

# Java EE开发技术概述

邹国兵

上海大学计算机学院

[gbzou@shu.edu.cn](mailto:gbzou@shu.edu.cn)

# 目录

- 组件与组件开发
- 主流Web开发技术及比较
- Java EE基本内容
- Java EE轻量级框架
- Java EE开发与部署环境配置
- 开发第一个Java EE应用

# 软件领域的发展

- 软件开发模式的变化
- 软件需求的变化
- 软件环境的变化
- 程序设计方法的变化

# 软件开发模式

机器码



高级语言  
和脚本代码

# 软件需求的变化

- 计算
- ↓
- 实用
- ↓
- 管理 (MIS)
- ↓
- 分布式系统

# 软件环境的变化

单任务



多任务

文字界面



图形界面

单线程



多线程

平台相关



跨平台

单机(本地)



网络(分布式)

单一语言



多种语言

# 程序设计方法的发展

- 结构化程序设计 —— 以数据为中心
- 面向对象程序设计 —— 以对象为中心
- 组件程序设计 —— 以组件为中心

# 对象技术

- 用一种新的思路来看待问题
  - 10多年的发展证明了OO符合软件的规律
  - 对象技术的发展需要开发工具和开发语言支持
- 三大特性：封装性、继承性、多态性

# 面向对象技术弱点

- 在中小规模的软件中，对象和对象之间的协作关系就能够满足需要。
- 当软件规模扩大，复杂度上升，面向对象技术强调的协作却表现出另一个极端特点—**耦合度太高导致的复杂度**。

# 面向组件编程

- 面向组件编程的缩写是**COP**。 **COP**是对**OOP**的补充，帮助实现更加优秀的软件结构。组件的粒度可大可小，需要取决于具体的应用。
- 在**COP**中有几个重要的概念：
  - **服务**：服务（**Service**）是一组接口，供客户端程序使用。例如，验证和授权服务，任务调度服务。服务是系统中各个部件相互调用的接口；
  - **组件**：组件（**Component**）实现了一组服务，此外，组件必须符合容器订立的规范，例如初始化，配置、销毁。

# 面向组件编程

- COP是一种组织代码的思路，尤其是服务和组件这两个概念。在 Spring框架中，就采用了COP的思路，将系统看作一个个的组件，通过定义组件之间的协作关系来完成系统构建。
- 这样做的好处是能够隔离变化，合理的划分系统。而框架的意义就在于定义一个组织组件的方式。

# 组件的粒度

- 组件的粒度是和系统的架构息息相关的。组件的粒度确定了，系统的架构也就确定了。
- 在小规模的软件中，可能组件的粒度很小，仅相当于普通的对象，但是对于大规模的系统来说，一个组件可能包括几十，甚至上百个对象。
- 对使用**COP**技术的系统来说，需要正确的定义组件的粒度。定义粒度的方法是对核心流程进行分析。

# 针对接口编程

- 接口和实现分离是COP的基础，没有接口和实现的分离，就没有COP。接口的高度抽象特性使得各个组件能够被独立的抽取出来，而不影响到系统的其它部分。

# 接口和实现分离的好处

- 在模块/组件/对象之间解耦。
- 轻松的抽换实现，而不用修改客户端。
- 用户只需要了解接口，不需了解实现细节。
- 增加了重用的可能性。

# 基于组件开发的好处

- 降低应用开发的难度
- 大大提高了软件开发效率
- 提升了软件的质量
- 提高了系统维护的便利性
- 促进了代码重用

# 目录

- 组件与组件开发
- 主流**Web**开发技术及比较
- Java EE基本内容
- Java EE轻量级框架
- Java EE开发与部署环境配置
- 开发第一个Java EE应用

# 主流Web开发技术

- LAMP
- Java EE
- .NET

# LAMP

- LAMP: Linux+Apache+MySQL+PHP
- LAMP是一个缩写，它指一组通常一起使用，运行动态网站或者服务器的开源软件。
- 包括: Linux操作系统, Apache网络服务器, MySQL数据库, Perl、PHP或Python语言。
- 核心技术: PHP
- 应用领域: 大型商业网站（如开心网、校内网）

# Java EE

- 经过多年沉淀， Java EE平台已经成为电信、金融、电子商务、保险、证券等各行业的大型系统首选开发平台。
- Java行业的软件开发技术已经基本稳定， 目前， Java EE大致分为两种方式：
  - (1) 以Spring为核心轻量级Java EE企业开发平台。
  - (2) 以EJB 3+JPA为核心的的经典Java EE开发平台。

# Java EE业界成功案例

- [www.taobao.com](http://www.taobao.com)
- [www.ebay.com](http://www.ebay.com)
- [www.tudou.com](http://www.tudou.com)
- [www.pconline.com.cn](http://www.pconline.com.cn)
- [www.javaeye.com](http://www.javaeye.com)

# 目录

- 组件与组件开发
- 主流Web开发技术及比较
- **Java EE**基本内容
- Java EE轻量级框架
- Java EE开发与部署环境配置
- 开发第一个Java EE应用

# Java & Java EE

- Java和Java EE是两个不同概念，Java不只是指一种语言，已经代表与微软不同的另外一个巨大阵营。所以，Java有时是指一种软件系统的流派，目前主要是.NET和Java两大主流体系。
- 目前Java有三个平台：
  - 适用于小型设备和智能卡的Java平台Micro版 (Java Platform Micro Edition, **Java ME**)
  - 适用于桌面系统的Java平台标准版 (Java Platform Standard Edition, **Java SE**)
  - 适用于创建服务器应用程序和服务的Java平台企业版 (Java Platform Enterprise Edition, **Java EE**)。

# Java EE应用概述

- 实际上，Java Web应用核心技术JSP、Servlet与JavaBean。但遗憾的是，一直没有一套有效的开发规范来约束JSP程序员，导致不同程序员写出不同风格的JSP程序。
- 项目的规模越大，这种弊端就越明显，项目难以维护和升级。

# Java EE应用概述

- 这时，一个基于Web框架开发的Web应用时代已经来临。继Struts成功推出之后，大批优秀的框架紧跟其后，席卷了整个市场。
- 到目前为止，Java EE是目前开发Web应用（特别是企业级Web应用）最流行的平台之一。

# What is Java EE

- 经典Java EE应用以EJB（企业级JavaBean）为核心，以应用服务器为运行环境。所以，通常开发和运行成本高。
- 本课程介绍的轻量级Java EE应用具备了Java EE规范的种种特征。例如，面向组件建模思维、优秀的应用分层、良好的扩展性和可维护性。
- 轻量级Java EE保留了经典Java EE应用的架构，但开发更简单，成本更低。

# Java EE应用分层结构



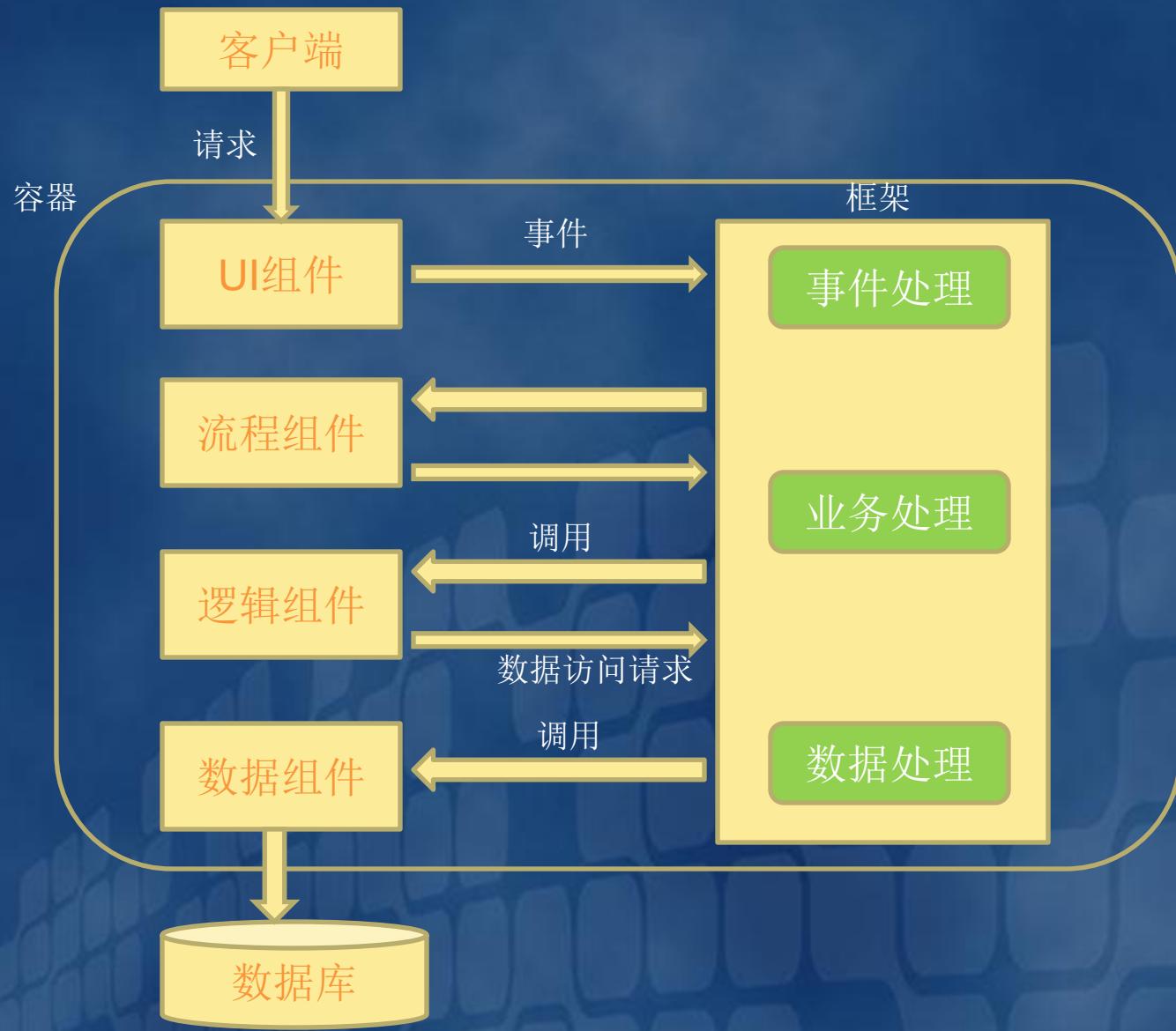
# Java EE应用的分层模型（1）

- 不管是经典的Java EE架构还是轻量级Java EE架构，大致上可以分为以下几层：
  - 领域对象（**Domain Object**）层：此层有系列的POJO（Plain Old Java Object，普通传统的Java对象）
  - 数据访问对象（**Data Access Object, DAO**）层：此层由系列的DAO组成，这些DAO实现了对数据库的创建、查询、更新和删除（CRUD）等原子操作。

