Web程序设计

▶▶第6章 动态网页技术概述

NCRE

第6章 动态网页技术概述

6.1静态网页与动态网页

6.1.1 静态网页

- 在网站设计中，纯粹HTML格式的网页通常被称为“静态网页”，早期的网站一般都是由静态网页制作的。
- 静态网页一般会以 .htm或者 .html扩展名存储在服务器的文件系统中。
- 在HTML格式的网页上，也可以出现各种动态的效果，如 .GIF格式的动画、FLASH、滚动字幕等，这些“动态效果”只是视觉上的，不是区分静态网页和动态网页的要素。

6.1静态网页与动态网页

6.1.1 静态网页

静态网页的特点主要有：

（1）静态网页的每个网页都有一个固定的URL，且网页URL以 .htm、.html、.shtml等常见形式为后缀；

（2）网页内容一经发布到网站服务器上，都是保存在网站服务器上的；

（3）网页内容不会发生变化

（4）静态网页一般没有数据库的支持

（5）静态网页不能和浏览器用户交互，只能单向发布信息，不适应动态和即时信息发布

（6）维护工作量较大。

6.1静态网页与动态网页

6.1.2 动态网页

- 动态网页的内容不是预先定制的静态HTML文档，而是在请求或使用过程中根据实际的数据内容和条件实时生成的页面。
- 动态网页的内容可随用户的输入和互动而有所不同，或者随着用户、时间、数据修正等而改变。
- 使用动态页面，用户可以提交信息给服务器，服务器可以将数据库中的数据返回给用户，例如股市行情、天气预报等动态检索结果。

6.1静态网页与动态网页

6.1.2 动态网页

客户机端动态网页采用客户机端脚本语言程序（Client-Side Scripting）在客户机端浏览器中动态地生成整个页面。通常，仅使用客户机端脚本程序实现一些轻量级或局部性的数据与外观处理，如数据检查、交互控制、动画演示等辅助性工作。

客户机端脚本主要采用：

- JavaScript
- VBScript
- ActionScript
- 应用小程序/插件
- Java Applet、ActiveX控件、Flash插件

6.1静态网页与动态网页

6.1.2 动态网页

服务器端动态网页由服务器端实时生成HTML文档并返回给浏览器。通常会通过专门的生成程序或者服务器端脚本语言生成动态页面，数据内容通常从后台数据库中取得，“拼装”成最终的HTML页面并返回给浏览器。

- 早期技术如CGI、Perl和Java Servlet等。
- 改进技术，如JSP、ASP、PHP等。允许在HTML页面中嵌入一些服务器端脚本语言程序，可以动态构筑页面内容。
- ASP.NET技术可将页面展示语义与数据内容的处理逻辑彻底分开，提供了面向对象和基于事件驱动的编程方式以完成网页页面的开发，其技术更先进，使用更方便。

6.1静态网页与动态网页

6.1.2 动态网页

客户机/服务器端混合式动态网页技术Ajax

无论是客户机端动态网页还是服务器端动态网页，一个共同特点是，一旦网页需要从服务器上请求哪怕是一点点数据，也一定要向服务器请求和重新装入整个页面，因此，访问效率不高，增加了网络负担。

为了解决网页局部更新的问题，出现了**Ajax**（**Asynchronous JavaScript and XML**）技术。**Ajax**是一个基于**JavaScript**并整合了**XHTML**、**XML**、**DOM**等技术实现的一个客户机端/服务器端混合式动态页面编程框架，它解决了网页的局部更新问题，大大降低了**Web**服务器的负担和网络通信开销，加快了服务器的响应速度。

6.1静态网页与动态网页

6.1.2 动态网页

动态网页的一般特点为：

（1）动态网页以数据库技术为基础，可以大大减少降低网站维护的工作量；

（2）采用动态网页技术的网站可以实现更多的动态访问功能，如用户注册、用户登录、在线调查、用户管理、订单管理等；

（3）动态网页只有当用户请求时服务器才从数据库中读取数据并动态产生一个完整的网页，并不占用独立的服务器空间。

6.1静态网页与动态网页

6.1.3 应用场景

- 从网页开发角度看，静态网页使用HTML语言，而动态网页使用的语言则包括HTML+ASP或HTML+PHP或HTML+JSP等。

- 静态网页相对更新起来比较麻烦，一般适用于更新较少的展示型网站。

- 动态网站也可以采用静动结合的原则，适合采用动态网页的地方用动态网页，如果有必要使用静态网页，则可以考虑用静态网页的方法来实现。

6.2 Java Servlet和JSP基本概念和原理

6.2.1 Servlet工作原理

- Java Servlet（简称Servlet）技术是在JSP技术产生之前的一种在Web服务端运行的Java技术，Servlet实质上是遵循一定规范的、运行于Web服务端、供服务端调用和执行的Java类，人们称Servlet为服务端小程序。

