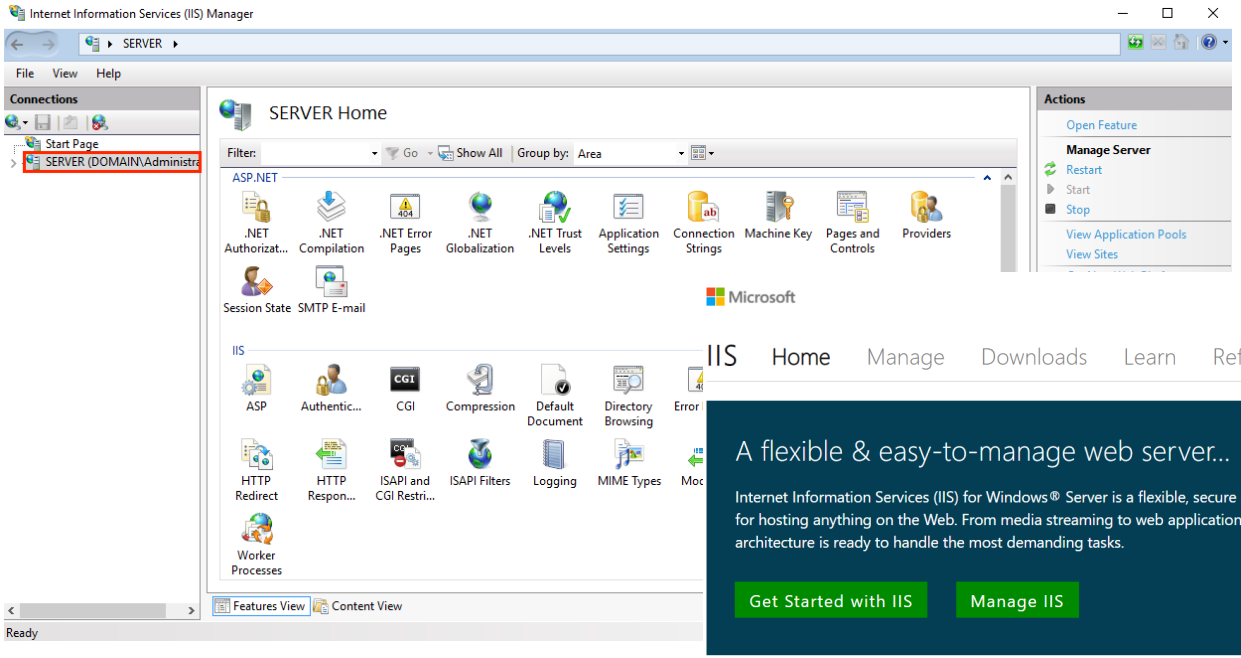
ORACLE
WebLogic Server

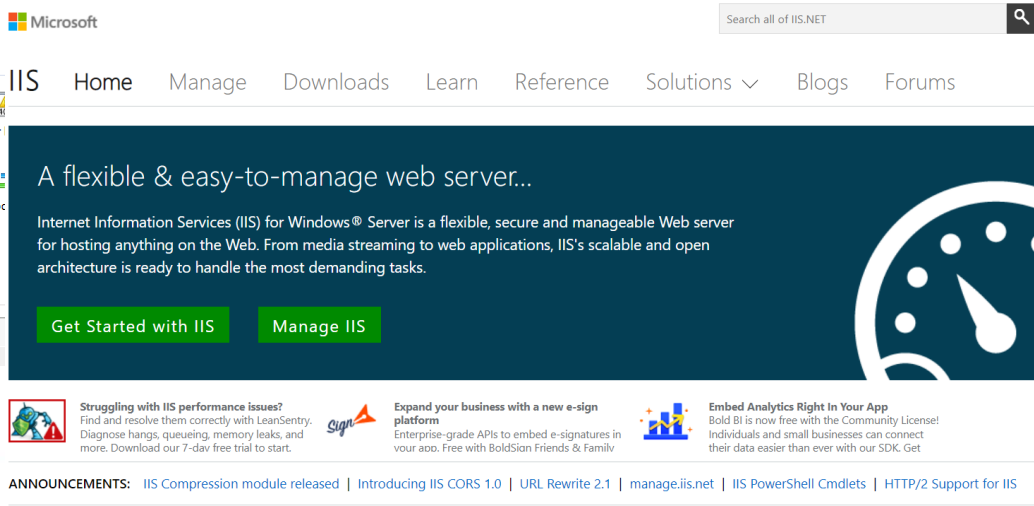
Web服务器与应用服务器



Web服务器与应用服务器

- **IIS**(Internet Information Services)
  - Microsoft IIS提供Web服务器、FTP服务器、NNTP服务器和SMTP服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送，
  - 提供ISAPI(Intranet Server API）作为扩展Web服务器功能的编程接口；提供数据库连接器
  - https://www.iis.net/





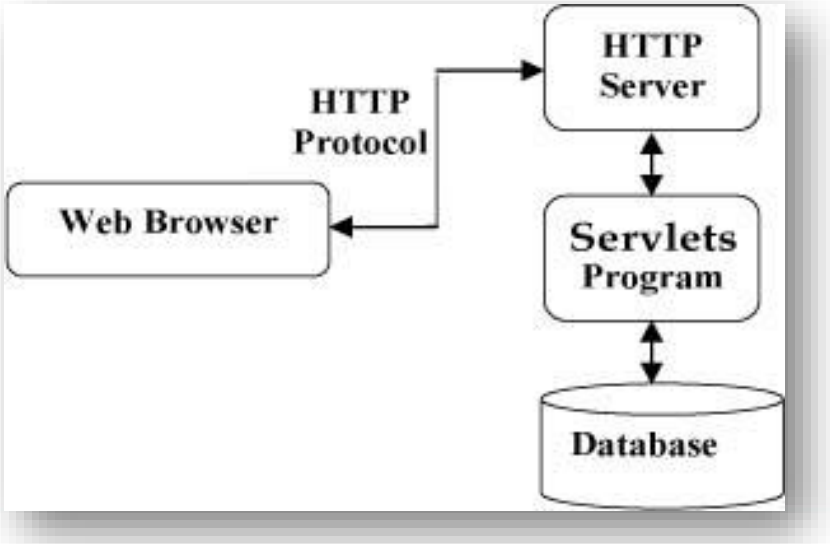
目录

Servlet

- Java Server Applet (Servlet)
  - Java编写的服务器端程序，更技术地讲，Java Servlet是运行在带有支持 Java Servlet 规范的Web服务器上的 Java 类。
  - 用于交互式浏览和修改数据，生成动态Web内容。
- 工作模式
  - 客户端发送请求至服务器
  - 服务器启动并调用Servlet，Servlet根据客户端请求生成响应内容并将其传给服务器
  - 服务器将响应返回客户端

Servlet

- 特点：
  - Servlet 在 Web 服务器的地址空间内执行。无需创建一个进程来处理每个客户端请求。
  - Servlet使用 Java 编写，跨平台。
  - Servlet 是可信的。服务器上的 Java 安全管理器执行了一系列限制，保护服务器计算机上的资源。
  - Java 类库的全部功能对 Servlet 来说都是可用的，可以通过 sockets 和 RMI 机制与 applets、数据库或其他软件进行交互。



Servlet

Servlet 任务

- 读取客户端（浏览器）发送的显式的数据。这包括网页上的 HTML 表单，或者也可以是来自 applet 或自定义的 HTTP 客户端程序的表单。
- 读取客户端（浏览器）发送的隐式的 HTTP 请求数据。这包括 cookies、媒体类型和浏览器能理解的压缩格式等等。
- 处理数据并生成结果。访问数据库，执行 RMI 或 CORBA 调用，调用 Web 服务，或者直接计算得出对应的响应。