# Java EE应用的分层模型（2）

- **业务逻辑层：**此层由系列的业务逻辑对象组成，这些业务逻辑对象实现了系统所需要的业务逻辑方法。
- **控制器层：**由一系列控制器组成，这些控制器用于拦截用户请求，并调用业务逻辑组件的业务逻辑方法处理用户请求，根据处理结果转发到不同的表现层组件。
- **表现层：**此层由一系列的**jsp**页面、**Velocity**页面或**PDF**文档视图组件组成，负责收集用户请求，并显示处理结果。

# 组件，框架和容器之间的关系



# 框架的作用

- 为组件运行提供基础服务
- 为组件开发提供基础类
- 实现了组件之间的隔离
- 支持组件间的协作
- 框架和组件运行于容器之内

# Java EE应用的组件（1）

- Java EE大致包括如下几个组件：
  - 表现层组件：主要负责收集用户输入数据和向用户显示系统状态。最常用表现层是JSP。
  - 控制器组件：对于Java EE的MVC框架而言，框架提供一个前端核心控制器，负责拦截用户请求，并将请求转发给控制器组件，控制器组件调用业务逻辑方法，处理用户请求。控制器组件有Struts等。

# Java EE应用的组件（2）

- **业务逻辑组件：**系统的核心组件，实现系统的业务逻辑。通常，一个业务逻辑方法对应一次用户操作。一个业务逻辑方法应该是一个整体，因此我们要求对业务逻辑方法增加事务性。业务逻辑方法仅仅负责实现业务逻辑，不应该进行数据库访问。因此，在业务逻辑层不应该出现Hibernate，JDBC等API。

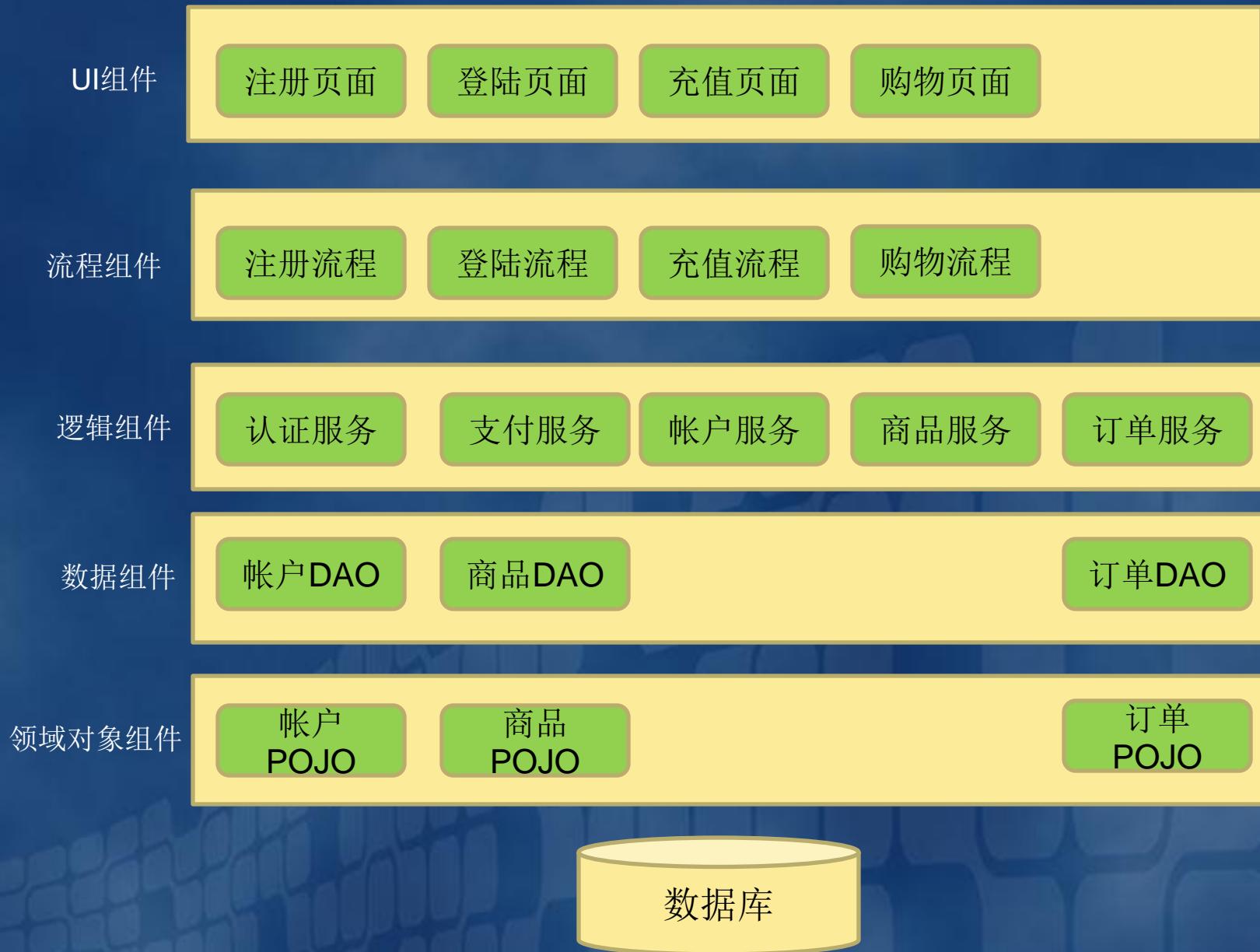
# Java EE应用的组件（3）

- **DAO组件：**数据访问对象组件。每个**DAO**组件都提供**Domain Object**对象基本的**CRUD**操作。采用不同的持久层访问技术，**DAO**组件的实现会完全不同。为了业务逻辑组件的实现与**DAO**组件的实现分离，我们为每个**DAO**组件提供接口，业务逻辑组件面向**DAO**接口编程，从而达到最好的解耦。

# Java EE应用的组件（4）

- **领域对象组件：**领域对象抽象了系统的对象模型。通常而言，这些领域对象的状态都必须保存在数据库里。因此，每个领域对象通常对应一个数据表。

# 应用的构成



# Java EE的优势

- 用简单的JSP、Servlet和JavaBean就能完成，为什么还要用Spring, Hibernate和Struts这些开发框架？
  - 软件开发不仅仅考虑开发过程，还要考虑后期的维护、扩展；不仅仅考虑那些小型系统，还要考虑大型系统的协同开发。
  - 对于大型系统而言，采用Java EE架构有很大优势。
    - 软件不是一次性系统，在软件的更新换代中，不是彻底替换软件，只能在其原来基础上延伸。如果支撑企业系统的软件不具备扩展性，将损失惨重。
    - 对于信息化系统，前期开发工作对于整个系统工作量而言只是一小部分，后期的维护、升级往往占更大比重，企业需求发生重大变化，这些都要求软件具有很好的伸缩性。
    - Java EE可以让软件系统中组件以松耦合的方式组织在一起，让应用之间的耦合停留在接口层次，而不是代码层次。

# Java EE的优势

Java EE为搭建具有可伸缩性、灵活性、易维护性的商业化系统提供了良好的机制：

- 保留现存的IT资产
- 高效的开发
- 坚持面向对象的设计原则
- 灵活性、可移植性和互操作性
- 轻松的企业信息系统集成
- 引进面向服务的体系结构
- 具有敏捷可维护特征

# HOW?



Java EE如此庞大！怎样入手学习？

# Solution

- Java EE without EJB
- Simplify the Best: Light Framework



# 目录

- 组件与组件开发
- 主流Web开发技术及比较
- Java EE基本内容
- Java EE轻量级框架
- Java EE开发与部署环境配置
- 开发第一个Java EE应用

# 概念理清

Java技术流（以Java作为开发语言的技术）

Java平台（以Java SE为核心API）

Java EE架构规范

EJB

非EJB

Java EE容器

JSP, Servlet ...

JDBC,JNDI,JMS...