- Servlet遵循HTTP协议的请求/响应交互模型，从客户端接收请求进行处理，并向客户端返回响应。Servlet技术是Java动态Web技术的基础

6.2 Java Servlet和JSP基本概念和原理

6.2.1 Servlet工作原理

❖ **Servlet的运行过程**

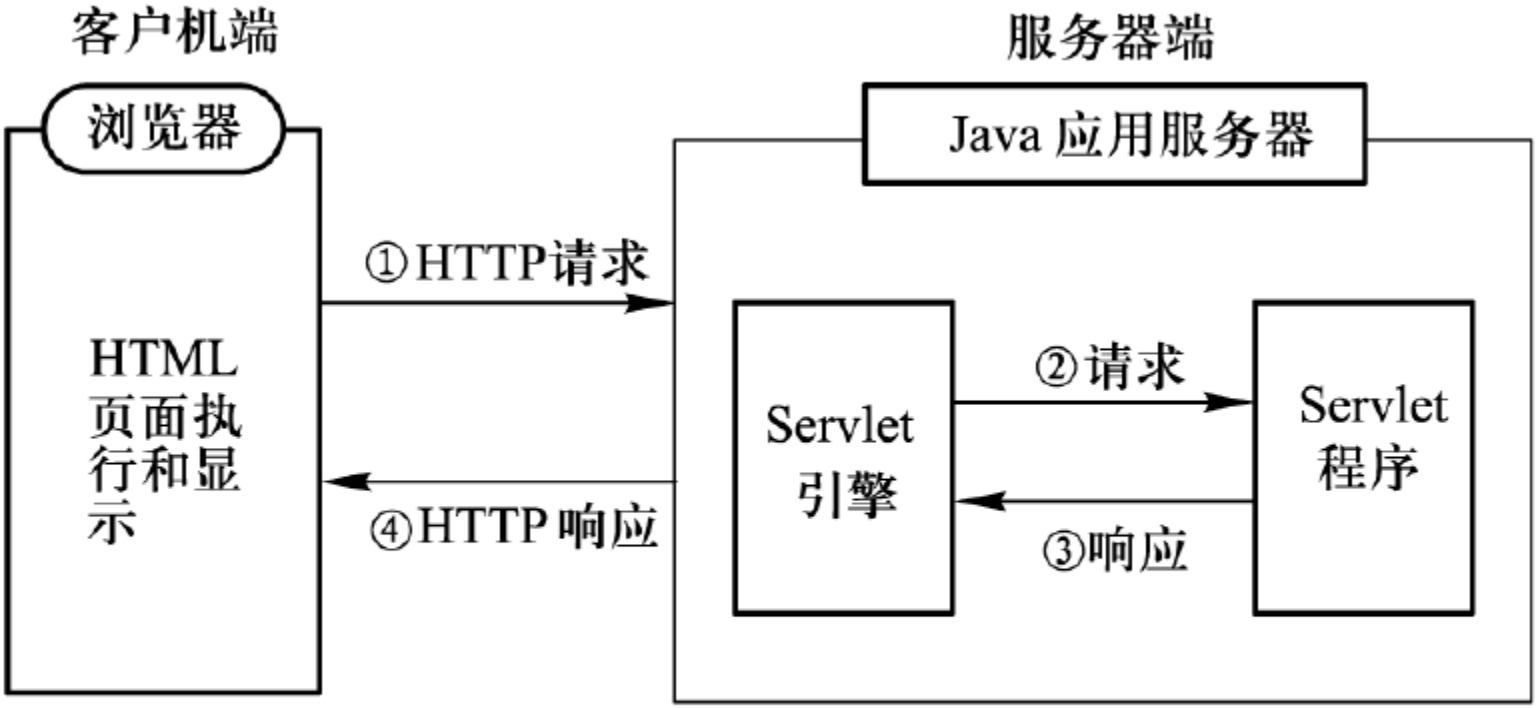


图 6-1　Servlet 的执行过程

6.2.2 JSP工作原理

- JSP的工作原理类似CGI程序，它替代了常规Web服务器后端的页面加载模块，由JSP引擎执行从JSP程序转换编译而成的Java Servlet程序来产生动态HTML页面。

- 一个JSP程序在第一次访问的时候等待响应的时间相对会比较长，这是由于JSP引擎需要先完成JSP程序的翻译和编译工作。第二次以后的执行速度就会快很多。

6.2 Java Servlet和JSP基本概念和原理

6.2.2 JSP工作原理

❖ JSP程序的执行过程

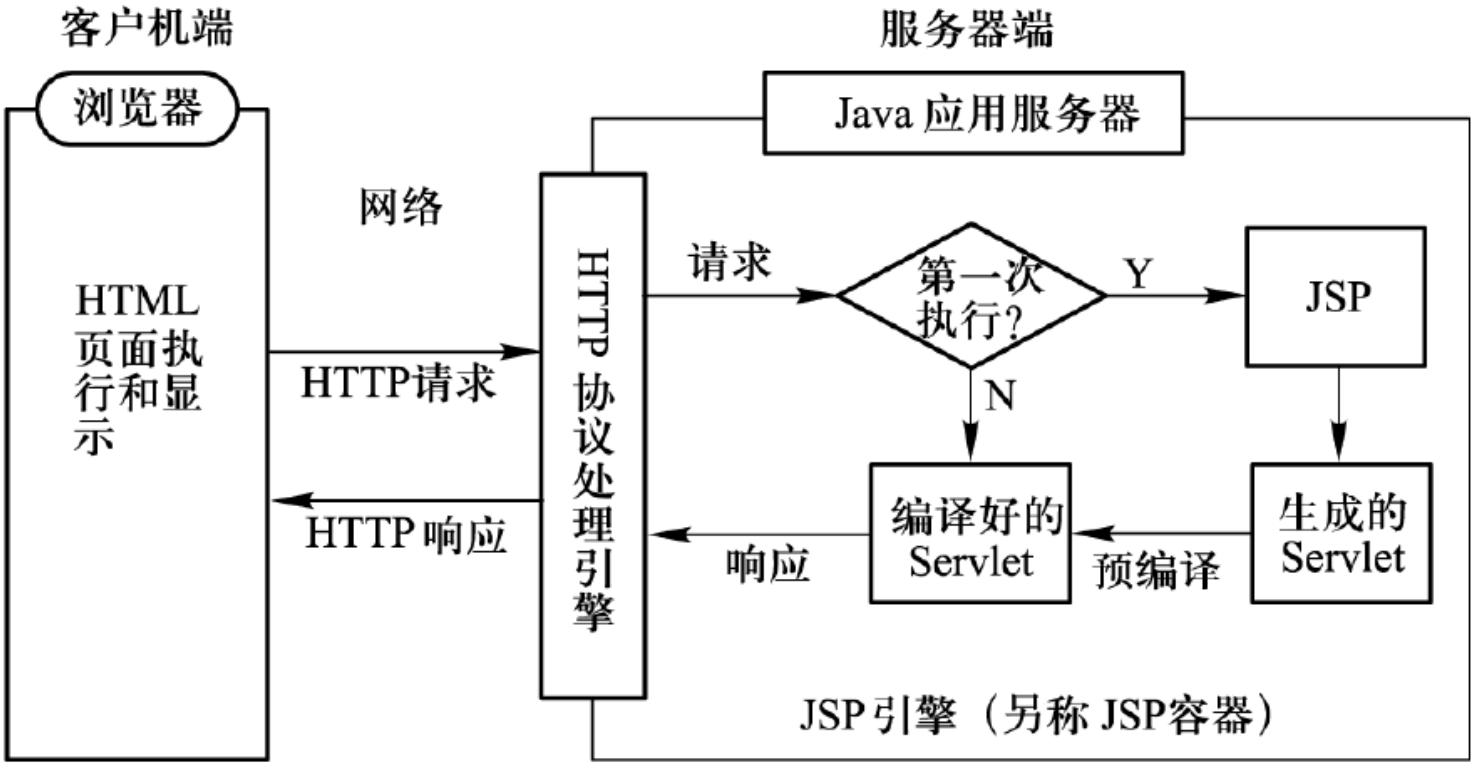


图6-2 JSP 程序的执行过程

6.2 Java Servlet和JSP基本概念和原理

6.2.3 JSP基本构成

完整JSP网页构成元素有三类：

- HTML标记：客户端浏览器看到的都是HTML标记语言，通过对HTML的解析，才能产生用户可见的页面。
- JSP标记，包括JSP指令标记和JSP动作标记；嵌入的部分以标记“<%”开始，以“%>”结束。
- Java脚本程序，是嵌入到JSP页面的可执行的Java程序，它们与JSP的其他内容一起被编译成为Servlet，在运行过程中被执行并输出结果作为HTML的一部分返回给客户端。

6.2 Java Servlet和JSP基本概念和原理

6.2.3 JSP基本构成

示例

<%@ page contentType="text/html;charset=utf-8" %>	JSP 指令
<html>	
<head>	
<title>Hello world demo</title>	HTML 标记
</head>	
<body>	
<jsp:include page="other.jsp" flush="true" />	JSP 动作
<!-- 以下是 Java 脚本程序 -->	HTML 注释
<%	
// 这是隐藏注释	Java 脚本程序（其中包含隐藏注释）
String s = "Hello, world!";	
out.println(s);	
%>	
<%-- 这也是隐藏注释 --%>	隐藏注释
 	HTML 标记
<%=s%>	Java 脚本程序—表达式
</body>	
</html>	HTML 标记