- 发送显式的数据（即文档）到客户端（浏览器）。该文档的格式可以是多种多样的，包括文本文件（HTML 或 XML）、二进制文件（GIF 图像）、Excel 等。
- 发送隐式的 HTTP 响应到客户端（浏览器）。这包括告诉浏览器或其他客户端被返回的文档类型（例如 HTML），设置 cookies 和缓存参数，以及其他类似的任务。

Servlet

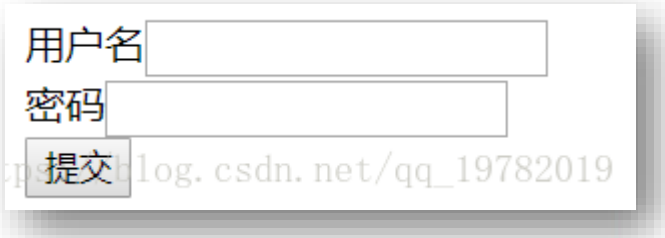
- Servlet包结构
  - 1.javax.servlet：包含定义servlet和servlet容器之间契约的类和接口。
  - 2.javax.servlet.http：包含定义HTTP Servlet 和Servlet容器之间的关系。
  - 3.javax.servlet.annotation：包含标注servlet，Filter,Listener的标注。它还为被标注元件定义元数据。
  - 4.javax.servlet.descriptor：包含提供程序化登录Web应用程序的配置信息的类型。

Servlet

Servlet实例

- 编写HTML表单，静态页面

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Title</title>
    </head>
   <body>
     <form action="/form" method="get">
         <span>用户名</span>
         <input type="text" name="username"><br>
         <span>密码</span>
         <input type="password" name="password"><br>
         <input type="submit" name="submit">
     </form>
   </body>
</html>
```



Servlet

编写Servlet

```
public class FormServlet extends HttpServlet {

private static final long serialVersionUID =
-4186928407001085733L;

@Override
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
}

@Override
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
  //设置响应的编码格式为UTF-8编码，否则发生中文乱码现象
  response.setContentType("text/html;charset=UTF-8");
  //1.获得请求方式
 String method = request.getMethod();
  //2.获得请求的资源相关的内容
 String requestURI = request.getRequestURI();//获得请求URI
 StringBuffer requestURL = request.getRequestURL();
 String webName = request.getContextPath();//获得应用路径
 String querryString = request.getQueryString();//获得查询字符串
  PrintWriter out = response.getWriter();
  out.write("<h1>下面是获得的字符串</h1>");
  out.write("<h1>method(HTTP方法):<h1>");
  out.write("<h1>"+method+"</h1><br>");
  out.write("<h1>requestURi(请求URI）:</h1>");
  out.write("<h1>" + requestURI + "</h1><br>");
  out.write("<h1>webname(应用名称):</h1>");
  out.write("<h1>" + webName + "</h1><br>");
  out.write("<h1>querrystring(查询字符串
):</h1>");
  out.write("<h1>" + querryString + "</h1>");
}
}
```

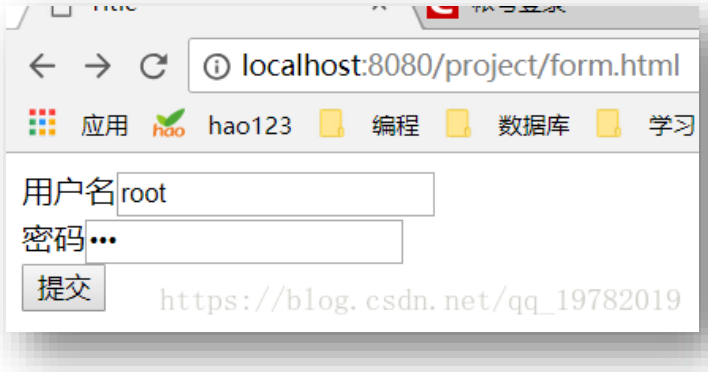
接收form登录表单发送过来的HTTP请求，解析出请求中封装的参数，回写到response响应中，在浏览器端显示。

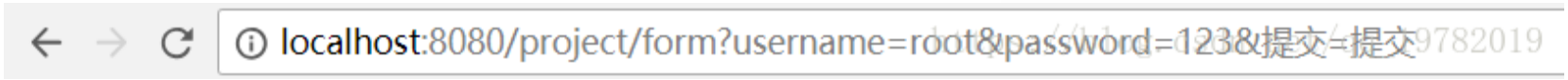
Servlet

- 配置Servlet映射关系：