# 相关概念

- Java EE=Java EE容器+Web应用
- Java EE容器包括EJB容器和Web容器
- Web容器是指JSP/Servlet容器，开发一个Web项目，无论是编译或运行，都必须要有JSP/Servlet库或API支持。

# Java Web服务器

- 常见的Web服务器有三个：
  - **Tomcat:** 开源的Java Web服务器，和Java结合得最好，Sun官方推荐的JSP服务器。性能和稳定性都非常优秀。
  - **Jetty:** 另一个优秀的Web服务器，优点是可以作为嵌入式服务器。在应用中加入Jetty的jar文件，应用在代码中对外提供Web服务。
  - **Resin:** 目前最快的JSP、Servlet运行平台，支持EJB。

# 专业Java EE服务器

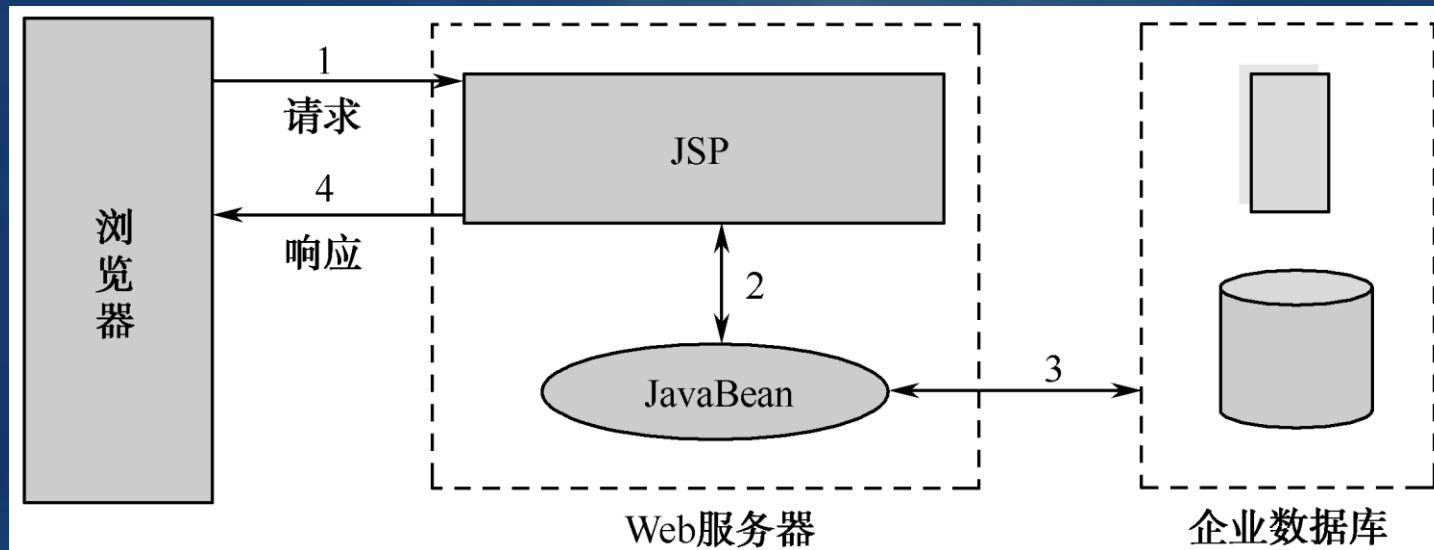
- Java EE服务器支持更多的Java EE特性，例如分布式事务、EJB容器等。
- 常见的Java EE服务器有：
  - **JBoss**: 开源的Java EE服务器，支持EJB。
  - **Weblogic**和**Websphere**: 两个专业的商用Java EE服务器，价格不菲。

# Java EE与设计模式

- **模式1： JSP+Java Beans**

- 以JavaBean封装部分业务逻辑的开发模式
- 这种模式的最大优势是实现起来比较简单，适合快速开发小规模的项目。

# 模式1： JSP+Java Beans

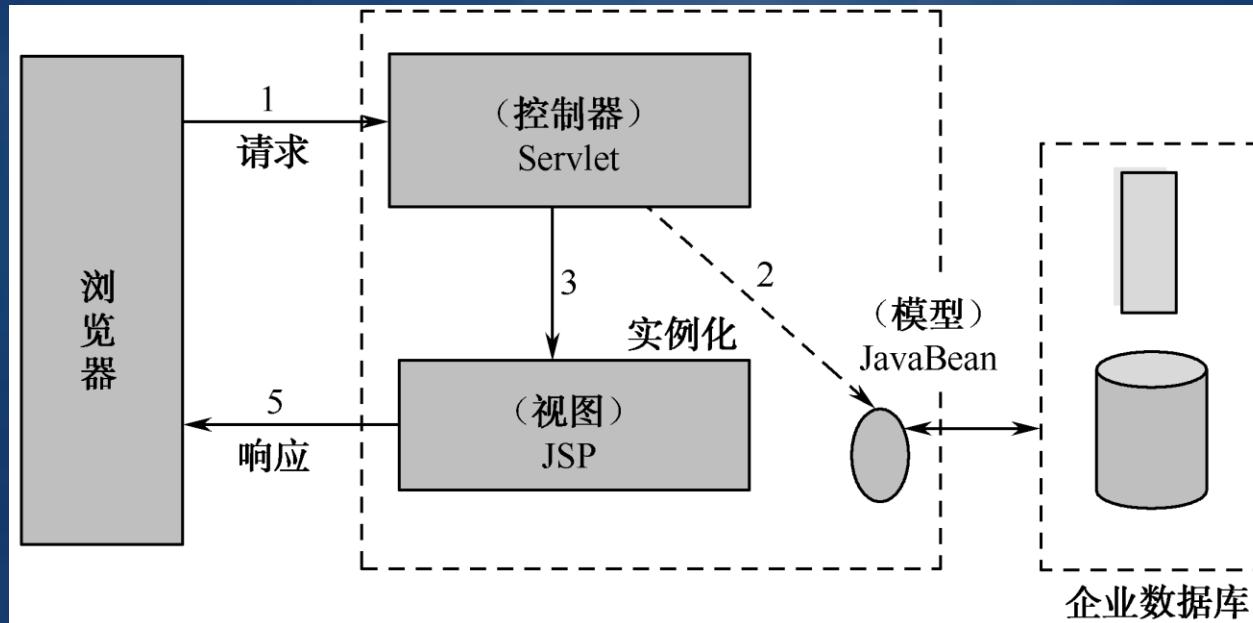


- **Model 1**模式实现比较简单，适用于快速开发小规模项目。
- 但从工程化的角度看，它的局限性非常明显：**JSP**页面身兼**View**和**Controller**两种角色，将控制逻辑和表现逻辑混杂在一起，从而导致代码的重用性非常低，增加了应用的扩展和维护难度。

# Java EE与设计模式

- 模式2：MVC（Model-View-Controller）
  - 合理使用了Servlet和JSP的各自特点，Servlet负责业务流程的控制，JavaBean负责业务数据的逻辑处理，JSP专注于页面表示。
  - 各个层次的责任明确而且独立，开发和维护非常容易，系统的可扩展性也比较好。

# 模式2：Model-View-Controller



- **Model 2**下**JSP**不再承担控制器的责任，它仅仅是表现层角色，仅仅用于将结果呈现给用户；
- **JSP**页面的请求与**Servlet**（控制器）交互，而**Servlet**负责与后台的**JavaBean**通信。
- 模型（**Model**）由**JavaBean**充当，视图（**View**）由**JSP**页面充当，而控制器（**Controller**）则由**Servlet**充当。

# 典型的MVC Framework

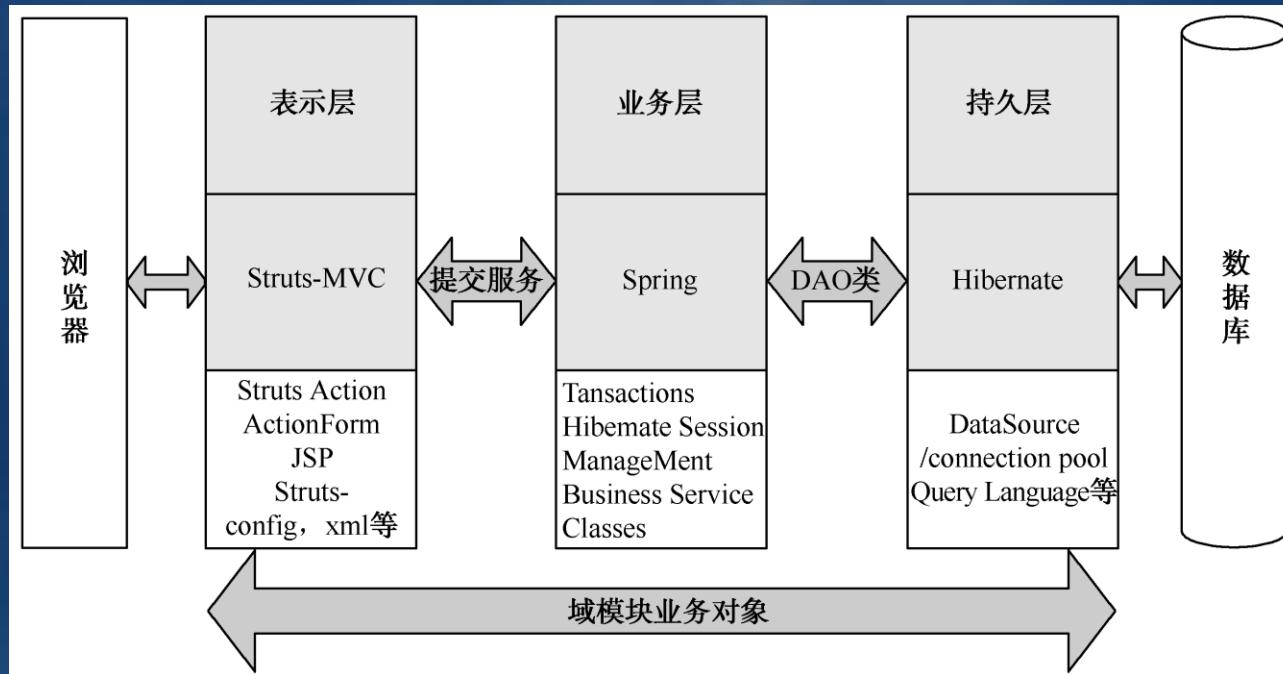
- Struts
- Spring MVC
- Tapestry
- Webwork
- JSF