6.2.4 Servlet与JSP的关系和区别

- Servlet和JSP都是运行在服务端的程序，JSP技术是在Servlet技术的基础上发展起来的，它们之间有很多联系，也有很大的区别。

- Servlet和JSP的编程方式不同。Servlet遵循的是Java语言的编程标准，而JSP更多的是遵循脚本语言的编程标准。Servlet编写的程序结构性更好，更加专业，而JSP是为了简化Servlet编程而发展起来的技术，JSP程序在编写方面比Servlet要容易。

6.2.4 Servlet与JSP的关系和区别

- JSP编程由于采用的是脚本语言编程标准，是由JSP引擎在第一次运行时翻译成Servlet后编译执行。Servlet则是需要在程序开发完成后进行编译才能部署安装。

- 在实际的运用中，通常会采用Servlet和JSP混合使用的MVC模式,由JSP充当View角色，负责显示动态内容；由Servlet充当Controller角色，负责对客户端的请求进行处理、响应和调用JavaBean；由JavaBean充当Model角色，负责提供可复用组件和数据的存储访问等。

6.2 Java Servlet和JSP基本概念和原理

6.2.5 Servlet编程简单示例

```
public class HelloWorldServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("Hello,world!");
        out.println("</body>");
        out.println("</html>");    } }
```

6.2.5 Servlet编程简单示例

配置文件：

```
<servlet>
  <servlet-name>HelloWorldServlet</servlet-name>
  <servlet-class>test.HelloWorldServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>HelloWorldServlet</servlet-name>
  <url-pattern>/helloworld</url-pattern>
</servlet-mapping>
```

6.2.6 JSP编程简单示例

```
<html>
<head>
<title>Hello world demo</title>
</head>
<body>
  <%    String s = “Hello, world!”;
    out.println(s);  %>
  <br />
  <%=s%>
</body>
</html>
```

6.3 ASP.NET基本概念和原理

6.3.1微软.NET框架基础

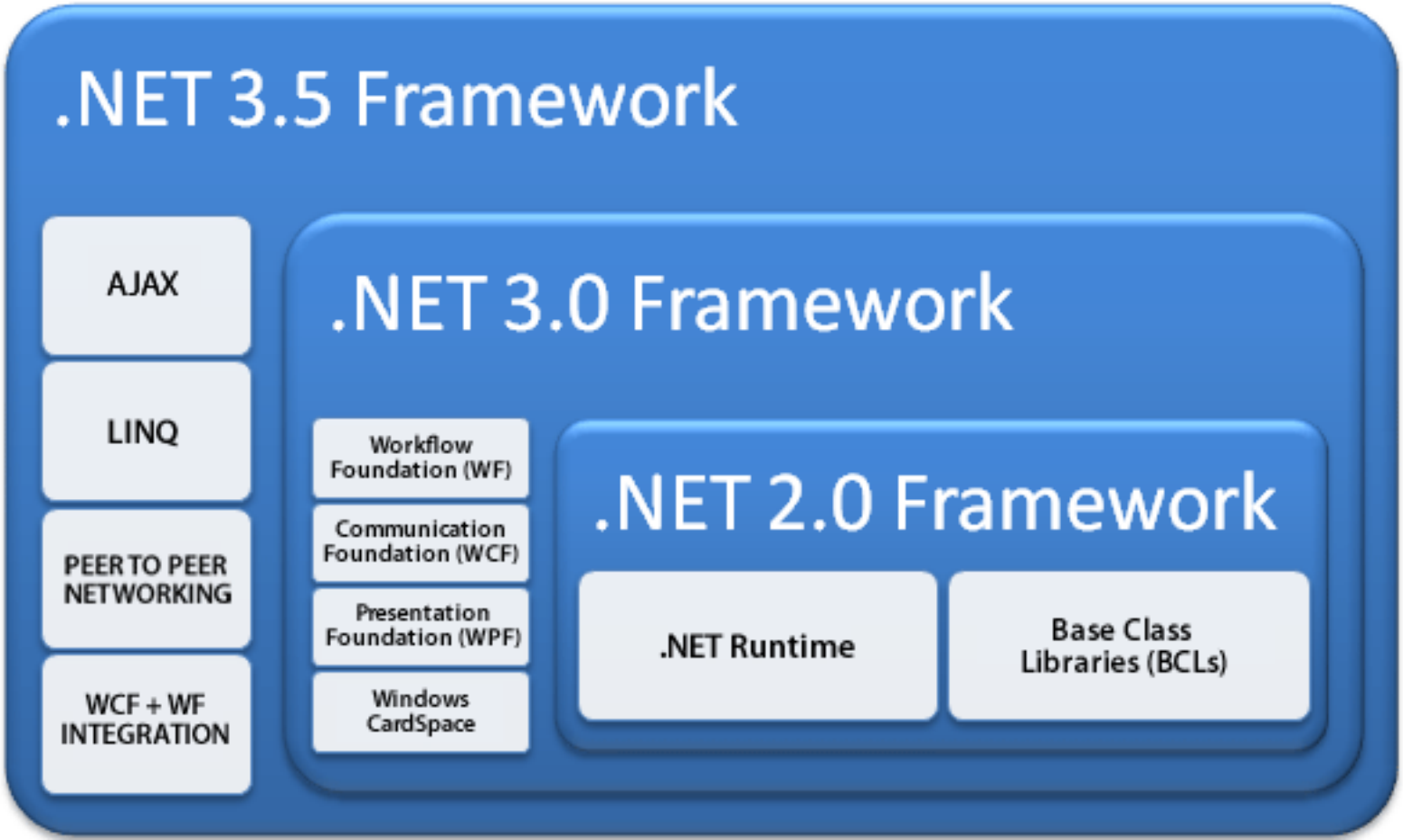
❖ .NET框架（.NET Framework）是由微软开发的一个致力于敏捷软件开发、快速应用开发、平台无关性和网络透明化的软件开发平台。

❖ 框架组成：公共语言运行时（CLR：Common Language Runtime）、服务框架（Services Framework）和上层的两类应用模板——传统的Windows应用程序模板（Win Forms）和基于ASP.NET的面向Web的网络应用程序模板（Web Forms和Web Services）。

