



第 1 章 JSP概述

- 1.1 JSP技术概述 ✓
- 1.2 JSP技术特征 ✓
- 1.3 JSP的处理过程 ✓
- 1.4 JSP与其他服务器端脚本语言的比较 ✓
- 1.5 JSP开发环境搭建 ✓
- 1.6 JSP开发工具 ✓
- 1.7 JSP程序开发模式 ✓
- 1.8 第一个JSP应用 ✓



1.1 JSP技术概述

在了解JSP技术之前，首先需要了解与JSP技术相关的一些概念，这样有助于学习后面的内容。

- Java语言 ✓
- Servlet技术 ✓
- JavaBean技术 ✓
- JSP技术 ✓





1.2 JSP技术特征

- 跨平台 ✓
- 分离静态内容和动态内容 ✓
- 可重复使用的组件 ✓
- 沿用了Java Servlet的所有功能 ✓
- 预编译 ✓



1.4 JSP与其他服务器端脚本语言的比较

- CGI ✓
- ASP ✓
- PHP ✓
- ASP.NET ✓





1.5 JSP开发环境搭建

- JSP的运行环境 ✓
- JDK的安装与配置 ✓
- Tomcat的安装与启动 ✓





1.6 JSP开发工具

Eclipse是一个基于**Java**的、开放源码的、可扩展的应用开发平台，它为编程人员提供了一流的**Java**集成开发环境（**Integrated Development Environment, IDE**）。它是一个可以用于构建集成**Web**和应用程序开发工具的平台，其本身并不会提供大量的功能，而是通过插件来实现程序的快速开发功能。

Eclipse是一个成熟的可扩展的体系结构，它的价值体现在为创建可扩展的开发环境提供了一个开放源代码的平台。这个平台允许任何人构建与环境或其他工具无缝集成的工具，而工具与**Eclipse**无缝集成的关键是插件。



1.6 JSP开发工具

Eclipse还包括插件开发环境（**Plug-in Development Environment, PDE**），**PDE**主要针对那些希望扩展**Eclipse**的编程人员而设定的。这也正是**Eclipse**最具魅力的地方。通过不断地集成各种插件，**Eclipse**的功能也在不断地扩展，以便支持各种不同的应用。

虽然**Eclipse**是针对**Java**语言而设计开发的，但是它的用途并不局限于**Java**语言，通过安装不同的插件，**Eclipse**还可以支持诸如**C/C++**、**PHP**、**COBOL**等编程语言。



1.6 JSP开发工具

- Eclipse的安装与启动 ✓
- 安装MyEclipse插件 ✓
- Eclipse3.2快捷键 ✓
- 应用Eclipse开发简单的JSP程序 ✓





1.7 JSP程序开发模式

- 单纯的JSP页面编程 ✓
- JSP+JavaBean编程 ✓
- JSP+Servlet+JavaBean编程 ✓
- MVC模式 ✓





单纯的JSP编程

在单纯的JSP编程模式下，通过应用JSP中的脚本标志，可直接在JSP页面中实现各种功能。虽然这种模式很容易实现，但是其缺点也非常明显。因为将大部分的Java代码与HTML代码混淆在一起，会给程序的维护和调试带来很多的困难，而且对于整个程序的结构更是无从谈起。这就好比规划管理一个大的企业，如果将负责不同任务的所有员工都安排在一起工作，势必会造成公司秩序混乱、不易管理等许多的隐患。所以说，单纯的JSP页面编程模式是无法应用到大型、中型甚至小型的JSP Web应用程序开发中。