# 我们选择的技术路线

技术路线：

- Struts 2+Spring 3+Hibernate 3, SSH

# Struts+Spring+Hibernate

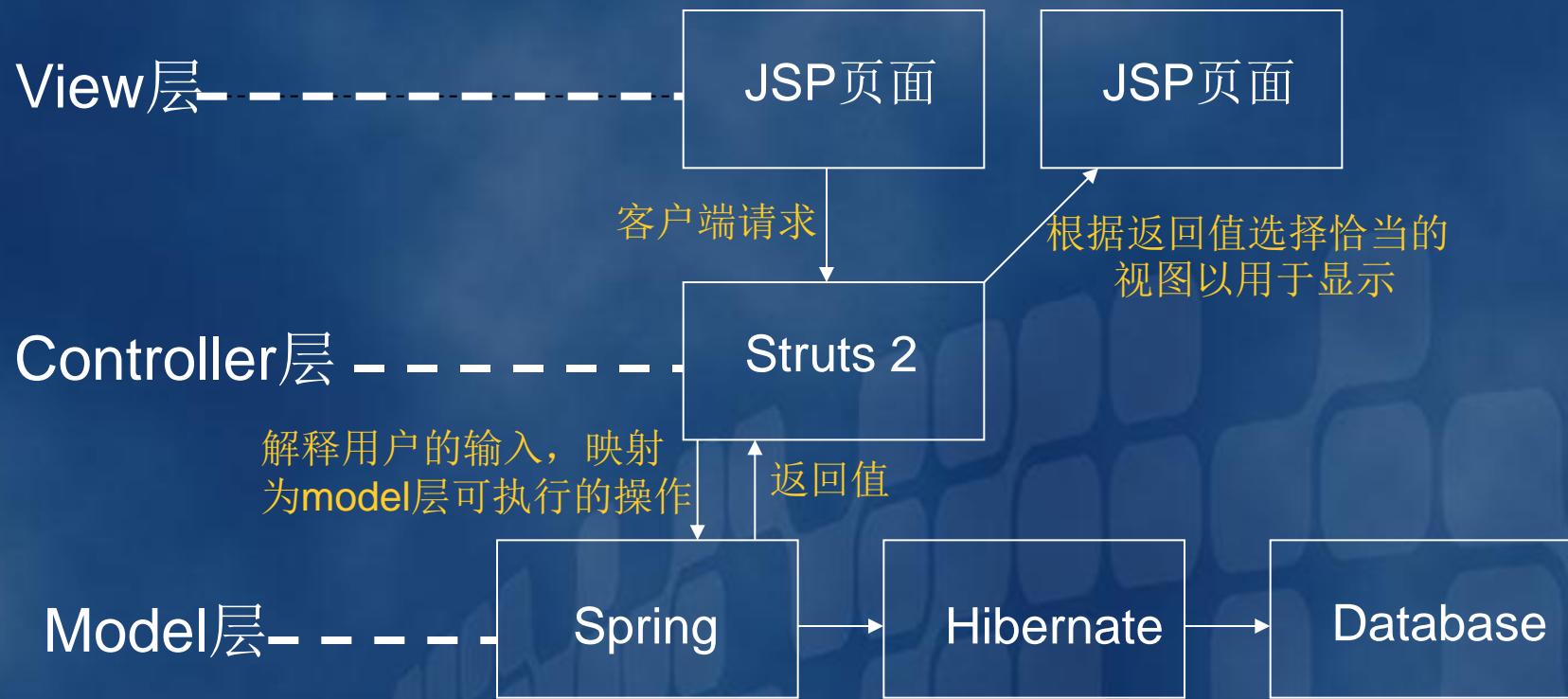


- 整体框架和业务层用**Spring**, 表示层用**Struts**, 而持久层用**Hibernate**。

# SSH:优秀的轻量级Java EE架构

- Struts+Spring+Hibernate
- Struts进行流程控制， Spring进行业务流转， Hibernate进行数据库操作的封装，这种新的开发模式让我们的开发更加方便、快捷！
- SSH无需专业的Java EE服务器,只需简单的Java Web服务器就可以运行。
- 大多数组件和Web服务器是开源和免费的。

# SSH结构分析



# 目录

- 组件与组件开发
- 主流Web开发技术及比较
- Java EE基本内容
- Java EE轻量级框架
- Java EE开发与部署环境配置
- 开发第一个Java EE应用

# Java EE开发运行环境配置

- JDK
- Tomcat服务器
- Eclipse
- MyEclipse

# JDK的安装与配置

通过设置系统环境变量，告诉Windows操作系统JDK的安装位置。下面具体介绍设置系统环境变量的方法。

① 设置系统变量JAVA\_HOME。右击【我的电脑】图标，选择【属性】→【高级系统设置】菜单项，弹出“环境变量”对话框，如图所示。



图 环境变量对话框

# JDK的安装与配置

在【系统变量】中单击【新建】按钮，弹出“新建系统变量”对话框，在“变量名”文本框中输入“JAVA\_HOME”，“变量值”文本框中输入JDK的安装路径“D:\Program Files\Java\jdk1.5”，如图所示，单击【确定】按钮完成配置。



图 新建系统变量对话框

# JDK的安装与配置

② 设置系统变量**Path**。选择【属性】→【高级】→【环境变量】菜单项，在【系统变量】中找到变量名为“**Path**”的变量，单击【编辑】按钮，在前面输入**JDK到bin的目录“D:\Program Files\Java\jdk1.5\bin;”**，如图所示，单击【确定】按钮完成配置。

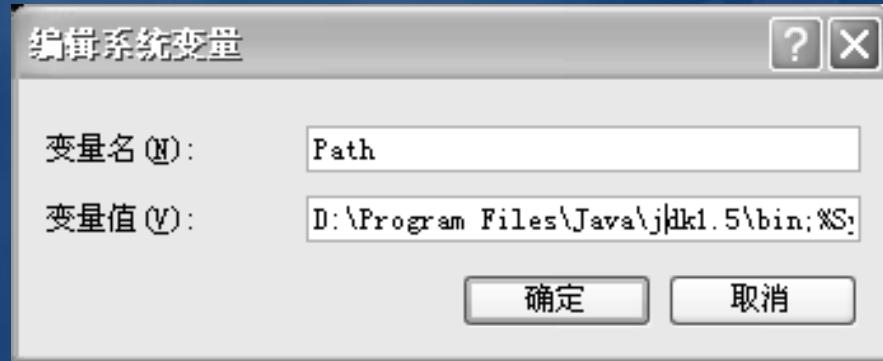


图 编辑系统

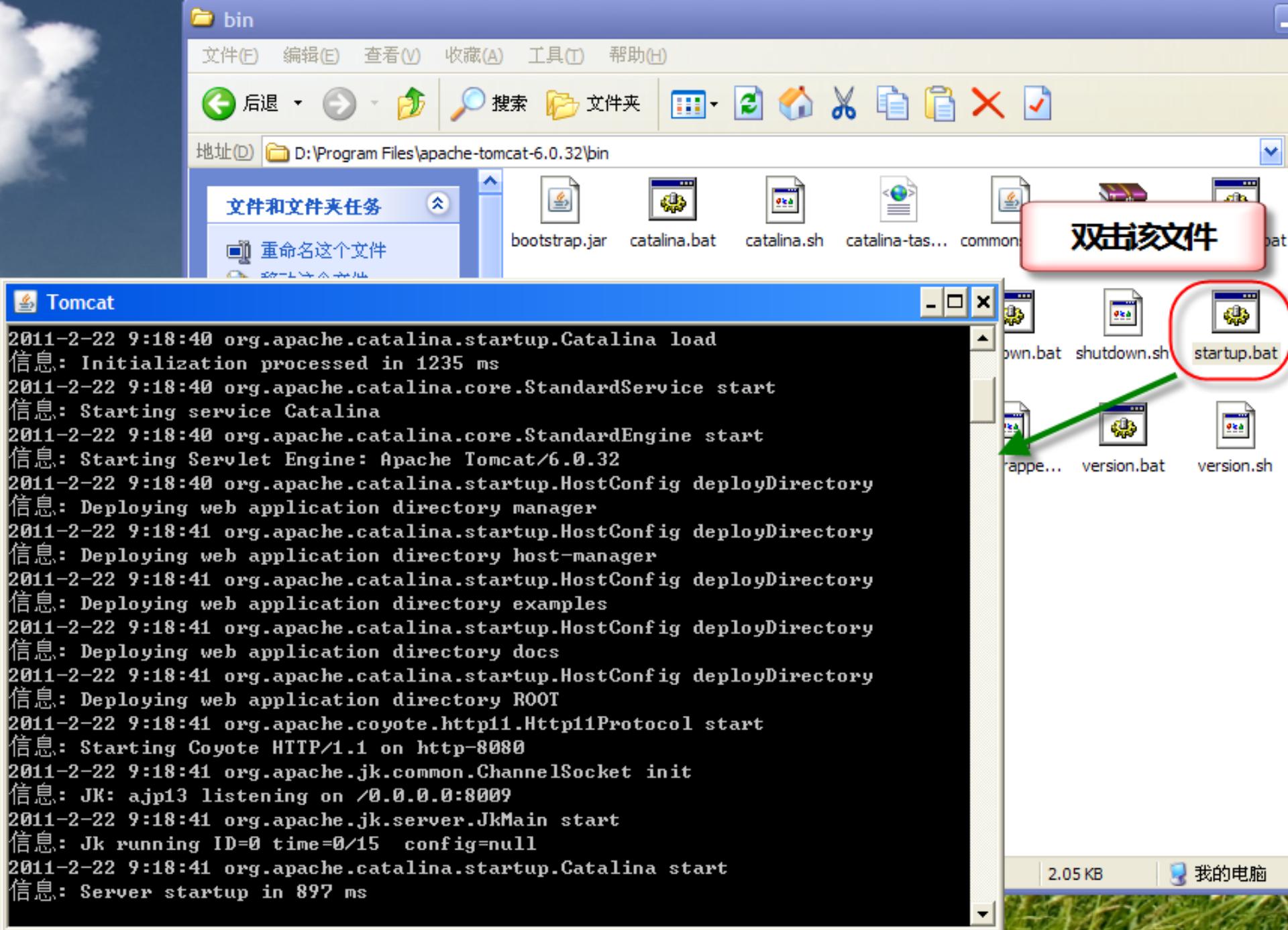
③ 设置系统变量**CLASSPATH**。与①同样的操作，不同的是变量名为“**CLASSPATH**”，变量值为“**.; D:\Program Files\Java\jdk1.5\lib\dt.jar; D:\Program Files\Java\jdk1.5\lib\tools.jar**”。