6.3 ASP.NET基本概念和原理

6.3.1微软.NET框架基础

- .NET框架



6.3 ASP.NET基本概念和原理

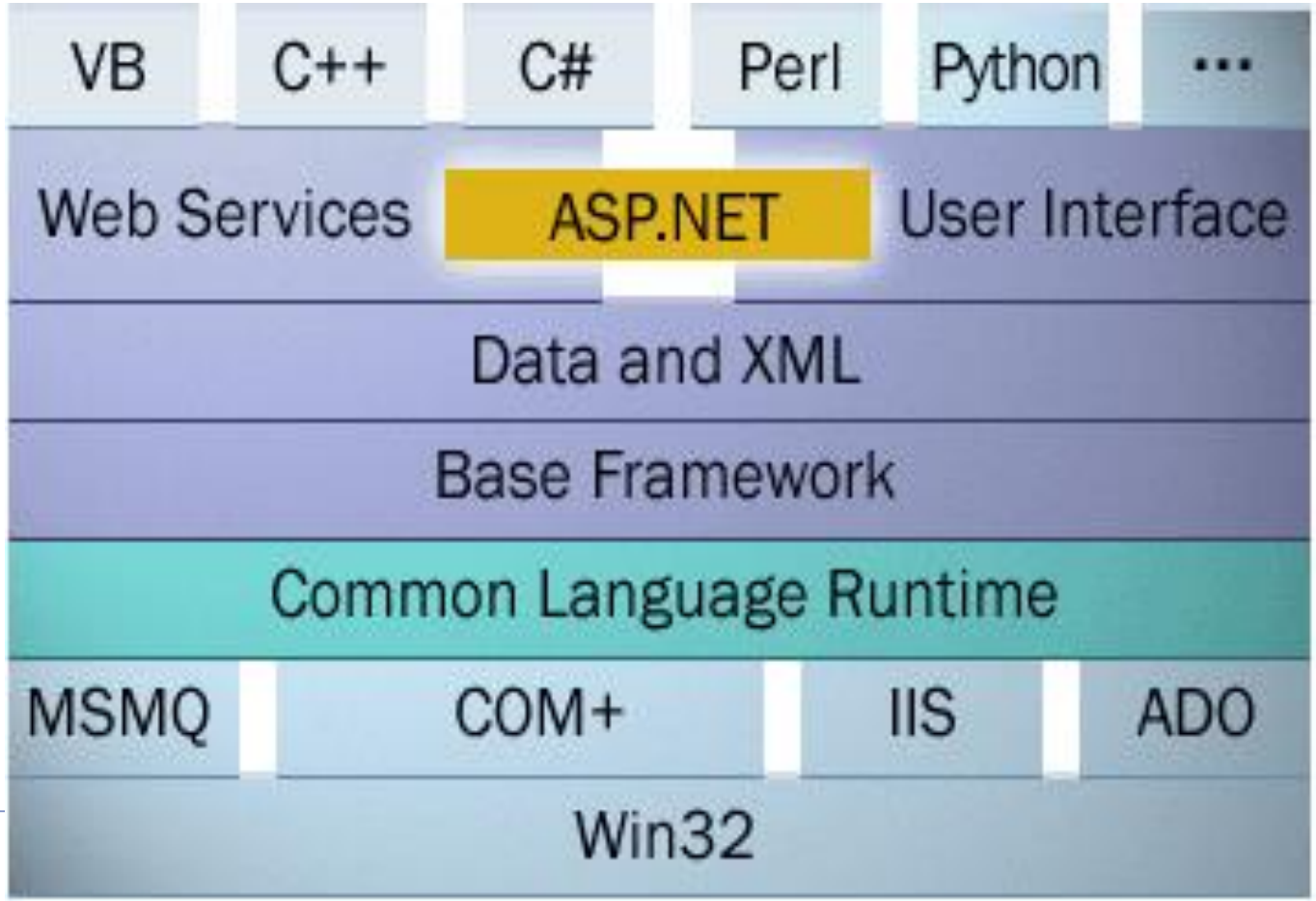
6.3.2 ASP.NET简介

- ASP.NET 是一个统一的 Web 开发模型，它包括开发者使用尽可能少的代码生成企业级 Web 应用程序所必需的各种服务。
- ASP.NET 作为 .NET 框架的一部分提供。开发者可以访问 .NET框架中的类。
- 开发者可以使用与公共语言运行库 (CLR) 兼容的语言来编写代码，包括 VB、C#、JScript .NET 和 J#。