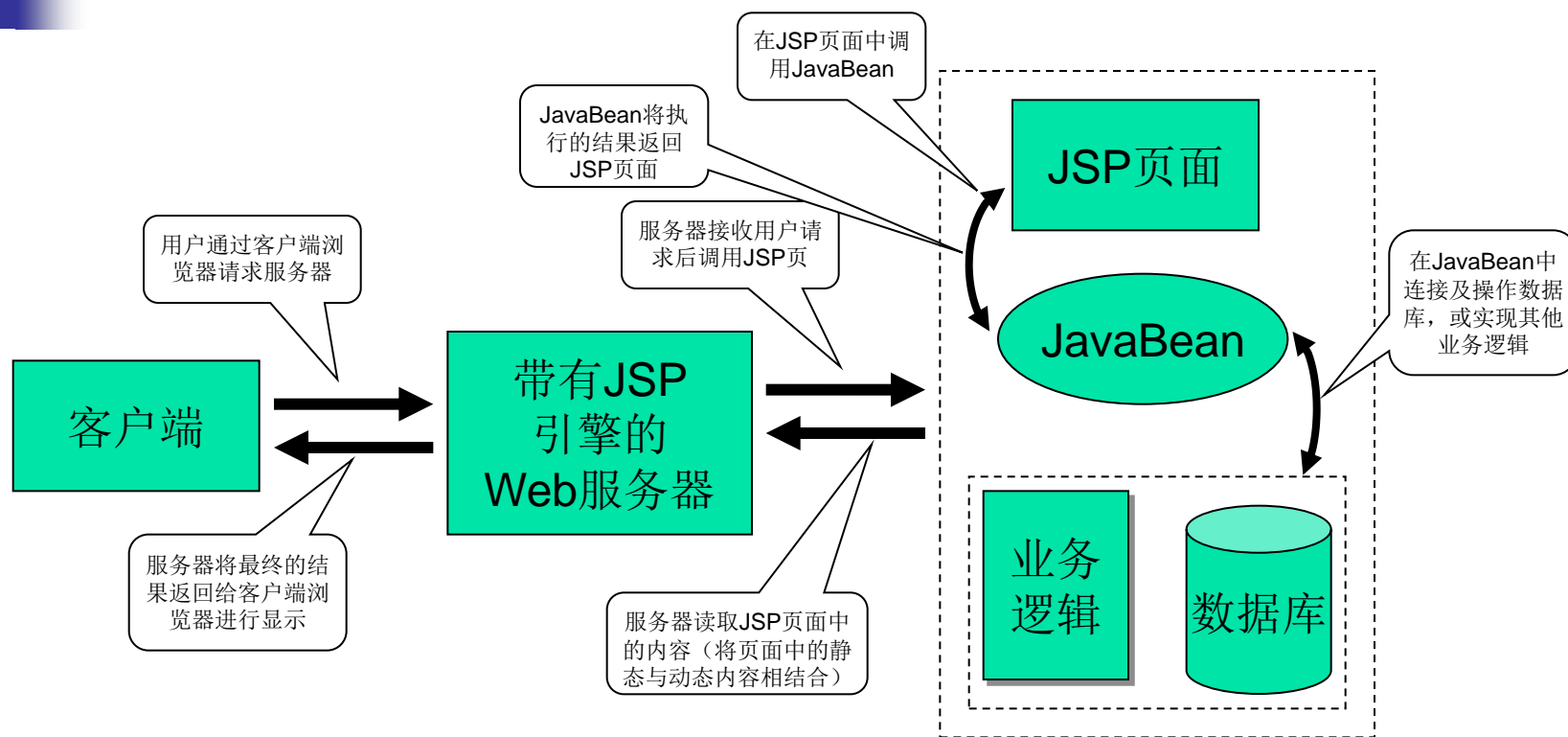




JSP+JavaBean编程

该模式是JSP程序开发经典设计模式之一，适合小型或中型网站的开发。利用JavaBean技术，可以很容易地完成一些业务逻辑上的操作，例如数据库的连接、用户登录与注销等。JavaBean是一个遵循了一定规则的Java类，在程序的开发中，将要进行的业务逻辑封装到这个类中，在JSP页面中通过动作标签来调用这个类，从而执行这个业务逻辑。此时的JSP除了负责部分流程的控制外，大部分用来显示页面，而JavaBean则负责业务逻辑的处理。可以看出，该模式具有一个比较清晰的程序结构，在JSP技术的起步阶段，JSP+JavaBean设计模式曾被广泛应用。下面将通过一个流程图说明该模式对客户端的请求进行处理的过程。