# 直接解压安装Tomcat服务器

- 下载Tomcat合适的版本；
- 解压缩下载的zip文件；
- 将解压缩后文件夹放在任意路径下；
- 设置Tomcat所需的环境变量；
- 启动Tomcat（双击安装路径下bin目录中的 startup.bat文件）；

# Tomcat 6文件结构

- **bin:**存放启动和关闭Tomcat的命令
- **conf:**存放Tomcat配置文件
- **lib:**存放Tomcat服务器的核心类库（jar文件），如果需要扩展Tomcat功能，也可将第三方类库复制到该路径下。
- **logs:**保存Tomcat每次运行后产生的日志
- **temp:**存放Web应用运行过程中生成的临时文件
- **webapps:**用于自动部署Web应用，将Web应用复制到该路径下，Tomcat会将该应用自动部署在容器中；
- **work:**保存web应用运行过程中，编译生成的class文件，可删除，每次启动tomcat将再次建立。





<http://localhost:8080/>



搜索

后退前进 停止 刷新 主页 恢复 收藏 历史 无痕



## Apache Tomcat



The Apache Software Foundation  
<http://www.apache.org/>

## *Administration*

## Status

## Tomcat Manager

*Documentation*

## Release Notes

## Change Log

## Tomcat Docum

[Tomcat Online](#)

[Home Page](#)  
[FAQ](#)  
[Bug Database](#)  
[Open Bugs](#)  
[Users Mailing List](#)  
[Developers Mailing List](#)  
[IRC](#)

## Miscellaneous

[Servlets Examples](#)  
[JSP Examples](#)  
[Sun's Java Server Pages Site](#)  
[Sun's Servlet Site](#)

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

`$CATALINA_HOME/webapps/ROOT/index.html`

where "\$CATALINA\_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the [Tomcat Documentation](#) for more detailed setup and administration information than is found in the INSTALL file.

**NOTE:** For security reasons, using the manager webapp is restricted to users with role "manager". Users are defined in \$CATALINA\_HOME/conf/tomcat-users.xml.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- [users@tomcat.apache.org](mailto:users@tomcat.apache.org) for general questions related to configuring and using Tomcat
  - [dev@tomcat.apache.org](mailto:dev@tomcat.apache.org) for developers working on Tomcat

Thanks for using Tomcat!