6.3 ASP.NET基本概念和原理

6.3.2 ASP.NET简介

- ASP.NET在整个开发架构中的位置



6.3 ASP.NET基本概念和原理

6.3.3 ASP.NET基本编程模型

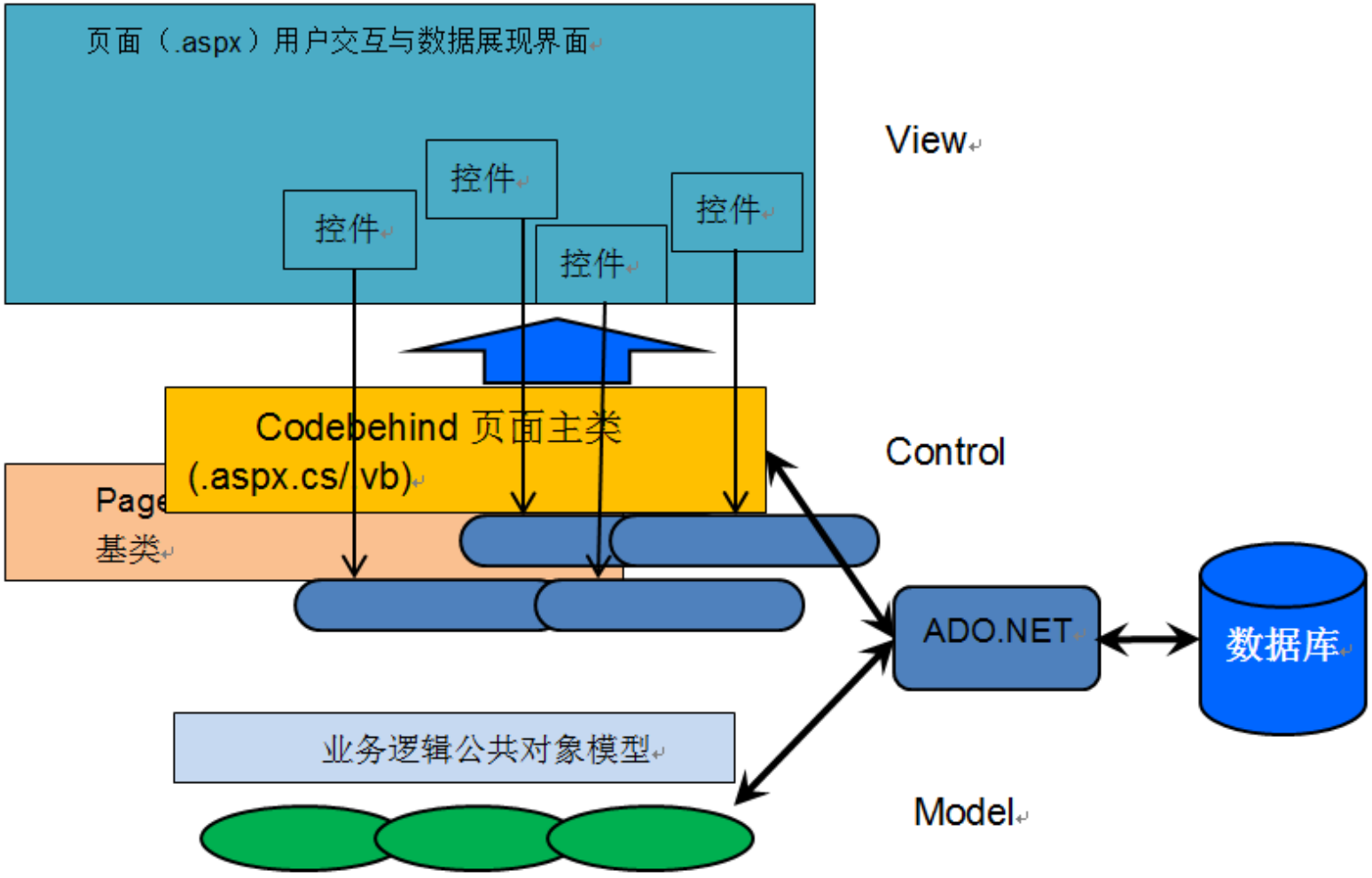


图 6-5 ASP.NET 基本编程模型

6.3 ASP.NET基本概念和原理

6.3.3 ASP.NET基本编程模型

- 在ASP.NET编程中，开发者在Web Form上拖放控件，完成想要的页面布局，用于和用户进行交互；

- Page类是所有Web Form类的父类，Page类的编程模型非常复杂，这是由于身上繁重的任务决定的：Page类必须应用用户个性化信息，应用主题，初始化和组织内部的控件，为控件加载和保存视图状态，为控件加载回传数据，引发控件回传，组织页面呈现内容并最终通过HtmlTextWriter输出结果。

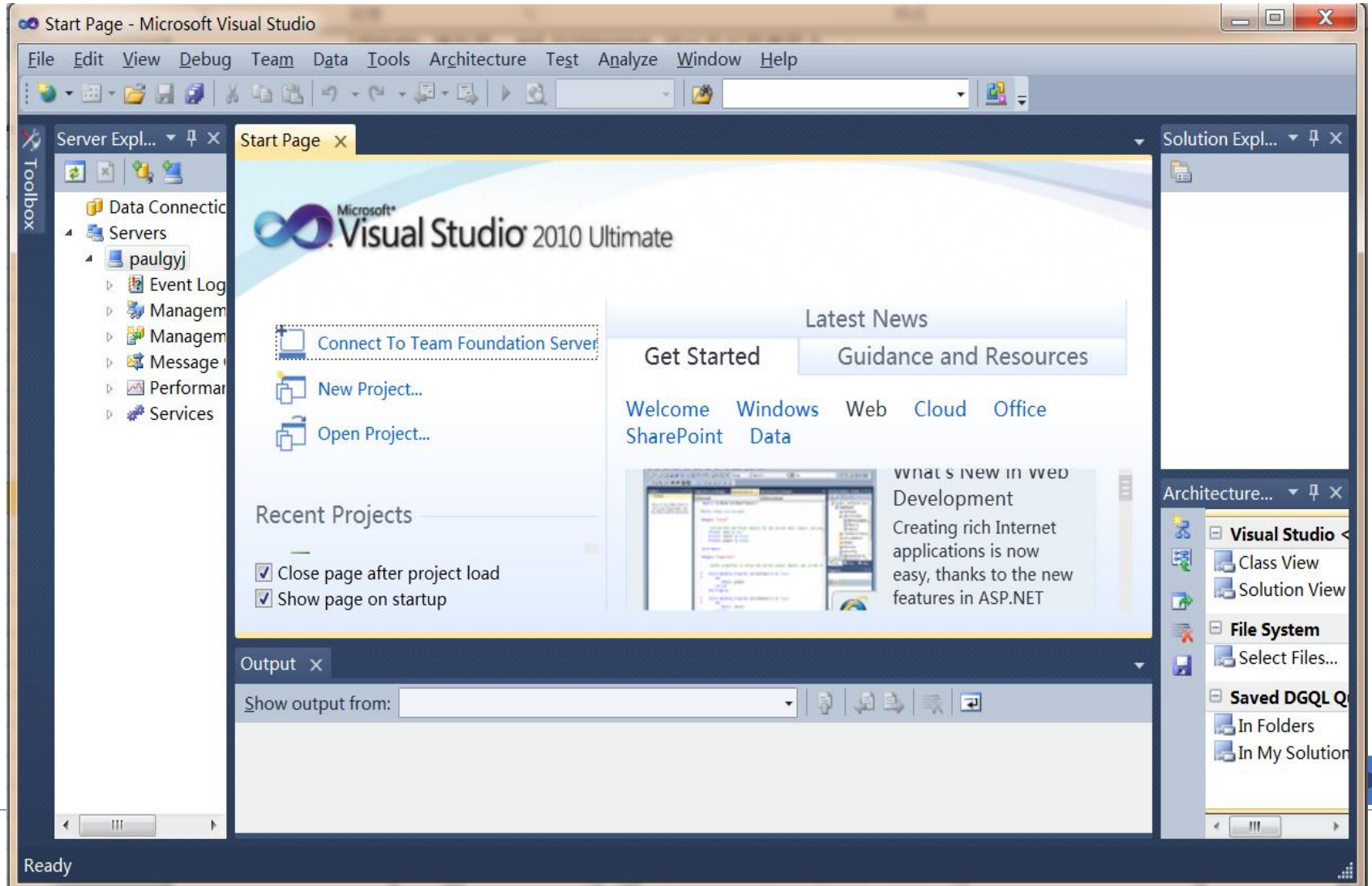
6.3 ASP.NET基本概念和原理

6.3.4 ASP.NET开发环境

❖ Visual Studio.NET 是一套完整的开发工具集，用于生成 ASP.NET Web 应用程序、XML Web Services、桌面应用程序和移动应用程序等各类应用程序。Visual Basic、Visual C++、Visual C# 和 Visual J# 全都使用相同的集成开发环境，利用此 IDE 可以共享工具且有助于创建混合语言解决方案。另外，这些语言利用了 .NET Framework 的功能，通过此框架可使用简化 Web 应用程序和 XML Web Services 开发的关键技术。

