高等教育规划教材

Java EE 基础实用教程

主编　崔岩

免费提供电子教案、源代码

下载网址 http://www.cmpedu.com

机械工业出版社
CHINA MACHINE PRESS

高等教育规划教材

Java EE 基础实用教程

主　编　崔　岩
副主编　董洋溢
参　编　陈　勇　吕亚荣　辛向丽

机械工业出版社

Java EE 是一个功能强大的中间件技术平台，也是一个企业级开发的主流平台。本书从该技术最实用、最基础的层面出发。以自顶向下的顺序详细介绍了 JSP 技术、Servlet 技术、JDBC 和 EJB 等最主要的组件技术。在书的后半部分，基于读者在前面对平台的学习，从宏观上介绍了 MVC 的理念与基于 Java EE 的主流框架的关系，并展开介绍了 Strust、Hibernet 与 Spring 框架。最后，通过两个实际的项目案例，详细介绍了多层结构开发的过程，以及相关组件的应用。第一个项目没有基于任何框架，是基于 MVC 理念设计出来的一个论坛系统。第二个项目是基于 Strust 框架的一个信息发布系统的管理子系统的设计。本书的内容阐述深入浅出，逐层递进，讲解生动，并且附有大量的开发实例。读者不仅可以将这些实例作为练习的对象，也可以作为实际工作中的参考。另外，本书的最后两章还可以作为配套实训课程的素材来使用。

本书适合作为高等学校计算机专业的教材，也可以作为相关开发人员的参考书，还可作为计算机开发爱好者的自学用书。

本书配套授课电子课件，需要的教师可登录 www. cmpedu. com 免费注册，审核通过后下载，或联系编辑索取（QQ：2850823885。电话：010 - 88379739）。

图书在版编目（CIP）数据

Java EE 基础实用教程/崔岩主编．—北京：机械工业出版社，2018. 1
高等教育规划教材
ISBN 978-7-111-58682-1

Ⅰ. ①J… Ⅱ. ①崔… Ⅲ. ①JAVA 语言 - 程序设计 - 高等学校 - 教材
Ⅳ. ①TP312. 8

中国版本图书馆 CIP 数据核字（2017）第 321525 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：郝建伟 责任编辑：郝建伟
责任校对：张艳霞 责任印制：张 博
河北鑫兆源印刷有限公司印刷

2018 年 1 月第 1 版 · 第 1 次印刷
184mm × 260mm · 16. 75 印张 · 406 千字
0001 - 3000 册
标准书号：ISBN 978-7-111-58682-1
定价：49. 00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
服务咨询热线：（010）88379833
读者购书热线：（010）88379649

网络服务
机 工 官 网：www. cmpbook. com
机 工 官 博：weibo. com/cmp1952
教育服务网：www. cmpedu. com
金 书 网：www. golden - book. com

出版说明

当前，我国正处在加快转变经济发展方式、推动产业转型升级的关键时期。为产业转型升级提供高层次人才，是高等院校最重要的历史使命和战略任务之一。高等教育要培养基础性、学术型人才，但更重要的是加大力度培养多层次、多样化的应用型、复合型人才。

为顺应高等教育迅猛发展的趋势，配合高等院校的教学改革，满足各院校对高质量教材的迫切需求，机械工业出版社邀请了全国多所高等院校的专家、一线教师及教务部门，通过充分的调研和讨论，针对相关课程的特点，总结教学中的实践经验，组织出版了这套“高等教育规划教材”。