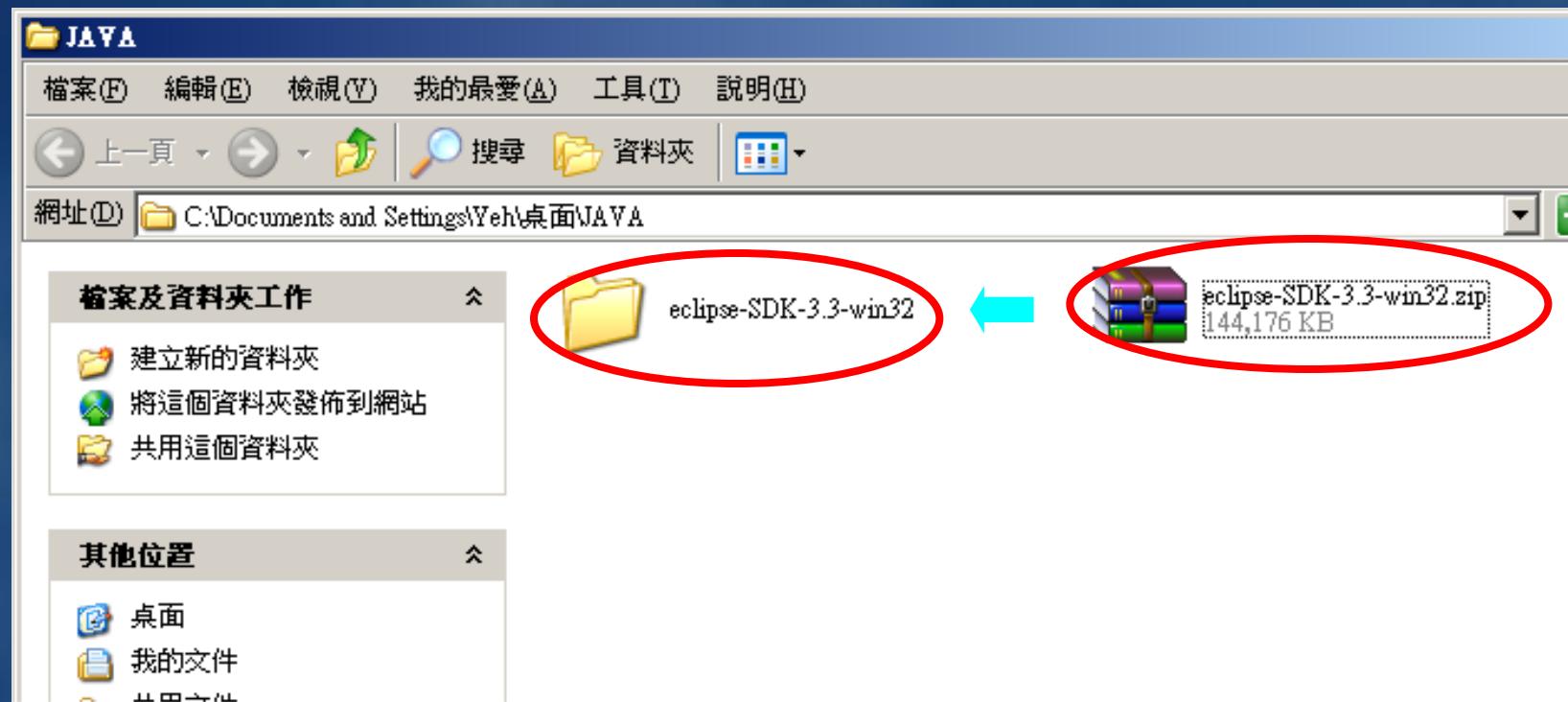
Powered by



Copyright © 1999-2011 Apache Software Foundation  
All Rights Reserved

# Eclipse Platform 安裝

- 解压缩所下载回来之 Eclipse
- 得到 Eclipse-SDK-3.3-win32 目錄



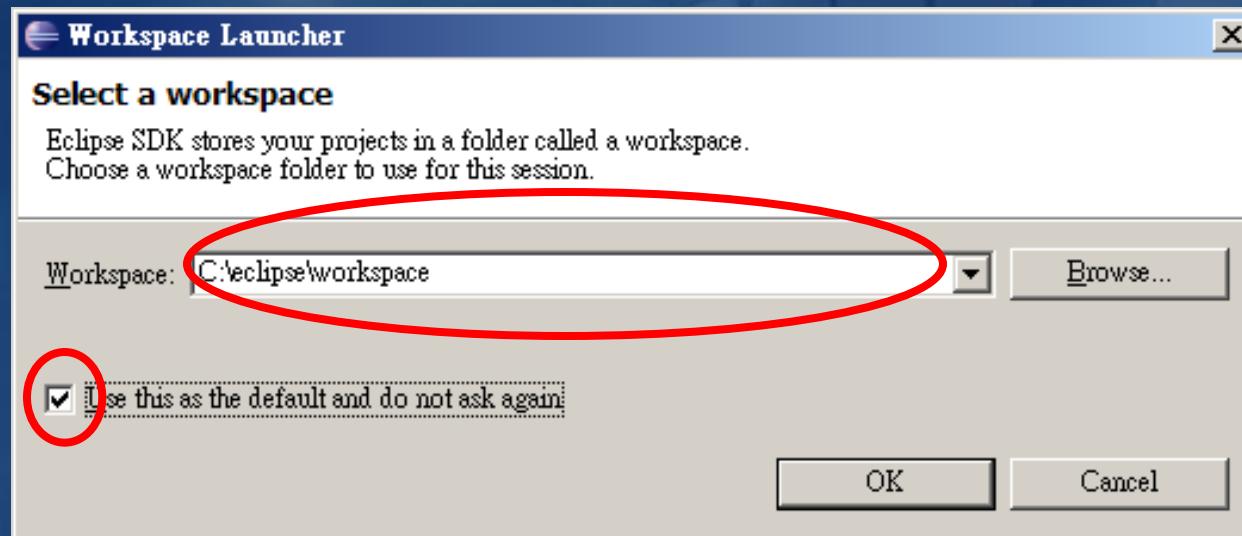
# Eclipse Platform 安裝

- 将该目录中的eclipse.exe，发送快捷方式到桌面



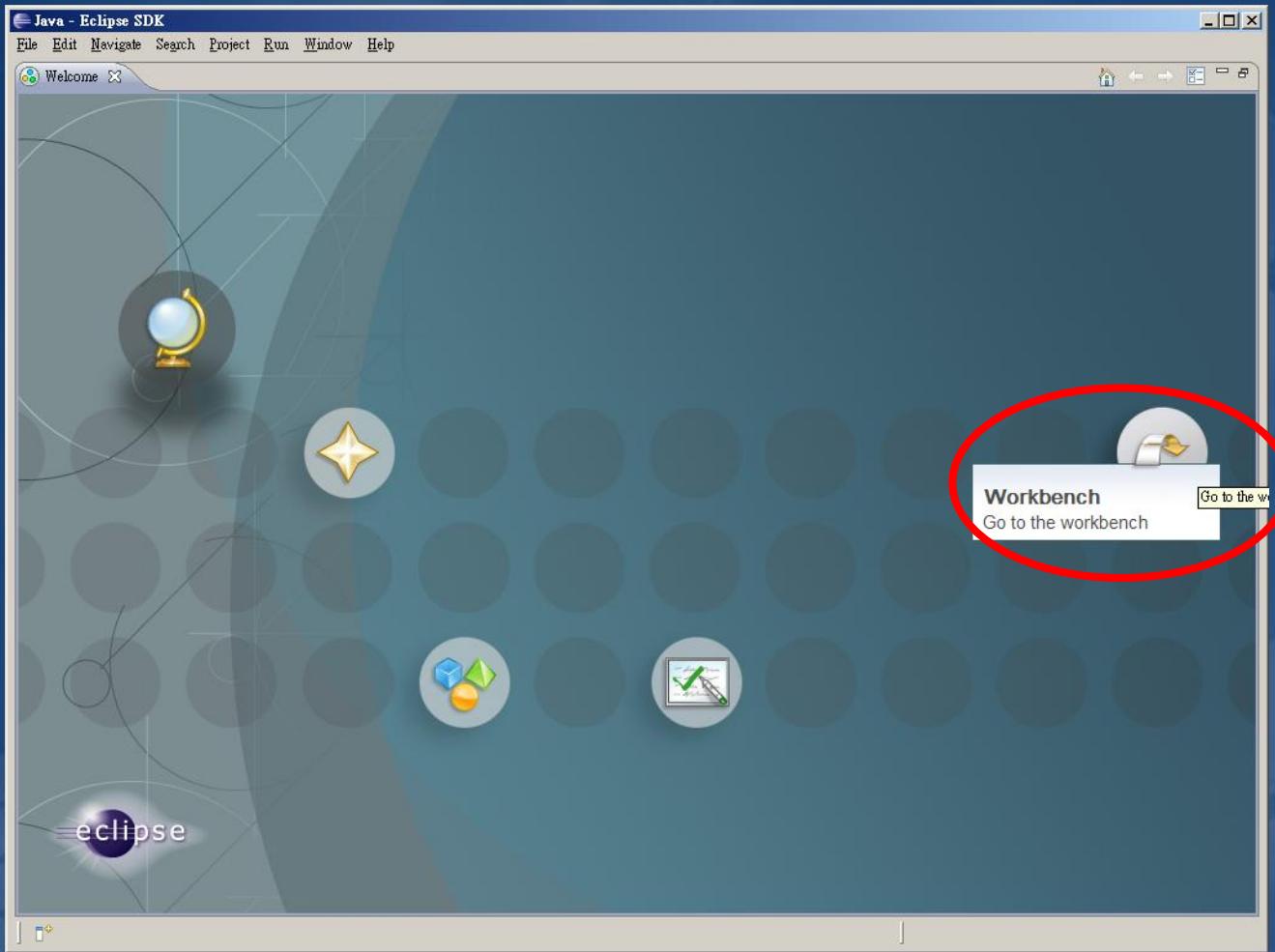
# Eclipse Platform 安裝

- 配置与运行Eclipse
  - 执行桌面快捷→Eclipse.exe
  - 指定工作区，用来存放项目



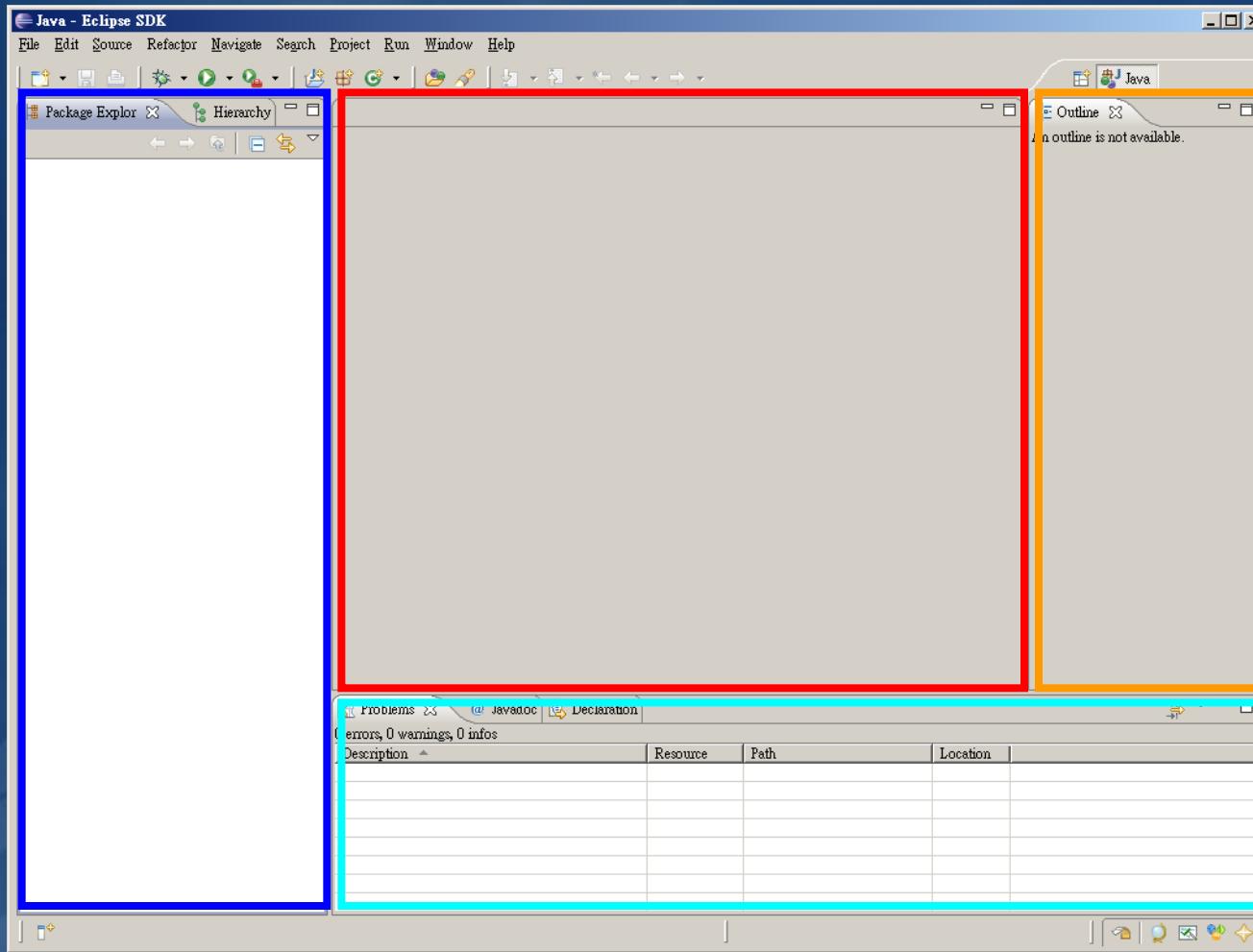
# Eclipse Platform 安裝

- 配置与运行Eclipse
  - 关闭Workbench进入编辑视图



# Eclipse Platform 安裝

- 配置与运行Eclipse环境
  - Eclipse编辑视图

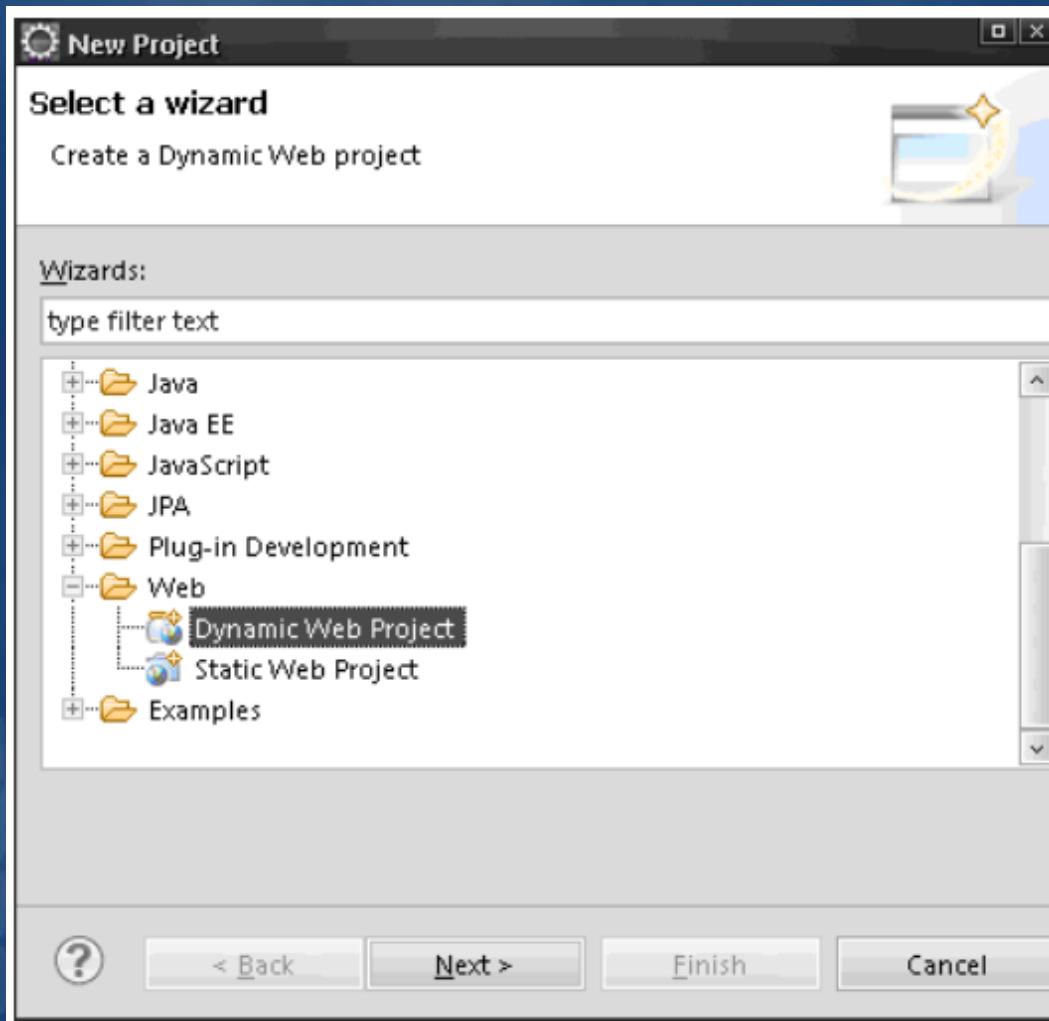


# 目录

- 组件与组件开发
- 主流Web开发技术及比较
- Java EE基本内容
- Java EE轻量级框架
- Java EE开发与部署环境配置
- 开发第一个Java EE应用