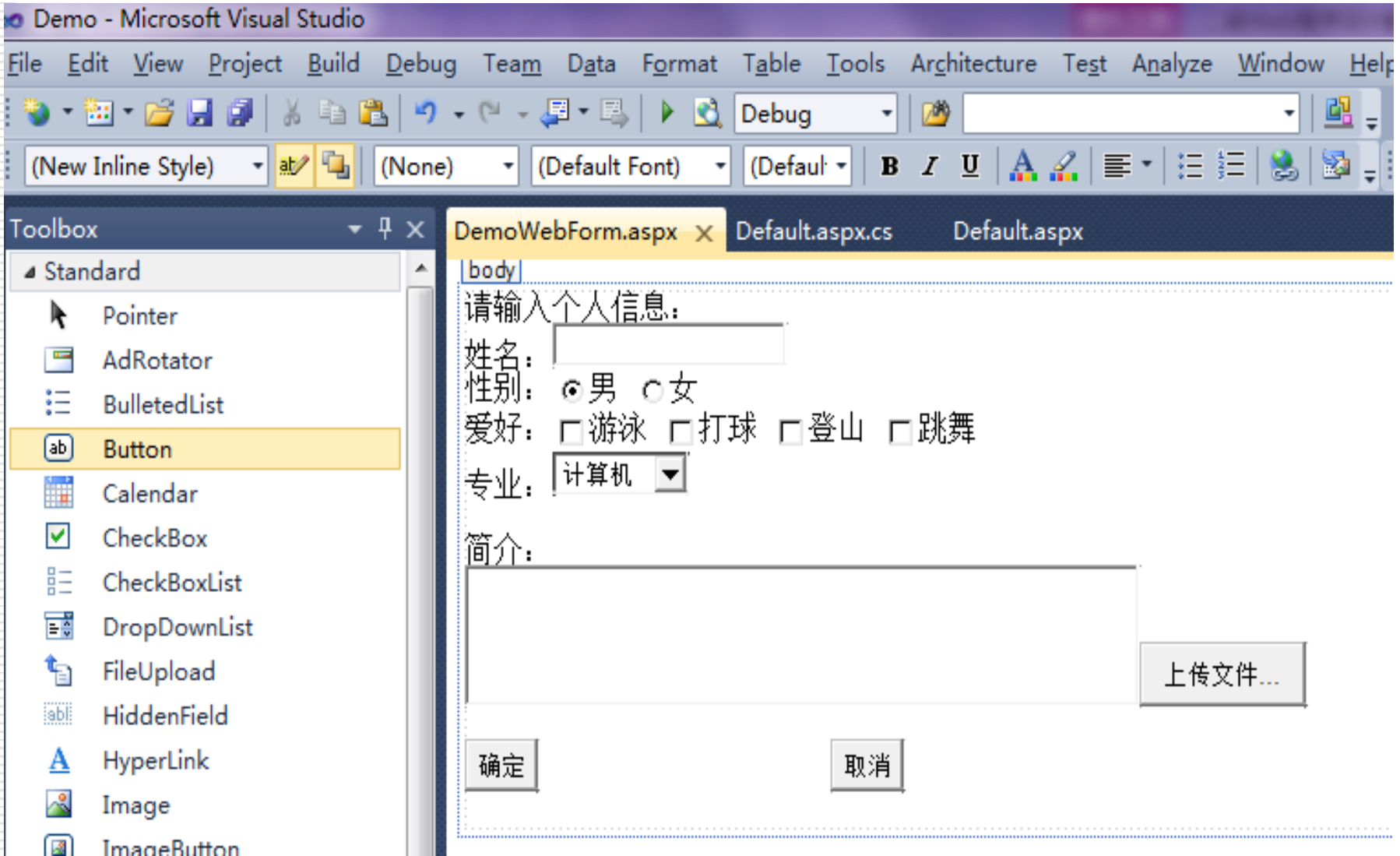
6.3 ASP.NET基本概念和原理

6.3.4 ASP.NET开发环境



6.3 ASP.NET基本概念和原理

6.3.5 ASP.NET网页开发简单示例



6.3 ASP.NET基本概念和原理

6.3.5 ASP.NET网页开发简单示例

代码分离：

```
namespace Demo
{   public partial class DemoWebForm : System.Web.UI.Page
    {     protected void Page_Load(object sender, EventArgs e)
        {     }//这里往往对控件数据进行初始化
        protected void btnOk_Click(object sender, EventArgs e)
        {     }//用户点击“确定”按钮后的动作
        protected void btnCancel_Click(object sender, EventArgs e)
        {     }//用户点击“取消”按钮后的动作
        protected void btnUpload_Click(object sender, EventArgs e)
        {     }//用户点击“文件上传”按钮后的动作   }}
```

6.4 PHP基本概念和原理

6.4.1 PHP简介

- PHP是由Rasmus Lerdorf 于1994年创建的，最初只是一个简单的用Perl语言编写的程序，用来统计网站的访问者情况。
- 在1995年以Personal Home Page Tools (PHP Tools)的名称开始对外发表第一个版本。
- 后来缩写更改为Hypertext Preprocessor（超文本预处理语言）。

6.4 PHP基本概念和原理

6.4.1 PHP简介

一个典型的早期PHP代码例子如下：

```
<HTML>
<HEAD>
<TITLE>My Page</TITLE>
</HEAD>
<BODY>
Hello, <?PHP echo ”Michael”;?>, Welcome to my site!
</BODY>
</HTML>
```

6.4 PHP基本概念和原理

6.4.2 PHP的工作原理

一个完整的PHP系统由以下几个部分构成。

- 服务器：搭建PHP运行环境时所选择的服务器。PHP支持多种服务器软件，包括Apache、IIS等。
- PHP引擎：实现对PHP文件的解析和编译。
- 数据库系统：实现系统中数据的存储。PHP支持多种数据库系统，包括MySQL、SQL Server、Oracle及DB2等。
- 浏览器：浏览网页。由于PHP在发送到浏览器的时候已经被解析器编译成HTML代码，所以PHP对浏览器没有任何限制。

