第一图书网, tushu007.com

<<Struts 2 技术详解>>

图书基本信息

书名：<<Struts 2 技术详解>>

13位ISBN编号：9787121062216

10位ISBN编号：7121062216

出版时间：2008-6

出版时间：电子工业出版社

作者：闫术卓　等编著

页数：630

字数：965000

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

第一图书网, tushu007.com

<<Struts 2 技术详解>>

内容概要

Struts 2框架是Struts 1.X的替代版本，Struts 2框架整合了Struts 1.X框架和WebWork框架的优点。相对Struts 1.X，Struts 2已经有了非常大的改变，去掉了ActionForm，降低了框架组件之间的耦合性，Struts 2的Action只是普通的Java类（POJO），给模块测试工作带来了极大的方便。Struts 2提供了强大的整合能力，支持多种返回结果类型，改进了Struts 1.X的标签库，引入OGNL表达式和值栈的概念，给开发者带来了更好的体验。

本书结合Struts 2框架的技术特点，从最基础的框架处理机制讲起。介绍了Struts 2框架的核心组件和核心处理机制，并介绍了拦截器、国际化、输入校验、类型转换等Struts 2关键技术，同时介绍了如何在Struts 2框架中整合其他开源技术。

本书非常适合有过Struts 1.X和WebWork开发经验的读者，以及Java Web开发的初学者，对于Web高级开发者，也是一本非常好的参考书。

第一图书网, tushu007.com

<<Struts 2 技术详解>>

书籍目录

第一篇 Struts 2零基础详解 第1章 Struts 2概述 1.1 MVC介绍 1.1.1 Web技术发展 1.1.2 MVC设计模式 1.1.3 MVC的处理过程 1.1.4 MVC的优点 1.1.5 MVC的适用范围 1.1.6 Model 1和Model 2体系介绍 1.2 Struts 1简介 1.2.1 Struts 1框架介绍 1.2.2 web.xml配置文件 1.2.3 struts-config.xml配置文件 1.2.4 Action和ActionForm 1.2.5 Struts 1处理过程 1.2.6 Struts 1的优点 1.2.7 Struts 1的缺点 1.3 WebWork简介 1.3.1 WebWork框架 1.3.2 WebWork的特性 1.4 Struts 2概述 1.4.1 Struts 2框架 1.4.2 Struts 2配置件 1.4.3 Struts 2控制器 1.4.4 Struts 2标签库 1.4.5 Struts 2与Struts 1比较 1.4.6 Struts 2与WebWork比较 1.5 为什么使用Struts 2 1.6 本章小结 第2章 Struts 2的HelloWorld 2.1 搭建Struts开发环境 2.1.1 安装JDK 2.1.2 安装Eclipse 2.1.3 安装Tomcat 2.1.4 安装Struts 2 2.1.5 Eclipse安装Struts 2 2.2 一个简单的HelloWorld 2.2.1 配置web.xml文件 2.2.2 配置struts.xml文件 2.2.3 Action业务控制器 2.2.4 视图资源 2.2.5 运行HelloWorld 2.2.6 HelloWorld小结 2.3 Struts 2特性演示 2.3.1 基本实现 2.3.2 标签库 2.3.3 改进Action 2.3.4 国际化 2.3.5 数据校验：使用validate()方法校验 2.3.6 数据校验：使用框架校验 2.4 本章小结 第3章 Struts 2核心剖析 3.1 Struts 2工流程 3.1.1 核心控制器FilterDispatcher 3.1.2 业务控制器Action 