# Eclipse开发Java EE应用

在eclipse窗体的左侧的“Project Explorer”右键单击，  
New -> Project， 弹出“New Project”窗口。



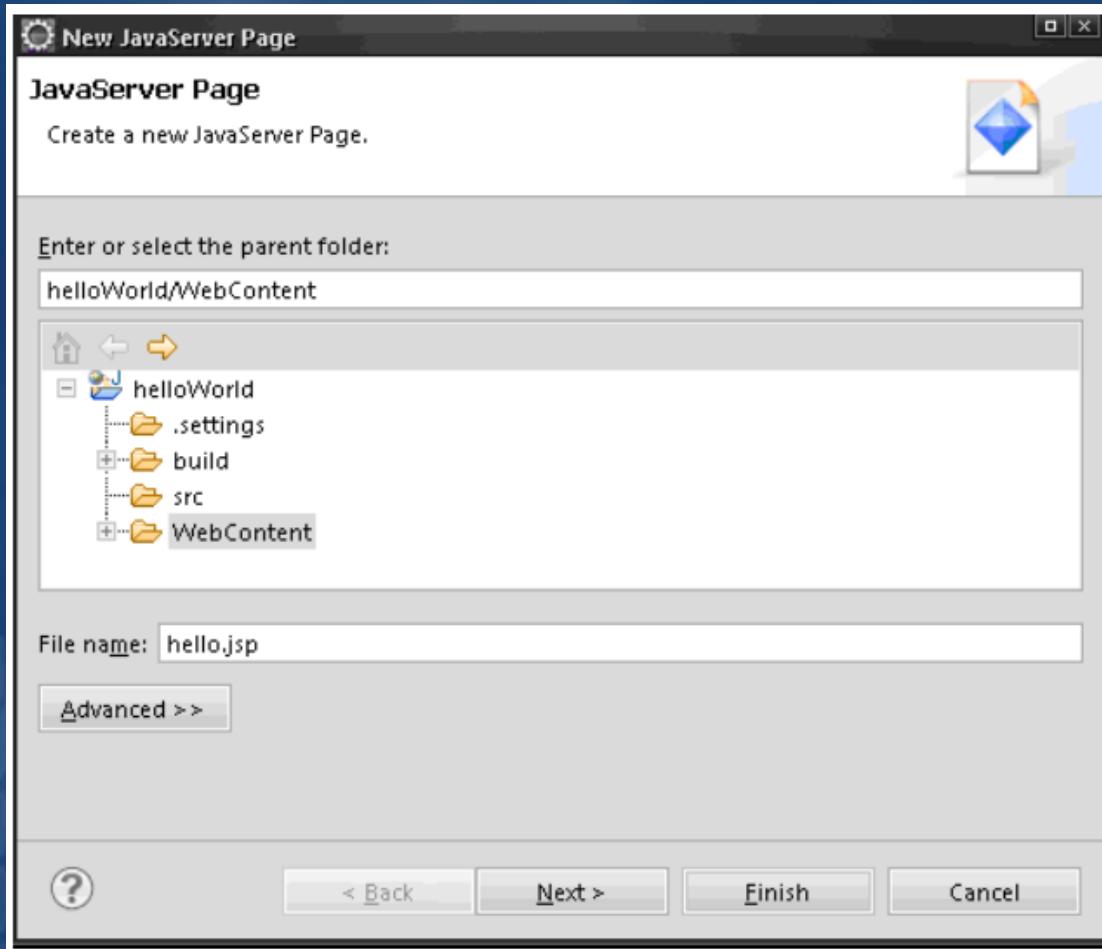
# Eclipse开发Java EE应用

选择“Dynamic Web Project”然后点击“Next”，在“Project Name”中输入想要的项目名称；输入后可直接点击“Finish”完成新项目的创建。



# Eclipse开发Java EE应用

在项目结构中的“WebContent”右键单击， New -> JSP，  
在弹出的“New JavaServer Page”窗口， 填写File name，  
hello.jsp， 直接点击“Finish”按钮。



# Eclipse开发Java EE应用

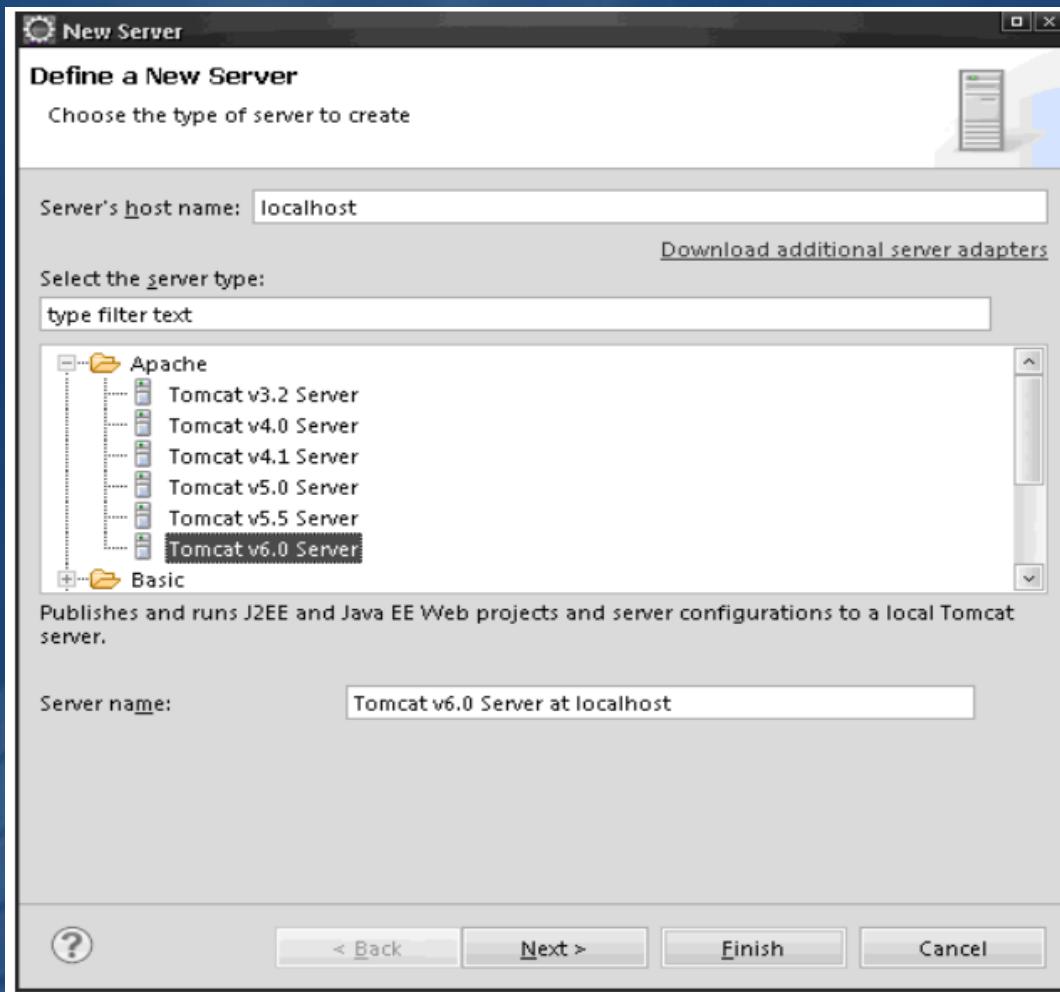
在Eclipse打开了hello.jsp，在里面写上“hello world！”，保存完成。