JSP+JavaBean编程



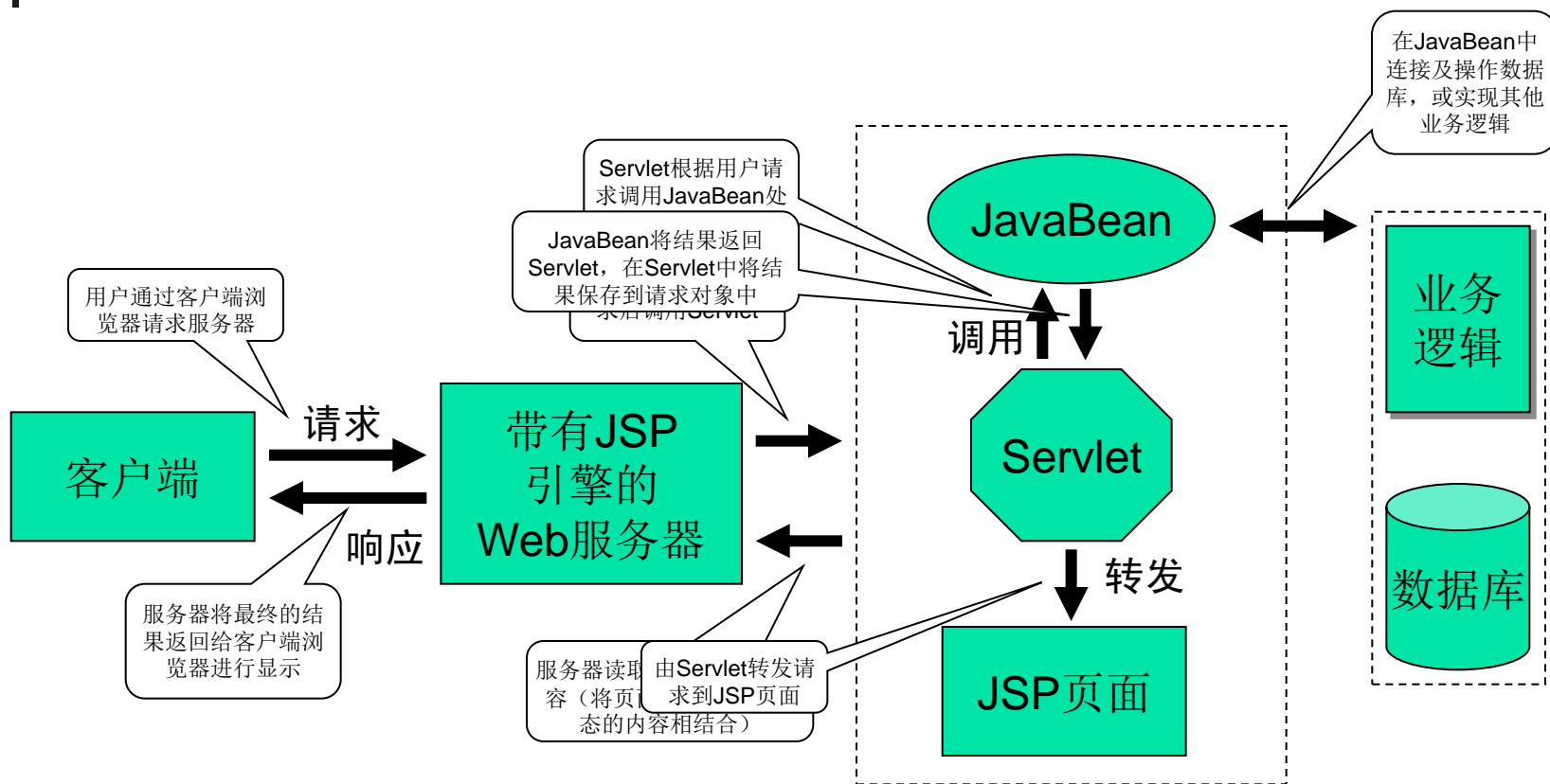


JSP+Servlet+JavaBean编程

JSP+JavaBean设计模式虽然已经将网站的业务逻辑和显示页面进行分离，但这种模式下的JSP不但要进行程序中大部分的流程控制，而且还要负责页面的显示，所以仍然不是一种理想的设计模式。

在JSP+JavaBean设计模式的基础上加入Servlet来实现程序中的控制层，是一个很好的选择。在这种模式中，由Servlet来执行业务逻辑并负责程序的流程控制，JavaBean组件实现业务逻辑，充当着模型的角色，JSP用于页面的显示。可以看出这种模式使得程序中的层次关系更明显，各组件的分工也非常明确。下面将通过一个流程图说明该模式对客户端的请求进行处理的过程。

JSP+Servlet+JavaBean编程





JSP+Servlet+JavaBean编程

但JSP+Servlet+JavaBean模式同样也存在缺点。该模式遵循了MVC设计模式，MVC只是一个抽象的设计概念，它将待开发的应用程序分解为三个独立的部分：模型（Model）、视图（View）和控制器（Controller）。虽然用来实现MVC设计模式的技术可能都是相同的，但各公司都有自己的MVC架构。也就是说，这些公司用来实现自己的MVC架构所应用的技术可能都是JSP、Servlet与JavaBean，但他们的流程及设计却是不同的，所以工程师需要花更多的时间去了解。从项目的开发观点上来说，因为需要设计MVC各对象之间的数据交换格式与方法，所以会需要花费更多的时间在系统的设计上。