3.1.3 业务模型组件 3.1.4 视图组件 3.2 Struts 2配置文件 3.2.1 配置web.xml文件 3.2.2 配置struts.xml文件 3.2.3 常量配置 3.2.4 包配置 3.2.5 命名空间配置 3.2.6 包含配置 3.2.7 Bean配置 3.2.8 拦截器配置 3.2.9 配置struts.properties文件 3.2.10 配置通配符 3.2.11 Struts 2的零配置 3.3 Struts 2的Action 3.3.1 Action实现类 3.3.2 实例验证：Action属性和用户参数之间的关系 3.3.3 Action访问ActionContext 3.3.4 值栈（ValueStack） 3.3.5 Action直接访问Servlet API 3.3.6 Action的配置 3.3.7 Action的动态方法调用 3.3.8 通配符配置 3.4 处理结果 3.4.1 处理结果流程 3.4.2 配置result 3.4.3 result的类型 3.4.4 action-chain类型示例 3.4.5 Freemarker类型示例 3.4.6 redirect类型示例 3.4.7 redirect-action类型示例 3.4.8 Stream类型示例 3.4.9 使用通配符动态配置result 3.4.10 使用OGNL动态配置result 3.5. 模型驱动 3.5.1 模型驱动的意义 3.5.2 模型驱动示例 3.6. 异常处理 3.6.1 Java的异常处理 3.6.2 Struts 2框架的异常处理 3.6.3 异常的配置 3.7 如何提高Struts 2性能 3.8 支持Struts 2框架的应用服务器 3.9 本章小结 第二篇 Struts 2框架技术 第4章 国际化 4.1 软件的国际化 4.2 Java的国际化支持 4.2.1 使用资源文件 4.2.2 使用资源类文件 4.2.3 MessageFormat类 4.3 Struts 2的国际化支持 4.3.1 配置资源文件 4.3.2 Struts 2国际化应用 4.3.3 使用占位符 4.3.4 范围资源文件 4.3.5 Action范围资源文件 4.3.6 临时资源文件 4.3.7 加载资源文件的顺序 4.4 动态访问国际化资源文件 4.4.1 动态访问国际化资源文件原理 4.4.2 建立资源文件和配置文件 4.4.3 建立Action和JSP 4.4.4 动态访问资源文件示例 4.5 Eclipse编写资源文件的插件 4.6 本章小结 第5章 Struts 2进阶——拦截器 5.1 拦截器介绍 5.1.1 AOP介绍 5.1.2 拦截的实现原理 5.1.3 拦截的意义 5.2 Struts 2拦截器 5.2.1 Struts 2拦截器原理 5.2.2 HelloWorld拦截器 5.2.3 定义拦截器 5.2.4 使用拦截器 5.2.5 默认拦截器 5.3 自定义拦截器 5.3.1 自定义拦截器实现类 5.3.2 使用自定义拦截器 5.4 拦截器深度剖析 5.4.1 拦截器的方法过滤 5.4.2 拦截器的执行顺序 5.4.3 拦截结果监听器 5.4.4 设置拦截器栈中拦截器参数 5.4.5 Struts 2框架的系统拦截器 5.5 拦截器应用示例 5.5.1 权限拦截器 5.5.2 配置拦截器 5.5.3 业务控制器Action 5.5.4 JSP视图 5.5.5 运行示例 5.6 本章小结 第6章 Struts 2的类型转换 6.1 类型转换 6.2 编写一个类型转换器 6.2.1 类型转换需求 6.2.2 编写自定义类型转换器 6.2.3 视图资源文件 6.2.4 运行示例 6.3 自定义类型转换器 6.3.1 基于OGNL的类型转换器 6.3.2 基于Struts 2的类型转换器 6.3.3 注册自定义类型转换器 6.3.4 数组属性类型转换器 6.3.5 集合属性类型转换器 6.4 使用Struts 2的类型转换 6.4.1 Struts 2系统内建的类型转换器 6.4.2 项