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    hello world!
</body>
</html>
```

# Eclipse开发Java EE应用

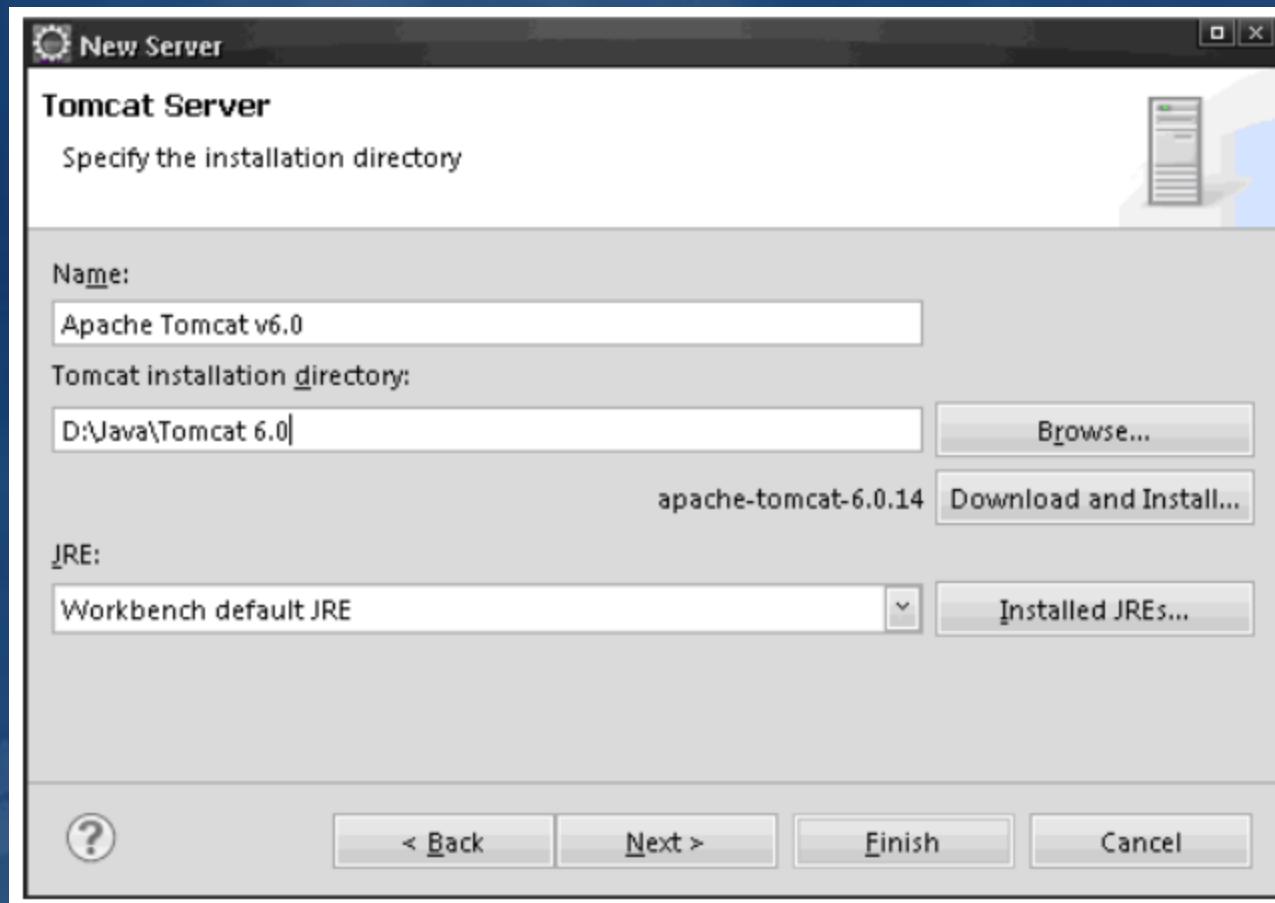
Servers”的卡片；点击它，使它显示在最前面。

在大片的空白处右键单击， New -> Server， 弹出“New Server”窗口， 展开Apache节点选择“Tomcat v6.0 Server”， Next。



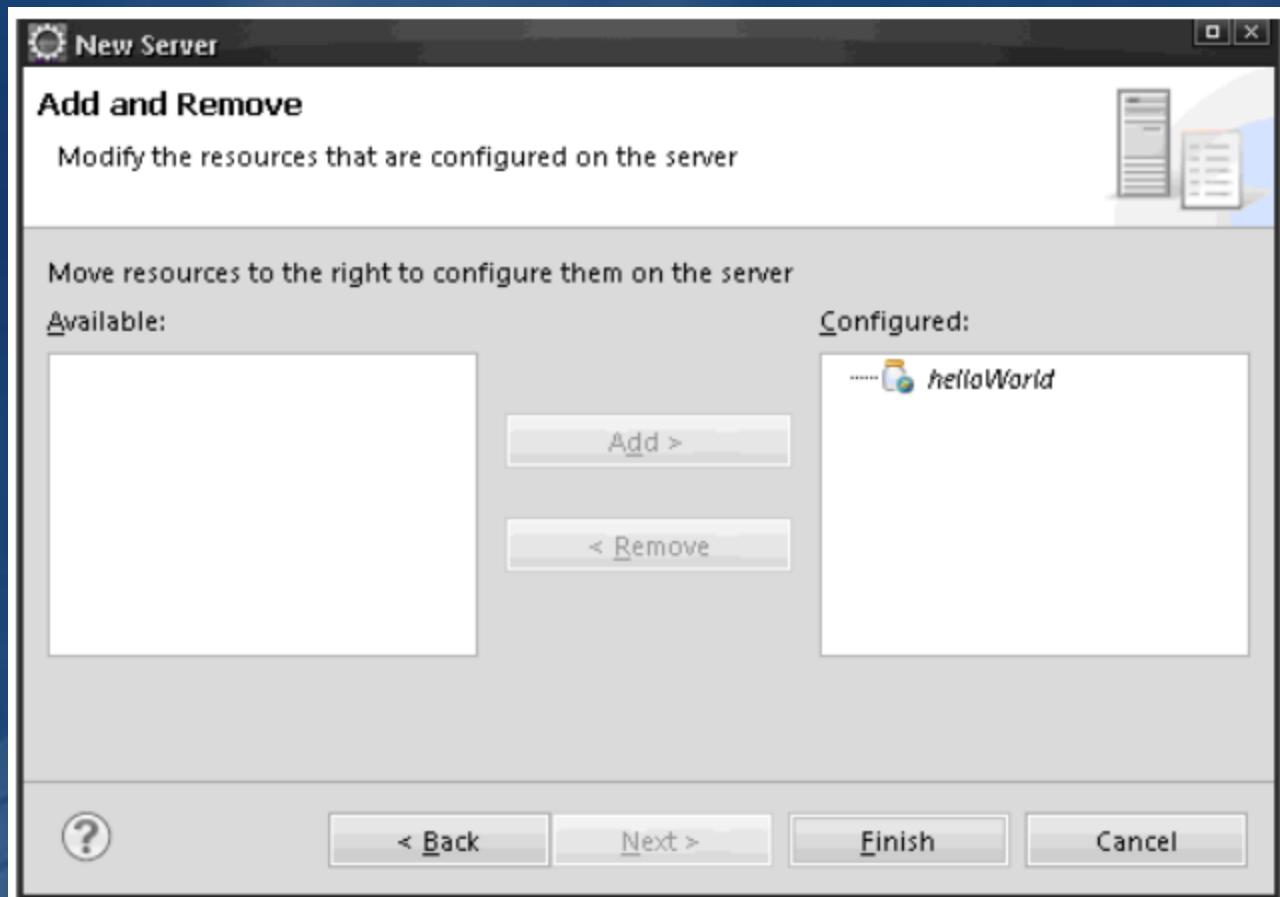
# Eclipse开发Java EE应用

点击“Browse...”选择Tomcat安装的路径，再持续点击“Next”。



# Eclipse开发Java EE应用

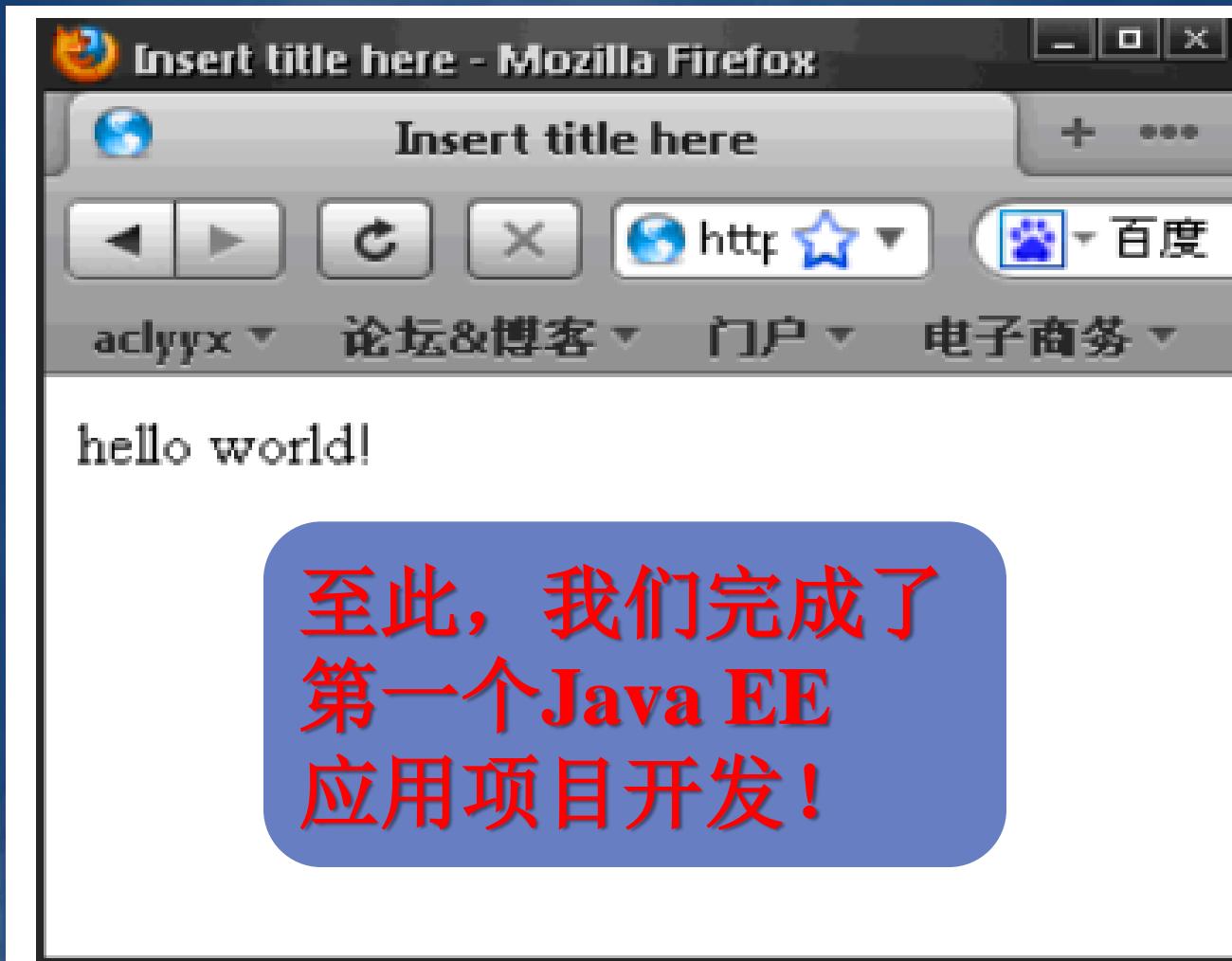
到了这个页面其实服务器就已经创建完成了，这页的功能只是向服务器中添加应用，否则服务器中没有应用可以运行，只能空跑。



# Eclipse开发Java EE应用

启动服务器。打开浏览器，输入地址

<http://localhost:8080/helloWorld/hello.jsp>，看到hello world。



# 课后实验

- **任务1：**下载和安装jdk 1.7，配置和测试JDK运行环境；
- **任务2：**下载和安装Tomcat 7.0.6，配置和运行Tomcat 7.0.6，用不同的方法启动和关闭Tomcat服务；
- **任务3：**下载和安装MyEclipse 10.0，配置MyEclipse运行环境；
- **任务4：**使用MyEclipse 10.0，创建一个Web应用项目，并使之在Tomcat服务器上配置运行。