JSP+Servlet+JavaBean编程

使用JSP+Servlet+JavaBean模式进行项目开发时，可以选择一个实现了MVC模式的现成的框架，在此下进行开发，大大节省了开发时间，会取得事半功倍的效果。目前已有许多可以使用的现成的MVC框架，例如Struts框架。

JSP+JavaBean编程与JSP+Servlet+JavaBean编程，是JSP开发中的两种经典设计模式。





MVC模式

MVC（Model-View-Controller，模型—视图—控制器）是一种程序设计概念，它同时适用于简单的和复杂的程序。使用该模式可将待开发的应用程序分解为3个独立的部分：模型、视图和控制器。提出这种设计模式主要是因为应用程序中用来完成任务的代码——模型（也称为“业务逻辑”）通常是程序中相对稳定的部分，并且会被重复使用，而程序与用户进行交互的页面——视图，却是经常改变的。如果因需要更新页面而不得不对业务逻辑代码进行改动，或者要在不同的模块中应用到相同的功能而重复地编写业务逻辑代码，不仅降低了整体程序开发的进程，而且会使程序变得难以维护。因此，将业务逻辑代码与外观呈现分离，将会更容易地根据需求的改变来改进程序。



MVC模式

MVC模式中的**Model**（模型）指的是业务逻辑的代码，是应用程序中真正用来完成任务的部分。

View（视图），实际上就是程序与用户进行交互的界面，用户可以看到它的存在。视图可以具备一定的功能并应遵守对其的约束，在视图中不应包含对数据处理的代码，即业务逻辑代码。

Controller（控制器），主要控制用户请求并作出响应。它根据用户的请求选择模型或修改模型，并决定返回怎样的视图。



Eclipse3.2开发工具中常用快捷键

名 称	功 能	名 称	功 能
F3	跳转到类或变量的声明	Ctrl + D	删除光标所在行的代码
Alt + 上下方向键	将选中的一行或多行向上或向下移动	Alt + /	代码提示
Ctrl + K	将光标停留在变量上，按Ctrl+K键可查找下一个同样的变量	Ctrl + O	打开视图的小窗口
Alt + 左右方向键	跳到前一次或/后一次的编辑位置，在代码跟踪时用的比较多	Ctrl + W	关闭单个窗口
Ctrl + /	注释或取消注释	Ctrl + 鼠标单击	可以跟踪方法和类的源码
Ctrl + 鼠标停留	可以显示方法和类的源码	Ctrl + Shift + K	和Ctrl+K键查找的方向相反
Ctrl + M	将当前视图最大化	Ctrl + Shift + O	快速地导入类的路径
Ctrl + I	光标停留在某变量，按Ctrl+I键，可提供快速实现的重构方法。选中若干行，按Ctrl+I键可将此段代码放入for, while, if, do或try等代码块中	Ctrl + Shift + F	代码格式化。如果将代码进行部分选择，仅对所选代码进行格式化
Ctrl + Q	回到最后编辑的位置	Ctrl + Shift + X	将所选字符转为大写
Ctrl + F6	切换窗口	Ctrl + Shift + Y	将所选字符转为小写
Ctrl + Shift + /	注释代码块	Ctrl + Shift + D	在debug模式里显示变量值
Ctrl + Shift + \	取消注释代码块	Ctrl + Shift + T	查找工程中的类
Ctrl + Shift + M	导入未引用的包	Ctrl + Alt + Down	复制光标所在行至其下一行
双击左括号（小括号，中括号，大括号）	将选择括号内的所有内容		





JSP的运行环境

使用**JSP**进行开发，需要具备以下对应的运行环境：**Web**浏览器、**Web**服务器、**JDK**开发工具包以及数据库。下面分别介绍这些环境。

1. **Web**浏览器

浏览器主要用于客户端用户访问**Web**应用的工具，与开发**JSP**应用不存在很大的关系，所以开发**JSP**对浏览器的要求并不是很高，任何支持**HTML**的浏览器都可以。