第一图书网, tushu007.com

目应用中常见的类型转换 6.4.3 使用OGNL表达式 6.4.4 使用集合类型属性 6.4.5 使用Set类型属性 6.5 类型转换中的异常处理 6.5.1 类型转换异常拦截器 6.5.2 一个简单的类型转换异常处理 6.5.3 改进类型转换异常显示信息 6.5.4 处理复合类型转换异常 6.6 本章小结 第7章 输入校验 7.1 输入校验介绍 7.1.1 输入校验的原因 7.1.2 使用JavaScript完成客户端校验 7.2 使用validate方法进行输入校验 7.2.1 validate()方法输入校验 7.2.2 validateXxx()方法输入校验 7.2.3 输入校验流程 7.3 基于框架的输入校验 7.3.1 使用字段校验的输入校验 7.3.2 使用客户端的输入校验 7.3.3 使用非字段校验的输入校验 7.3.4 输入校验的国际化信息 7.3.5 校验的搜索顺序 7.4 AJAX输入校验 7.4.1 配置AJAX环境 7.4.2 建立业务控制器 7.4.3 建立校验规则文件 7.4.4 建立JSP视图 7.4.5 运行AJAX校验示例 7.5 复合类型属性的输入校验 7.5.1 复合属性的校验 7.5.2 集合属性的校验 7.6 Struts 2框架的校验器 7.6.1 内建校验器 7.6.2 类型转换校验器 7.6.3 日期校验器 7.6.4 浮点数值校验器 7.6.5 邮件地址校验器 7.6.6 表式校验器 7.6.7 字段表达式校验器 7.6.8 整数校验器 7.6.9 正则表达式校验器 7.6.10 必填校验器 7.6.11 必填字符串校验器 7.6.12 字符串长度校验器 7.6.13 网址校验器 7.6.14 visitor校验器 7.7 本章小结 第8章 OGNL 8.1 OGNL基础知识 8 OGNL的优势 8.1.2 OGNL语法 8.1.3 一个使用OGNL的示例 8.2 Struts 2的OGNL 8.2.1 Struts 2的OGNL表达式 8.2.2 OGNL的集合操作 8.2.3 Lambda表达式 8.3 Struts 2中使用OGNL 8.3.1 业务控制器 8.3.2 JSP视图 8.3.3 运行示例 8.3.4 OGNL中的#、%和$符号 8.4 本章小结 第9章 Struts 2标签库 9.1 Struts 2标签库概述 9.1.1 标签简介 9.1.2 Struts 2标签库组成 9.1.3 Struts 2标签库的使用 9.1.4 Struts 2同Struts 1标签库的比较 9.2 控制标签 9.2.1 if/elseif/else标签 9.2.2 iterator标签 9.2.3 append标签 9.2.4 generator标签 9.2.5 merge标签 9.2.6 subset标签 9.2.7 sort标签 9.3 数据标签 9.3.1 action标签 9.3.2 bean标签 9.3.3 date标签 9.3.4 debug标签 include标签 9.3.6 param标签 9.3.7 push标签 9.3.8 set标签 9.3.9 url标签 9.3.10 property标签 9.4 主题与模板 9.4.1 主题 9.4.2 模板 9.5 表单标签 9.5.1 表单标签通用属性 9.5.2 checkbox标签 9.5.3 checkboxlist标签 9.5.4 combobox标签 9.5.5 doubleselect标签 9.5.6 datetimepicker标签 9.5.7 head标签 9.5.8 file标签 9.5.9 hidden标签 9.5.10 select标签 9.5.11 optiontransferselect标签 9.5.12 radio标签 9.5.13 optgroup标签 9.5.14 token标签 9.5.15 textarea标签 9.5.16 updownselct标签 9.5.17 password标签 9.5.18 textfield标签 9.6 非表单标签 9.6.1 actionerror和actionmessage标签 9.6.2 component标签 9.6.3 tree和treenode标签 9.7 本章小结 第10章 AJAX技术支持 10.1 AJAX介绍 10.1.1 为什么使用AJAX 10.1.2 AJAX技 10.1.3 常见的浏览器端AJAX框架 10.1.4 常见的服务器端AJAX框架 10.2 Struts 2的AJAX支持 10.2.1 ajax主题 10.2.2 AJAX输入校验 10.2.3 div标签 10.2.4 a标签 10.2.5 submit标签 10.2.6 autocompleter标签 10.2.7 tabbedPanel标签 10.2.8 AJAX表单 10.2.9 widgets 10.3 本章小结 第11章 文件的上传与下载 11.1 Struts 2框架的文件上传 11.1.1 Common-fileUpload组件 11.1.2 文件上传的JSP 11.1.3 文件上传的Action 11.1.4 配置文件和success视图 11.1.5 运行文件上传示例 11.1.6 上传文件的过滤 11.1.7 文件上传的常量 11.2 多个文件上传 11.2.1 使用数组上传多个文件 11.2.2 使用List上传多个文件 11.3 Struts 2控制文件下载 11.3.1 在配置文件中指定下载资源 11.3.2 在Action中指定下载资源 11.3.3 文件下载的权限控制 11.4 本章小结 第三篇 Struts 2框架中整合其他技术 第12章 使用FreeMarker技术 12.1 FreeMarker介绍 12.1.1 FreeMarker基础 12.1.2 FreeMarker简单示例 12.2 Struts 2中使用FreeMarker 12.2.1 FreeMarker使用Struts 2标签库 12.2.2 FreeMarker访问Servlet和JSP对象 12.2.3 使用FreeMarker示例 12.3 本章小结 第13章 整合Spring 13.1 Spring介绍 13.1.1 IoC和DI 13.1.2 Spring优点 13.2 Struts 2合Spring 13.2.1 整合步骤 13.2.2 整合原理 13.3 整合开发示例 13.4 本章小结 章 整合JSF 14.1 JSF介绍 14.1.1 JSF体系结构 14.1.2 JSF同Struts 2的比较 14.1.3