```
</servlet-mapping>
  <servlet>
    <servlet-name>FormServlet</servlet-name>
    <servlet-class>com.javaee.util.FormServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FormServlet</servlet-name>
    <url-pattern>/form</url-pattern>
  </servlet-mapping
```

启动tomcat，在浏览器中输入登录表单的地址：





登录表单为了安全性大多采用Post方式提交，这里是为了展示方便

Servlet

- Servlet与CGI对比
  - 进程与线程
    - 当用户浏览器发出一个Http/CGI的请求，或者说 调用一个CGI程序的时候，服务器端就要新启用一个进程 (而且是每次都要调用)
    - Servlet充分发挥了服务器端的资源并高效的利用。每次调用Servlet时并不是新启用一个进程 ，而是在一个Web服务器的进程内生成线程
  - 平台无关性
    - 传统的CGI程序，不具备平台无关性特征，Servlet具备Java的平台无关性
  - 二层架构与三层架构
    - CGI: Web服务器+数据库服务器
    - Servlet: Web服务器 + 连接池 + 数据库服务器
      - 在系统缓存中事先建立好若干与数据库的连接，若想与数据库链接，可以随时向系统请求一个连接

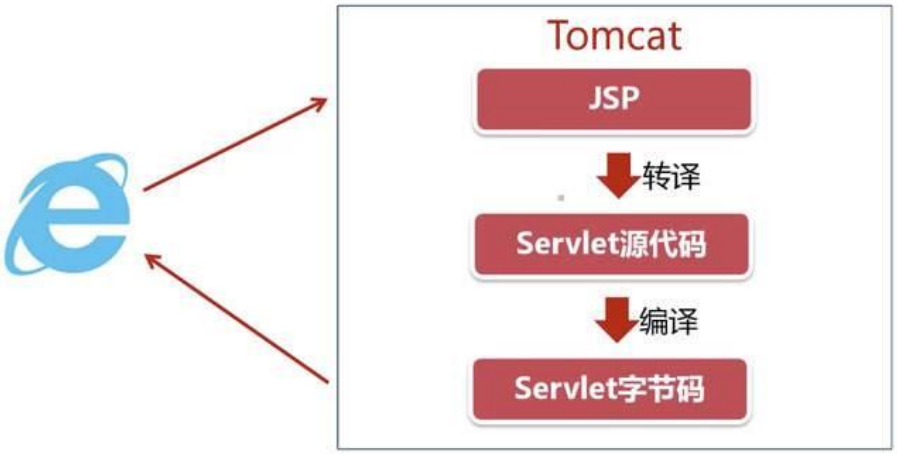
目录

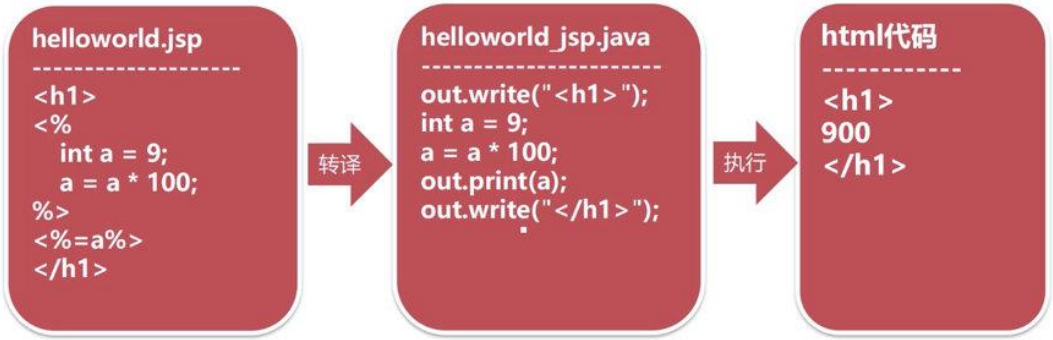
JSP

- Java Server Page
  - Servlet的缺点
    - Html与Java代码混合在一起,维护困难
    - 每一行都是一个println语句,效率低下
  - JSP，动态网页生成技术
    - JSP可将原本都在Servlet中的java代码与HTML分离,降低耦合,可维护性好
    - 可编写原生HTML,且编译器会进行语法检查,开发效率更高
    - JSP本质就是Servlet,被执行前会被先转译为java文件

JSP

JSP运行





转译为servlet后，同样利用printWriter来向前台返回响应数据

JSP

- JSP语法
  - 表达式
    - 语法格式：<%=表达式%>
    - 作用：通过表达式可以输出表达式的结果值。其本质就是执行了print语句;
    - 示例

Jsp页面书写