JSP的运行环境

2. Web服务器

Web服务器是运行及发布Web应用的大容器，只有将开发的Web项目放置到该容器中，才能使网络中的所有用户通过浏览器进行访问。开发JSP应用所采用的服务器主要是Servlet兼容的Web服务器，比较常用的有BEA WebLogic、IBM WebSphere和Apache Tomcat等。



JSP的运行环境

Weblogic是BEA公司的产品，它又分为WebLogic Server、WebLogic Enterprise和WebLogic Portal系列，其中WebLogic Server的功能特别强大，它支持企业级的、多层次的和完全分布式的Web应用，并且服务器的配置简单、界面友好，对于那些正在寻求能够提供Java平台所拥有的一切的应用服务器的用户来说，WebLogic是一个十分理想的选择。



JSP的运行环境

Tomcat服务器最为流行，它是Apache-Jakarta开源项目中的一个子项目，是一个小型的、轻量级的、支持JSP和Servlet技术的Web服务器，它已经成为学习开发JSP应用的首选。目前Tomcat的最新版本为apache-tomcat-6.0.16。



JSP的运行环境

3. JDK

JDK（Java Develop Kit，Java开发工具包）包括运行Java程序所必须的JRE环境及开发过程中常用的库文件。在使用JSP开发网站之前，首先必须安装JDK，目前JDK的最新版本为jdk1.6.0_03。



JSP的运行环境

4. 数据库

任何项目的开发几乎都需要使用数据库，数据库用来存储项目中需要的信息。根据项目的规模，应采用合适的数据库。如大型项目可采用Oracle数据库，中型项目可采用Microsoft SQL Server或MySQL数据库，小型项目可采用Microsoft Access数据库。

Microsoft Access数据库的功能远比不上Microsoft SQL Server和MySQL强大，但它具有方便、灵活的特点，对于一些小型项目来说是比较理想的选择。