第一图书网, tushu007.com

MyFaces 14.2 Struts 2整合MyFaces 14.2.1 整合步骤 14.2.2 整合原理 14.3 整合MyFaces示例 14.3.1 JSP视图 14.3.2 配置文件 14.3.3 业务控制器Action 14.3 运行示例 14.4 本章小结 第15章 Struts 2访问MySQL数据库 15.1 MySQL下载和安装 15.1.1 下载MySQL数据库 15.1.2 安装MySQL数据库服务器 15.1.3 安装MySQL数据库的JDBC驱动 15.2 访问MySQL数据库示例 15.2.1 创建一个数据库 15.2.2 建立JDBC连接 15.2.3 使用Tomcat数据源 15.2.4 建立一个数据库连接组件 15.3 本章小结 第16章 整合Log4.j 16.1 Log4.j概述 16.2 配置和使用Log4.j 16.2.1 一个简单的例子 16.2.2 Log的配置文件 16.2.3 在代码中使用记录日志 16.3 本章小结 第17章 整合ant 17.1 概述 17.2 安装运行 17.2.1 ant安装配置 17.2.2 HelloWorld实例 17.2.3 Web应用结构 17.2.4 ant运行命令 17.3 build.xml的基本结构 17.4 ant任务 17.4.1 常用内置任务 17.4.2 扩展可选任务 17.5 完整的build.xml例子 17.6 深入build.xml 17.6.1 project 17.6.2 target 17.6.3 task 17.6.4 properties 17.6.5 token filters 17.6.6 path-like structure 17.6.7 命令行变量 17.6.8 references 17.7 Eclipse 3.x中使用ant 17.8 JBuilde 2005中使用ant 17.9 本章小结 第18章 整合JUnit 18.1 JUnit概述 18.2 JUnit安装运行 18.2.1 安装配置 18.2.2 基本测试HelloWorld 18.2.3 JUnit的3种结果界面 18.3 测试Struts 2应用 18.3.1 Struts 2测试概述 18.3.2 创建测试类 18.4 在Eclipse 3.x中执行测试 18.5 在JBuilder 2005中执行测试 18.6 本章小结 第19章 整合Hibernate 19.1 Hibernate简介 19.1.1 Hibernate Hello World应用 19.1.2 理解Hibernate架构 19.1.3 Hibernate的核心接口 19.1.4 Hiebernate基本配置 19.1.5 对象标识符号 19.1.6 Hibernate映射类型 19.1.7 高级映射 19.1.8 Hibernate检索方式 19.2 Struts 2整合Hibernate 19.2.1 下载安装Hibernate 19.2.2 建立示例数据库表 19.2.3 hibernate.cfg.xml配置文件 19.2.4 Hibernate相关代码 19.2.5 DAO代码 19.2.6 Action代码 19.2.7 相关视图 19.2.8 运行示例 19.3 本章小结 第四篇 Struts 2实例验证 第20章 一个示例论坛应用 20.1 项目需求与系统架构 20.1.1 项目需求 20.1.2 系统架构 20.2 数据库设计 20.2.1 数据模型 20.2.2 建立数据库表 20.3 建立开发环境 20.3.1 配置Tomcat数据源 20.3.2 web.xml配置文件 20.3.3 applicationContext.xml配置文件 20.3.4 其他配置文件 20.4 建立数据库连接组件 20.5 建立业务实体对象 20.6 建立数据库访问组件 20.7 建立业务处理模块 20.7.1 权限检查模块 20.7.2 发表文章模块 20.7.3 文章列表导航模块 20.7.4 显示文章内容模块 20.7.5 用户登录模块 20.7.6 用户注册模块 20.8 建立业务控制器 20.8.1 文章处理业务控制器 20.8.2 用户注册业务控制器 20.8.3 用户登录业务控制器 20.9 国际化和输入校验 20.9.1 国际化 20.9.2 输入校验规则文件 20.10 建立视图 20.10.1 论坛列表视图 20.10.2 用户登录视图 20.10.3 用户注册视图 20.10.4 发表文章视图 20.10.5 显示文章视图 20.11 Struts 2配置文件 20.12 运行示例论坛 20.13 本章小结