```
<body>
    1+5的结果为：<%=1+5 %>
</body>
```

在网页的显示

1+5的结果为：6

JSP

- JSP语法
  - 代码块
    - 语法格式：<%代码内容%>
    - 作用：写任何Java代码,用来输出产生HTML文本内容
    - 示例:

Jsp页面书写

```
<body>
    <%
        for(int i=0;i<5;i++){
    %>
            <a href="test.jsp">我是a链接</a><br>
    <%
        }
    %>
</body>
```

在网页的显示

我是a链接
我是a链接
我是a链接
我是a链接
我是a链接

JSP

- JSP语法
  - 声明块
    - 语法格式：<%!代码内容%>
    - 作用：在JSP中声明方法和变量
    - 示例:

定义

```
<%!
   int a = 100;
public int add(int a,int b){
   return a+b;
}
%>
```

使用

```
<body>
   <h1>10+20=<%=add(10,20)%></h1>
<h1>a的值为<%=a%></h1>
</body>
```

JSP

- JSP语法
  - 注释
    - <%-- jsp注释 --%>　　页面不可见
    - <!-- html注释 -->　　页面可见
    - //　/* */　　代码块内java注释 转译jsp时会放到java文件中

JSP

JSP语法

- **编译指令（directive)**
- 处理当前jsp全局配置,例如导入Java类，使用标签库，或内容编码等，由Servlet引擎处理，在JSP转译Servlet时生效
- 语法:<%@ page 属性名称="属性值">
  - 示例<%@ page import="java.util.HashMap" %>

指令名	作用
include	静态引入其他JSP页面
taglib	导入标签库,设置标签前缀等
page	导入Java类,设置响应编码等

JSP

- JSP语法
  - **动作指令（action）**
    - JSP执行处理阶段完成特定功能的标签，和JSP指令元素不同，它是在请求处理阶段执行，而指令则是在编译阶段被执行。
    - 利用JSP动作可以动态地包含其他文件、重定向页面等等，其中最常用的当属<jsp:include>

语法: <jsp:动作名称 名称="值">

示例<jsp:forward page="/FirstServlet"></jsp:forward>

JSP

JSP常用指令及作用

指令名	作用
jsp:forward	执行页面转向，将请求的处理转发到下一个页面/servlet。
jsp:param	用于传递参数，必须与其他支持参数的标签一起使用。
jsp:include	用于动态引入一个JSP页面。
jsp:plugin	用于下载JavaBean或Applet到客户端执行。
jsp:useBean	创建一个JavaBean实例。
jsp:setProperty	设置JavaBean实例的属性值。
jsp:getProperty	输出JavaBean实例的属性值。

JSP

- include编译指令与include动作指令对比

	include指令	jsp:include动作
语法格式	<%@include file=".." %>	<jsp:include page=".." >
发生作用的时间	页面转换时	请求期间
包含的内容	文件的实际内容（源代码）	页面的输出（结果）
转换成servlet	主页面和包含页面转换为一个servlet	主页面和包含页面分别转换为独立的servlet
编译时间	较慢——资源必须被解析	较快
执行时间	稍快	较慢——每次资源必须被解析
使用方式	页面内容不经常变化时	页面内容经常变化时

JSP

<%@ include file="URL"%>

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<h1>Include指令</h1>
<hr>
<%@include file="date.jsp" %>
</body>
</html>
```