6.4 PHP基本概念和原理

6.4.2 PHP的工作原理

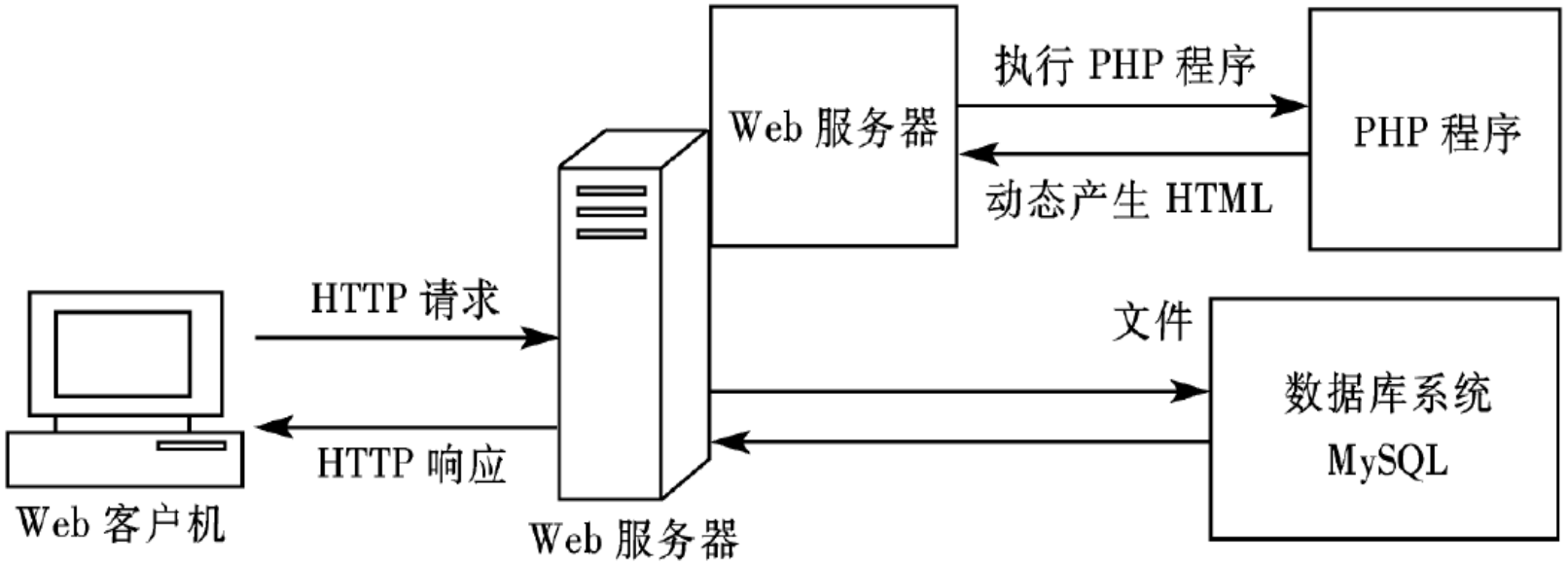


图 6-12　PHP 的工作原理

6.4 PHP基本概念和原理

6.4.2 PHP的工作原理

Smarty模板引擎工作原理图

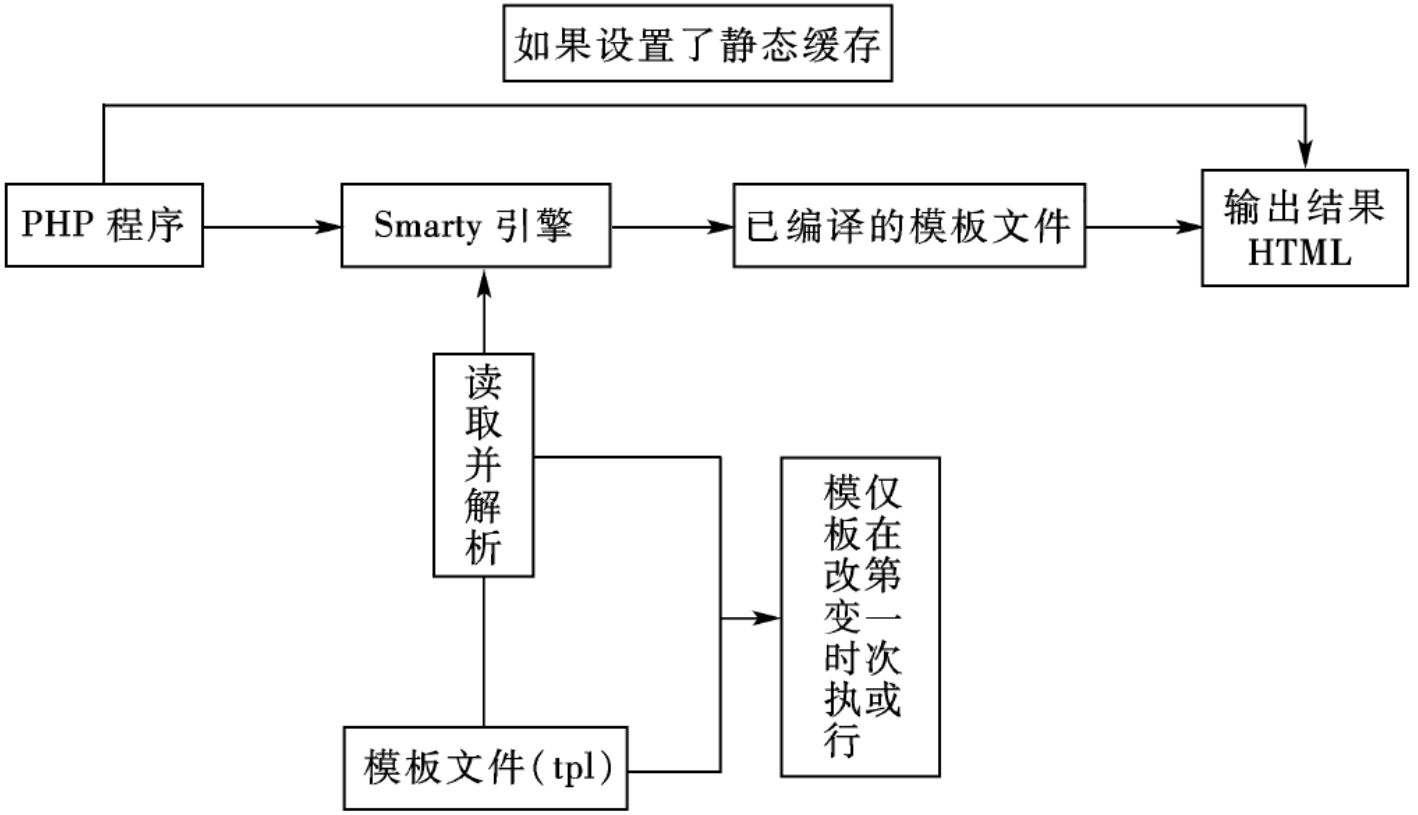


图 6-13　Smarty 工作原理图

6.4 PHP基本概念和原理

6.4.3 PHP网页开发简单示例

```
<HTML>
<HEAD> <TITLE> <? echo "Hello World!"; ?> </TITLE>
</HEAD>
<BODY>
<H1> First PHP page </H1>
<HR> <? /*
块注释：打印信息 */
//类C语言行注释
$today=date("Y-m-d G:i:s");
echo "<CENTER> 今天是：$today.</CENTER>";
?># Unix 类型的行注释 ?>
</BODY>
</HTML>
```

6.5 Ajax基本概念和原理

6.5.1 Ajax技术背景

- Ajax的全称是Asynchronous JavaScript and XML，表示异步JavaScript+XML的意思，它有别于传统web开发中采用的同步方式。

- 除了JavaScript之外，另外一个组成Ajax的关键技术帧及隐藏帧技术的出现，使得浏览器对服务器的独立请求与显示处理变得更加灵活，更加有助于提高用户体验。

- 动态HTML技术中用户可以用JavaScript 和CSS来更新页面的任何部分。

6.5 Ajax基本概念和原理

6.5.1 Ajax技术背景

- 2001年，微软公司以ActiveX对象的形式引入了XMLHttp工具，目的是向开发人员提供一个实现浏览器/服务器交互的更好的工具。

- 随后几乎所有的主流浏览器都通过将XMLHttp对象的主要方法和属性复制到浏览器的XMLHttpRequest对象中来支持这种技术，从而导致Ajax风格的页面在整个Web开发领域中迅速流行起来。

6.5 Ajax基本概念和原理

6.5.2 Ajax技术的构成

Ajax并非一种新的技术，而是几种原有技术的结合体。它实际上确是由下列技术组合而成。它们包括：

- HTML/XHTML，主要的内容表示语言；
- CSS，为XHTML提供文本格式定义；
- DOM，对已载入的页面进行动态更新；
- XML，数据交换格式；
- XSLT，将XML转换为XHTML（用CSS修饰其样式）；
- XMLHttp，用XMLHttpRequest来和服务器进行异步通信，是主要的通信代理；
- JavaScript，用来编写Ajax引擎的脚本语言。

6.5 Ajax基本概念和原理

6.5.2 Ajax技术的构成

- 除了客户端技术之外，还需要一个很重要的组件就是必要的服务器端处理逻辑。
- 开发人员所关注的技术主要与客户端的**Ajax**引擎直接相关，但如果没有一个稳定的、响应及时的服务器来向引擎发送内容，也就不会有**Ajax**的存在。
- 不管用户将服务器端组件编写为**PHP**页面、**Java servlet**还是**.NET**组件，只需要保证向**Ajax**引擎发送的数据格式是正确的即可