第一图书网, tushu007.com

章节摘录

第1章 Struts 2概述Struts 1与Tomcat、Turbine等诸多Apache项目一样，是开源软件，这是它的一大优点，使开发者能更深入地了解其内部实现机制。
除此之外，Struts 1的优点主要集中体现在两个方面： Taglib标签库：Taglib是Struts 1的标签库，灵活使用，能大大提高开发效率。
目前国内的JSP开发者，除了使用JSP自带的常用标签外，很少开发自己的标签，自从Struts 1广泛应用以来，很多公司或者开发团队越来越重视自己标签库的开发，这是一个很好的起点。
页面导航：页面导航将是今后的一个发展方向，事实上，这样使系统的脉络更加清晰。
通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。
尤其是当另一批开发者接手这个项目时，这种优势体现得更加明显。
但是，任何优秀的框架也不可能是十全十美的，存在着或多或少的问题，例如在一个复杂的大型应用中，Struts 1很容易引起类爆炸。
同时，在测试方面，编写测试用例类十分不方便，给测试工作带来了许多困难。
尽管现在有一个测试框架Struts Test提供Struts 1的测试编写，但是这样测试方式加剧了复杂化。
提示 Apache官方网站是这样介绍Struts 2的：Apache Struts 2 was originally known as WebWork2.After working independently for several years, the WebWork and Struts communities joinedforces to create Struts 2。
翻译过来大致意思是：Apache Struts 2就是之前大家所熟知的WebWork 2，在经历了几年的各自发展后，WebWork和Struts社区决定合二为一，也即是Struts 2。
WebWork项目已经被Apache所收购，不会再进行升级。
Struts 2中去掉了Struts 1中的ActionForm，实现了同Servlet API的松散耦合。
Struts 2并不是一个全新的架构，而是继承了Struts 1和WebWork的优点，在稳定性、易用性方面都有了很大的提高。
Apache Struts 2 GA版本已经发布，这是Apache Struts 2发行的首个稳定版本，GA意味着General Availability，也就是官方开始推荐广泛使用了。
Struts2 GA版本的发布，意味着核心开发力量将全部转移到Struts 2，对Struts 1的支持将会淡化。
Struts 2的发展前景非常看好，有望在不远的将来完全代替Struts 1。
1.1 MVC介绍 随着Web技术的发展，传统的C/S（客户端/服务器）开发模式正向B/S（浏览器/服务器）模式转换，更多的应用系统采用了B/S结构。
MVC设计思想是Model-View-Controller的简称，即模型-视图-控制器，在Web应用系统的设计开发中被广泛采用。
介绍MVC设计思想，先从Web技术发展说起。
1.1.1 Web技术发展 Web的前身是1980年Tim Berners—Lee负责的Enquire（Enquire Within Upon Everything的简称）项目。
1990年11月，第一个Web服务器nxoc01.CelTl.ch开始运行。
1991年，CERN（European Particle Physics Laboratory）正式发布了Web技术标准。
Web是一种典型的分布式应用架构。
Web应用中的每一次信息交换都要涉及客户端和服务端两个层面。
因此，Web开发技术大体上也可以被分为客户端技术和服务器端技术两大类。
Web客户端的主要任务是展现信息内容，而HTML语言则是信息展现的最有效载体之一。
最初的HTML语言只能在浏览器中展现静态的文本或图像信息，这满足不了人们对信息丰富性和多样性的强烈需求，由静态技术向动态技术的转变成为了Web客户端技术发展方向。
除了编写HTML页面之外，客户端应用的开发者还可以利用一些成熟的技术将浏览器的功能添加到自己的应用程序中。
从1992年开始，W3C向开发者提供libwww开发库。
借助libwww，可以自己编写Web浏览器和Web搜索工具，也可以分析、编辑或显示HTML页面。