<%--page表示要包含的页面 flush表示被包含的页面是否从缓冲区读取--%>

<jsp:include page="URL " flush="true|false/>

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<h1>Include动作</h1>
<hr>
<jsp:include page="date.jsp" flush="false"/>
</body>
</html>
```

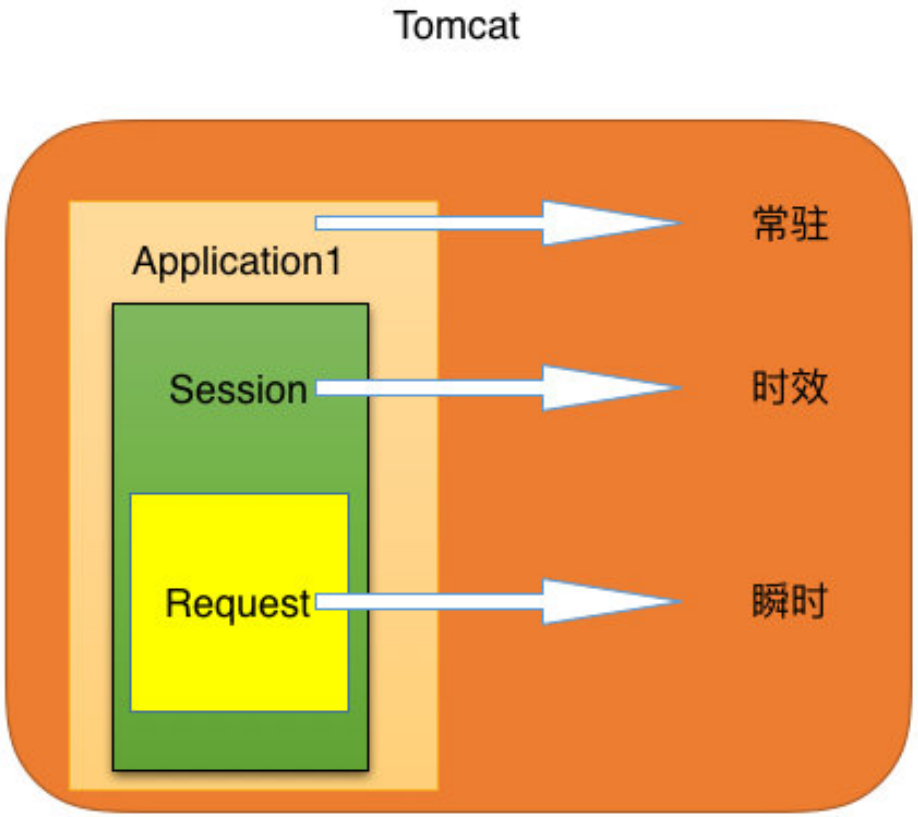
JSP

- JSP作用域
  - JSP内置的四个作用域

名称	描述
page	PageContext页面上下文,在当前页面有效
request	对同一个请求有效,(请求转发是同一个，重定向是新的)
session	当前会话有效
application	ServletContext，当前web应用程序全局有效

JSP

- 作用域范围



- 合理的选择最合适的作用域对象存储数据，节省服务器资源开销

目录

Cookie与Session

- HTTP协议的无状态性
  - 无状态：当浏览器给服务器发送请求时，服务器响应客户端请求。但是当同一个浏览器再次发送请求时，服务器并不知道这就是刚才的浏览器；
    - 客户端刚刚登陆成功，再次登录时又要重新登陆
  - 总结：协议对于交互性场景没有记忆能力
  - HTTP 1.0与1.1为无状态协议，HTTP 2.0严格意义上为有状态协议
  - 目前应用最多的是HTTP 1.1
  - 所以出现了Cookie、Session

有状态：

A：你今天中午吃的啥?

B：吃的大盘鸡。

A：味道怎么样呀?

B：还不错，挺好吃的。

无状态：

A：你今天中午吃的啥?

B：吃的大盘鸡。

A：味道怎么样呀?

B：？？？啊？啥？啥味道怎么样?

所以需要cookie这种东西：

A：你今天中午吃的啥?

B：吃的大盘鸡。

A：你今天中午吃的大盘鸡味道怎么样呀?

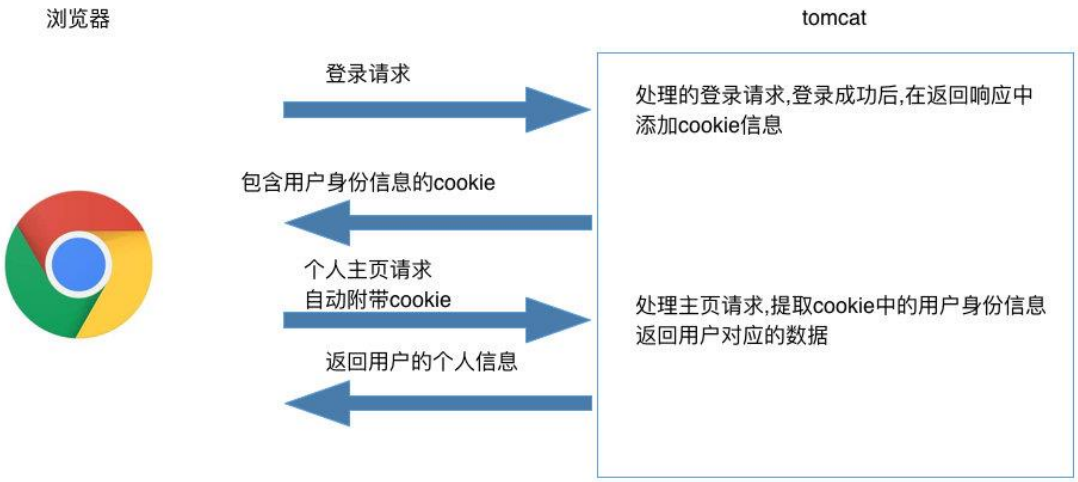
B：还不错，挺好吃的。