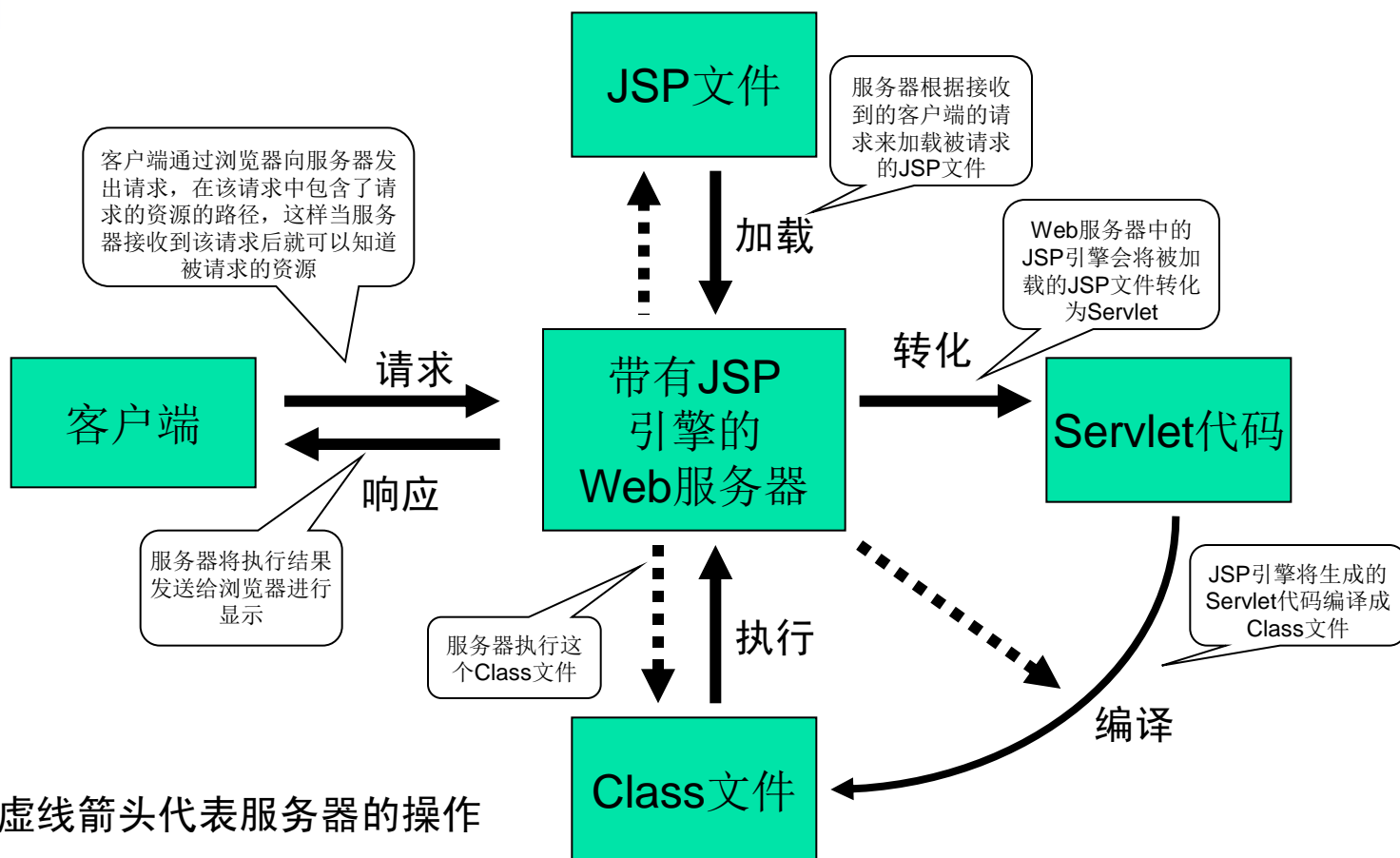


1.3 JSP的处理过程

当客户端浏览器向服务器发出请求访问一个JSP页面时，服务器根据该请求加载相应的JSP页面，并对该页面进行编译，然后执行。

JSP的具体处理过程如下图所示。

1.3 JSP的处理过程





1.3 JSP的处理过程

从前面的介绍中，可以知道JSP文件被JSP引擎进行转换后，又被编译成了Class文件，最终由服务器通过执行这个Class文件来对客户端的请求进行响应。


但并不是每次请求都需要重复进行这样的处理。当服务器第一次接收到对某个页面的请求时，JSP引擎就开始进行上述的处理过程，将被请求的JSP文件编译成Class文件。在后续对该页面再次进行请求时，若页面没有进行任何改动，服务器只需直接调用Class文件执行即可。所以当某个JSP页面第一次被请求时，会有一些延迟，而再次访问时会感觉快了很多。如果被请求的页面经过修改，服务器将会重新编译这个文件，然后执行。





CGI


CGI（**Common Gateway Interface**）即通用网关接口。是最早用来创建动态网页的一种技术，它可以使浏览器与服务器之间产生互动关系。它允许使用不同的语言来编写适合的**CGI**程序，该程序被放在**Web**服务器上运行。当客户端发出请求给服务器时，服务器根据客户请求建立一个新的进程来执行指定的**CGI**程序，并将执行结果以网页的类型传输到客户端的浏览器上进行显示。**CGI**可以说是当前应用程序的基础技术，但这种技术编制方式比较困难而且效率低下，因为每次页面被请求的时候，都要求服务器重新将**CGI**程序编译成可执行的代码。在**CGI**中使用最为常见的语言为**C/C++**、**Java**和**Perl**（**Practical Extraction and Report Language**，文件分析报告语言）。





ASP


ASP (Active Server Page) 是一种使用很广泛的开发动态网站的技术。它通过在页面代码中嵌入**VBScript**或**JavaScript**脚本语言来生成动态的内容，在服务器端必须安装了适当的解释器后，才可以通过调用此解释器来执行脚本程序，然后将执行结果与静态内容部分结合并传送到客户端浏览器上。对于一些复杂的操作，**ASP**可以调用存在于后台的**COM**组件来完成，所以说**COM**组件无限地扩充了**ASP**的能力；正因如此依赖本地的**COM**组件，使得**ASP**主要用于**Windows**平台中。**ASP**存在很多优点，简单易学，并且**ASP**是与微软的**IIS**捆绑在一起，在安装**Windows 2000**、**Windows XP**的同时安装上**IIS**，就可以运行**ASP**应用程序了。





PHP

PHP来自于Personal Home Page一词，但现在的PHP已经不再表示名词的缩写，而是一种开发动态网页技术的名称。PHP语法类似于C，并且混合了Perl、C++和Java的一些特性。它是一种开源的Web服务器脚本语言，与ASP和JSP一样可以在页面中加入脚本代码来生成动态内容。对于一些复杂的操作可以封装到函数或类中，在PHP中提供了许多已经定义好的函数，例如提供的标准的数据库接口，使得数据库连接方便，扩展性强。PHP可以被多个平台支持，主要被广泛应用于UNIX/Linux平台。由于PHP本身的代码对外开放，经过许多软件工程师的检测，因此到目前为止该技术具有公认的安全性能。