6.5 Ajax基本概念和原理

6.5.3 Ajax的工作原理

- Ajax模型提供了一个中间层（称之为Ajax引擎）来处理浏览器和服务器之间的通信。
- Ajax引擎实际上只是一个JavaScript对象或函数，只有当信息必须从服务器上获得的时候才调用它。
- 当需要调度和执行浏览器请求时，向Ajax引擎发出一个函数调用。这些请求都可以异步完成的，这就意味着不必等收到响应之后就可以继续执行后续的代码，从而可以提升用户体验。

6.5 Ajax基本概念和原理

6.5.3 Ajax的工作原理

❖ 传统Web应用程序模型和Aiax模型之间的区别

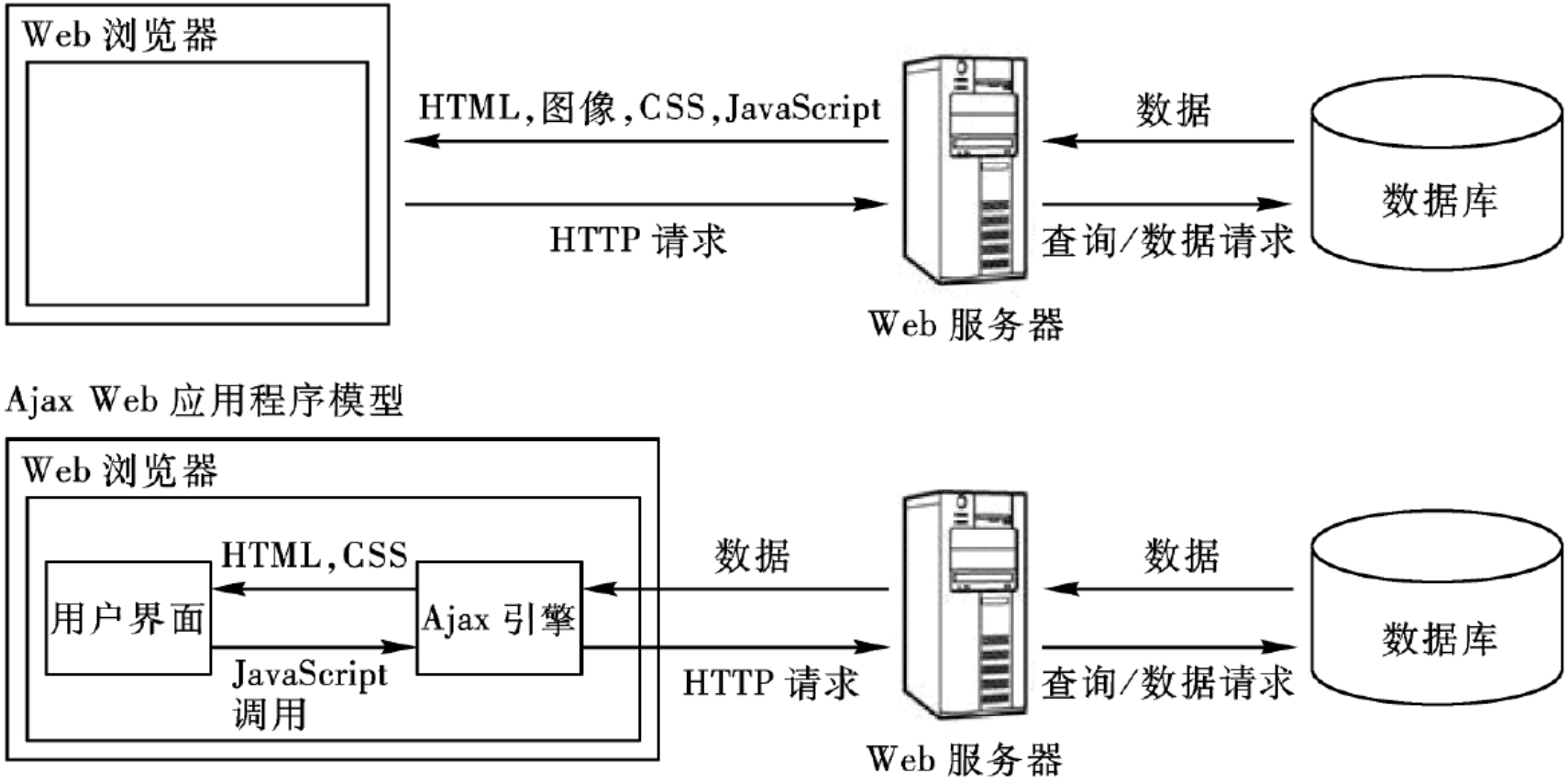


图 6-14　传统 Web 模式(上)与 Ajax 模型(下)

6.5 Ajax基本概念和原理

6.5.3 Ajax的工作原理

- XMLHttpRequest对象是一项关键功能，浏览器通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用JavaScript来操作DOM而更新页面。
- XMLHttpRequest 对象提供了对 HTTP 协议的完全的访问。XMLHttpRequest 可以同步或异步返回 Web 服务器的响应，并且能以文本或者一个 DOM 文档形式返回内容。
- XMLHttpRequest，可以接收任何形式的文本文档

6.5 Ajax基本概念和原理

6.5.4 Ajax开发简单示例

在页面中添加一个文本框标签用于输入用户名，添加一个按钮用于提交用户数据。验证的结果被直接打印在页面上。

操作步骤如下：

1） 打开Visual Studio 2010建立一个网站

2） 在VS2010工具箱的HTML栏中添加一个文本控件和一个按钮控件。

3） 在<Head></Head>中加入<script type ="text/javascript"></script>标签，以便进行Ajax引擎的编写，再定义一个XMLHttpRequest对象，但是并不进行初始化操作。

6.5 Ajax基本概念和原理

6.5.4 Ajax开发简单示例

4） 添加OnMessageBack()函数的内容。代码如下：

```
function OnMessageBack()
    {   //判断请求状态及HTTP状态是否都能满足条件
        if (xmlhttp.readystate==4 &&
xmlhttp.status==200)
        {//将返回的文本打印到页面上
            document .write xmlhttp .responsetext);   }}
```

5） 为Button1添加事件代码，将其标签修改为

```
<input id="Button1" type="button" value="button"
onclick="Validation()" />
```

6）页面中的Ajax引擎已经编写完毕。

本章小结

本章主要介绍目前主流的动态网页技术。

动态网页技术是在传统的静态网页技术的基础上发展而来的，尤其适合现代人们对网络信息的获取速度和用户体验的要求。

本章主要介绍了目前的三种动态网页技术：**JSP**、**ASP.NET**和**PHP**。分别介绍了三种技术的产生背景和工作原理，并给出了简单示例，便于读者对这些技术有个形象的认识。

本章最后介绍了目前非常流行的异步网页无刷新技术**Ajax**。**Ajax**不是一种全新的技术，而是基于原有的**Web**技术开发出来的一种**Web**交互的方法。