1999年，Microsoft在IE 5.0中引入的HTAs（HTML Applications）技术则允许开发者直接将HTML页面转换为一个真正的应用程序。
从1997年的IE 4.0开始，Microsoft为开发者提供了WebBrowser控件和其他相关的COM接口，允许程序员在自己的程序中直接嵌入浏览器窗口，或调用各种浏览器的功能，如分析或编辑HTML页面等。
　　与客户端技术从静态向动态的演进过程类似，Web服务器端的开发技术也是由静态向动态逐渐发展、完善起来的。
第一种真正使服务器能根据运行时的具体情况，动态生成HTML页面的技术是CGI（Common Gateway Interface）技术。
1995年，NCSA（National Centerfor Supercomputing Applications）开始制定GGI 1.1标准。
GGI技术允许服务器端的应用程序根据客户端的请求，动态生成HTML页面，这使客户端和服务器端的动态信息交换成为了可能。
　　随着GGI技术的普及，聊天室、论坛、电子商务、信息查询、全文检索等各式各样的Web应用蓬勃兴起。
早期的CGI程序大多是编译后的可执行程序，其编程语言可以是C、C++、Pascal等任何通用的程序设计语言。
为了简化GGI程序的修改、编译和发布过程，人们开始探寻用脚本语言实现GGI应用的可行方式。
Perl结合了C语言的高效以及sh、awk等脚本语言的便捷，适用于GGI程序的编写。
　　1994年，出现了PHP（Personal Home Page Tools）语言。
与以往的GGI程序不同，PHP语言将HTML代码和PHP指令合成为完整的服务器端动态页面，Web应用的开发者可以用一种更加简便、快捷的方式实现动态Web功能。
1996年，Microsoft发布了ASP技术。
ASP使用的脚本语言是熟悉的VBScript和JavaScript。
ASP迅速成为了Windows系统下Web服务器端的主流开发技术。
1997年，Servlet技术问世，1998年，JSP技术诞生，Servlet和JSP被后来的J2EE平台吸纳为核心技术。
　　ASP技术和JSP技术面世之后，导致了微软和以Sun公司为首的Java体系的竞争。
JSP与微软的ASP十分相似，但事实上，两者是有着本质的不同的，主要从以下几个方面对其进行比较：　　Web服务器的支持：大多数通用的Web服务器，如Apache、Netscape和Microsoft IIS都支持JSP页面。
只有微软本身的Microsoft IIS和Personal Web Server可以支持ASP。
平台的支持：JSP具有平台独立性，只要是一般的Java程序可以运行的平台，都支持JSP程序。
Windows平台可以很好地支持ASP，但ASP对于基于Win32组件模型的依赖，使得它难于移植到其他平台上。
　　组件模型：JSP是建立在可重用的、跨平台的组件（如JavaBeans、Enterprises JavaBeans和用户定制的标签库等组件）之上的。
而ASP使用的是基于Win32的COM组件模型。
脚本语言：JSP可以使用Java编程语言或JavaScript作为脚本语言。
而ASP使用VBScript或JScript作为脚本语言。
　　安全性：JSP使用Java安全模型，而ASP使用Windows NT的安全结构。
与数据库的连接：JSP使用JDBC建立与数据库的连接，而ASP对数据库使用ODBC。
用户定制的标签：JSP可以使用用户定制标签库进行扩充，而ASP中没有用户定制标签库，ASP是不能扩充的。
　　Web服务器端开发技术的完善使开发复杂的Web应用成为了可能。
为了适应企业级应用开发的各种复杂需求，两个最重要的企业级开发平台——J2EE和.NET在2000年前后分别诞生，导致了开发平台之争。
正是这种针锋相对的竞争关系促使了Web开发技术以前所未有的速度提高和跃进。
　　J2EE是纯粹基于Java的解决方案。
1998年，Sun发布了EJB 1.0标准。