ASP.NET

ASP.NET也是一种建立动态Web应用程序的技术，它是.NET框架的一部分，可以使用任何.NET兼容的语言，如Visual Basic.NET，C#，J#等来编写ASP.NET应用程序。这种ASP.NET页面（Web Forms）编译后可以提供比脚本语言更出色的性能表现。Web Forms允许在网页基础上建立强大的窗体。当建立页面时，可以使用ASP.NET服务端控件来建立常用的UI元素，并对它们编程来完成一般的任务。这些控件允许开发者使用内建可重用的组件和自定义组件来快速建立Web Form，使代码简单化。





Java语言

Java语言是由Sun公司于1995年推出的编程语言，一经推出，就赢得了业界的一致好评，并受到了广泛关注。Java语言适用于Internet环境，目前已成为开发Internet应用的主要语言之一。它具有简单、面向对象、可移植性、分布性、解释器通用性、稳健、多线程、安全和高性能等优点。其中最重要的就是实现了跨平台运行，这使得应用Java开发的程序可以方便地移植到不同的操作系统中运行。



Java语言


Java是完全面向对象的编程语言，它的语法规则和**C++**类似，但**Java**语言对**C++**进行了简化和提高。例如，**C**语言中的指针和多重继承通常会使程序变得复杂，而**Java**通过接口取代了多重继承，并取消了指针、内存的申请和释放等影响系统安全的部分。



Java语言

在Java语言中，最小的单位是类，不允许在类外面定义变量和方法，所以就不存在所谓的“全局变量”这一概念。在Java类中定义的变量和方法分别称为成员变量和成员方法，其中成员变量也叫做类的属性。在定义这些类的成员时，需要通过权限修饰符来声明它们的使用范围。

Java语言编写的程序应被保存为后缀名为.java的文件，然后编译成后缀名为.class的字节码文件，最终通过执行该字节码文件执行Java程序。





Servlet技术

Servlet是在**JSP**之前就存在的运行在服务端的一种**Java**技术，它是用**Java**语言编写的服务器端程序，**Java**语言能够实现的功能，**Servlet**基本上都可以实现（除图形界面外）。**Servlet**主要用于处理**Http**请求，并将处理的结果传递给浏览器生成动态**Web**页面。**Servlet**具有可移植（可在多种系统平台和服务器平台下运行）、功能强大、安全、可扩展和灵活等优点。



Servlet技术

在JSP中用到的Servlet通常都继承自 `javax.servlet.http.HttpServlet` 类，在该类中实现了用来处理Http请求的大部分功能。

JSP是在Servlet的基础上开发的一种新的技术，所以JSP与Servlet有着密不可分的关系。JSP页面在执行过程中会被转换为Servlet，然后由服务器执行该Servlet。





JavaBean技术

JavaBean是根据特殊的规范编写的普通的Java类，可称它们为“独立的组件”。每一个JavaBean实现一个特定的功能，通过合理地组织具有不同功能的JavaBean，可以快速地生成一个全新的应用程序。如果将这个应用程序比作一辆汽车，那么程序中的JavaBean就好比组成这辆汽车的不同零件。对于程序开发人员来说，JavaBean的最大优点就是充分提高了代码的可重用性，并且对程序的后期维护和扩展起到了积极的作用。



JavaBean技术

JavaBean可按功能划分为可视化和不可可视化两种。可视化JavaBean主要应用在图形界面编程的领域中，在JSP中通常应用不可可视化JavaBean，应用这种JavaBean可用来封装各种业务逻辑，例如连接数据库、获取当前时间等。这样，当在开发程序的过程中需要连接数据库或实现其他功能时，就可直接在JSP页面或Servlet中调用实现该功能的JavaBean来实现。

通过应用JavaBean，可以很好地将业务逻辑和前台显示代码分离，这大大提高了代码的可读性和易维护性。





JSP技术

Java Server Pages简称JSP，是由Sun公司倡导，与多个公司共同建立的一种技术标准，它建立在Servlet之上。应用JSP，程序员或非程序员可以高效率地创建Web应用程序，并使得开发的Web应用程序具有安全性高、跨平台等优点。