Cookie与Session

Cookie

- cookie是服务器保存在浏览器的一小段文本数据(4k内)，通常与用户个人信息有关
- cookie数据以键值对形式存在
- 用于在HTTP协议下维持客户端与服务器的状态
- cookie具有时效性，在有效期内
- 每次请求相同的服务器(域名相同，路径相同)时，浏览器都会自动将cookie放入请求头一起发送给服务器

Cookie工作过程



Cookie与Session

Session

- 浏览器和服务器在一段时间内的所有交互动作
  - session用于保存与浏览器进程对应的数据
  - 数据存储在服务器内存中,具有时效性,默认为30分钟，可通过代码或web.xml修改
  - 服务器从Cookie中获取sessionID来找到对应的session对象，从而实现维持用户的状态
  - session在服务器是一个Java对象，可以添加任何类型的属性到其中
  - 当session在一段时间内没有被访问时将被销毁

Session工作过程

- 如果请求的是JSP将自动创建session对象

课程材料外参照资料

- CoderJerry. JSP + Session Cookie详解. https://www.cnblogs.com/yangyuanhu/p/12030367.html, 2019
- 进阶技术. JavaWeb学习总结(一)——JavaWeb开发入门 . https://www.cnblogs.com/ljangle/p/10078919.html, 2018
- 刘扬俊. JavaWeb——Servlet. https://blog.csdn.net/qq_19782019/article/details/80292110, 2018
- CoderJerry. Java Web. https://www.cnblogs.com/yangyuanhu/p/11978941.html, 2019
- 码农BookSea. Ajax介绍以及工作原理和实现详解. http://www.xiaoheidiannao.com/66583.html, 2019

西安电子科技大学

感谢！