第一图书网, tushu007.com

EJB为企业级应用中必不可少的数据封装、事务处理、交易控制等功能提供了良好的技术基础。
至此，J2EE平台的三大核心技术Servlet、JSP和EJB都已先后问世。
1999年，Sun正式发布了J2EE的第一个版本。
紧接着，遵循J2EE标准，为企业级应用提供支撑平台的各类应用服务软件争先恐后地涌现了出来。
IBM的WebSphere、BEA的WebLogic都是这一领域里最为成功的商业软件平台。
随着开源运动的兴起，JBoss等开源世界里的应用服务新秀也吸引了许多用户的注意力。
到2003年时，Sun的J2EE版本已经升级到了l.4版，其中三个关键组件的版本也演进到了Servlet 2.4、JSP 2.0和EJB 2.1。
至此，J2EE体系及相关的软件产品已经成为了Web服务器端开发的一个强有力的支撑环境。
Microsoft的.NET平台是一个强调多语言间交互的通用运行环境。
依靠微软强大的操作系统和开发工具的支持，.NET吸引了很多Web开发者的目光。
C＃语言和CLI这两个技术标准构成了.NET平台的基石，也于2003年成为了ISO的国际标准。
2002年，Microsoft正式发布.NET Framework和Visual Studio.NET开发环境。
AS P.NET超越了ASP的局限，可以使用VB.NET、c#等编译型语言，支持Web Form、.NETServerControl、ADO.NET等高级特性。
客观地讲，.NET平台，尤其是.NET平台中的ASP.NET的确不失为Web开发技术在Windows平台上的一个集大成者。
2000年以后，随着Web应用的日益复杂，人们逐渐意识到，单纯依靠某种技术多半无法达到快速开发、快速验证和快速部署的最佳境界。
研究者开始尝试着将已有的Web开发技术综合起来，形成完整的开发框架或应用模型，并以此来满足各种复杂的应用需求。
在Web服务器端，2000年以后出现了几种主要的技术融合方式。
越来越多的Web开发环境开始支持MVC（Model.View—Contorller）的设计模型，为开发者提供了全套的开发框架。
实际上，J2EE和.NET平台本身就是这种开发框架的典型代表。
1.1.2 MVC设计模式　　MVC是Model一View—Controller的简称，即模型-视图-控制器。
MVC是Xerox PARC在20世纪80年代为编程语言Smalltalk.80发明的一种软件设计模式，至今已被广泛使用。
MVC把应用程序分成3个核心模块：模型（Model）、视图（View）和控制器（Controller），它们分别担当不同的任务。
如图1.2所示显示了这几个模块各自的功能及它们的相互关系。

第一图书网, tushu007.com

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com