JSP是运行在服务器端的脚本语言之一，与其他的服务端脚本语言一样，是用来开发动态网页的一种技术。



JSP技术

JSP页面由传统的HTML代码和嵌入到其中的Java代码组成。当用户请求一个JSP页面时，服务器会执行这些Java代码，然后将结果与页面中的静态部分相结合返回给客户端浏览器。JSP页面中还包含了各种特殊的JSP元素，通过这些元素可以访问其他的动态内容并将它们嵌入到页面中，例如访问JavaBean组件的<jsp:useBean>动作元素。程序员还可以通过编写自己的元素来实现特定的功能，开发出更为强大的Web应用程序。



JSP技术

JSP是在Servlet的基础上开发的技术，它继承了Java Servlet的各项优秀功能。而Java Servlet是作为Java的一种解决方案，在制作网页的过程中，它继承了Java的所有特性。因此JSP同样继承了Java技术的简单、便利、面向对象、跨平台和安全可靠等优点，比起其他服务器脚本语言，JSP更加简单、迅速和有力。在JSP中利用JavaBean和JSP元素，可以有效地将静态的HTML代码和动态数据区分开来，给程序的修改和扩展带来了很大方便。





跨平台

JSP是以Java为基础开发的，所以它不仅可以沿用Java强大的API功能，而且不管是在何种平台下，只要服务器支持JSP，就可以运行使用JSP开发的Web应用程序，体现了它的跨平台、跨服务器的特点。例如在Windows NT下的IIS通过JRUN或ServletExec插件就能支持JSP。如今最流行的Web服务器Apache同样能够支持JSP，而且Apache支持多种平台，从而使得JSP可以在多个平台上运行。



跨平台

在数据库操作中，因为JDBC同样是独立于平台的，所以在JSP中使用Java API提供的JDBC来连接数据库时，就不用担心平台变更时的代码移植问题。正是因为Java的这种特征，使得应用JSP开发的Web应用程序能够很简单地运用到不同的平台上。





分离静态内容和动态内容

在前面提到的Java Servlet，对于开发Web应用程序而言是一种很好的技术。但同时面临着一个问题：所有的内容必须在Java代码中来完成，包括HTML代码同样要嵌入到程序代码中来生成静态的内容。这使得即使因HTML代码出现的小问题，也需要有熟悉Java Servlet的程序员来解决。



分离静态内容和动态内容

JSP弥补了Java Servlet在工作中的不足。使用JSP，程序员可以使用HTML或XML标记来设计和格式化静态内容，并通过JSP标记及JavaBean组件来制作动态内容。服务器将执行JSP标记和小脚本程序，并将结果与页面中的静态部分结合后以HTML页面的形式发送给客户端浏览器。程序员可以将一些业务逻辑封装到JavaBean组件中，Web页面的设计人员可以利用程序员开发的JavaBean组件和JSP标记来制作出动态页面，而且不会影响到内容的生成。

将静态内容与动态内容的明确分离，是以Java Servlet开发Web应用发展为以JSP开发Web应用的重要因素之一。





可重复使用的组件

JavaBean组件是JSP中不可缺少的重要组成部分之一，程序通过JavaBean组件来执行所要求的更为复杂的运算。JavaBean组件不仅可以应用于JSP中，同样适用于其他的Java应用程序中。这种特性使得开发人员之间可以共享JavaBean组件，加快了应用程序的总体开发进程。

同样，JSP的标准标签和自定义标签与JavaBean组件一样可以一次生成重复使用。这些标签都是通过编写的程序代码来实现特定功能的，在使用它们时与通常在页面中用到的HTML标记用法相同。这样可以将一个复杂而且需要出现多次的操作简单化，大大提高了工作效率。





沿用了Java Servlet的所有功能

相对于Java Servlet来说，使用从Java Servlet发展而来的JSP技术开发Web应用更加简单易学，并且JSP同样提供了Java Servlet所有的特性。实际上服务器在执行JSP文件时先将其转换为Servlet代码，然后再对其进行编译，可以说JSP就是Servlet，创建一个JSP文件其实就是创建一个Servlet文件的简化操作。理所当然，Servlet中的所有特性在JSP中同样可以使用。





预编译

预编译是JSP的另一个重要的特性。JSP页面在被服务器执行前，都是已经被编译好的，并且通常只进行一次编译，即在JSP页面被第一次请求时进行编译，在后续的请求中如果JSP页面没有被修改过，服务器只需要直接调用这些已经被编译好的代码，这大大提高了访问速度。