本套教材具有以下特点：

1）符合高等院校的人才培养目标及课程体系设置，注重培养学生的应用能力，加大案例篇幅或实训内容，强调知识、能力与素质的综合训练。

2）针对多数学生的学习特点，采用通俗易懂的方法讲解知识，逻辑性强、层次分明、叙述准确而精炼、图文并茂，使学生可以快速掌握，学以致用。

3）凝结一线骨干教师的课程改革和教学研究成果，融合先进的教学理念，在教学内容和方法上做出创新。

4）为了体现建设“立体化”精品教材的宗旨，本套教材为主干课程配备了电子教案、学习与上机指导、习题解答、源代码或源程序、教学大纲、课程设计和毕业设计指导等资源。

5）注重教材的实用性、通用性，适合各类高等院校、高等职业学校及相关院校的教学，也可作为各类培训班教材和自学用书。

欢迎教育界的专家和老师提出宝贵的意见和建议。衷心感谢广大教育工作者和读者的支持与帮助！

机械工业出版社

前　言

Java EE 是目前最为主流的企业级开发平台，它提供了一套从设计到开发、部署的一系列完整的规范。本书在力求全面系统地介绍 Java EE 体系结构的同时，突出实用性和基础性。附有大量实例，基本做到“一事一例”，每个知识点都通过例子来展现，力求使读者能够通过实践了解其中的理论，从而更快地掌握相关的技术和知识。

写作目的：

作者作为常年在一线从事计算机专业技术教学的老师，常常感到很多学生十分欠缺实际动手能力。除常规的问题外，还有一个原因就是所选用的教材只注重理论、过于晦涩和抽象而让很多学生敬而远之。目前很多有关 Java EE 的教材大多能体现出作者极高的水平，不过在内容上有的比较抽象，有的内容庞大而忽略细节。作为教学老师，一直在寻找一本能够适应当前初学者学习的入门教材。因此，作者产生了编写本书的想法，目的是想从实际教学出发，编写一本合适的教材。

本书特色：

本书在内容选择上是有所取舍的，并没有为了突出全面而将所有相关的技术全部呈现出来。因为这样做很可能让初学者面对众多的内容无所适从，抓不住重点，反而影响学习效果。这里的取舍主要是从初学者的角度出发，首先把最贴近实际操作、最实用的知识介绍给读者，随着内容的逐步推进，再介绍一些复杂的、高级的、抽象的技术。在有限的篇幅内，尽力做到有的放矢，使每一页的文字都能起到学习的效果。对于需要读者扩展的知识和内容，也都在书中给予了提示，为读者指明自学研究的方向。

本书的另外一个特色是有大量的实例可供读者学习。本书的实例注重的是实用性，在阐述理论和技术时，先强调这个技术的实用场景是什么，然后再配以实例，这样能让读者更加直观和深刻地理解所学习的技术。不仅要让读者会用，还要让他们知道在什么时候用。

在开发环境的选择上，本书并没有选择目前非常流行的 Eclipse 作为演示工具，而是选择了甲骨文公司的免费学习工具 NetBeans。之所以选择 NetBeans，是因为它是目前唯一一个集成了完全兼容 Java EE 6 规范的应用服务器的开发环境，它极大地减少了开发环境在搭建方面的配置问题，特别适合初学者使用。

适用读者：

本书适合对 Java 语言已经有一定的了解，想要学习有关 Web 开发的读者。本书分为两部分，前 5 章是基础篇，作为初学者可反复研读这部分的内容。掌握之后可以学习进阶篇（包括第 6 章到第 9 章的内容），这部分是对一些主流框架的介绍。最后两章是两个综合开发项目实例，其中，第 10 章是针对基础篇的一个总结应用，第 11 章是针对进阶篇的一个应用。

编写人员：

本书是由多位具有多年教学经验的教师合作编写而成的，在总结实际教学经验和以往的开发经验的同时，引进新的概念和想法。其中第1章、第9~11章由崔岩编写，第5章、第7、8章由董洋溢编写，第3章由陈勇、崔岩共同编写，第6章由陈勇编写，第2章由吕亚荣编写，第4章由辛向丽、崔岩共同编写。崔岩负责全书统稿。

感谢本书的责任编辑，他对本书提出了很多宝贵的建议，也感谢他的耐心等待！

由于作者水平有限，加之时间和精力的限制，本书也并没有完全体现作者的想法。书中难免存在疏漏和不足之处，欢迎广大读者指正批评。

编　者

目　录

出版说明

前言

第 1 章　Java EE 概述 …… 1

1.1　Java EE 的由来 …… 1

1.1.1　软件开发的发展历程 …… 1

1.1.2　企业级软件项目开发的体系结构 …… 3

1.2　认识 Java EE …… 5

1.2.1　Java EE 简介 …… 5

1.2.2　Java EE 的编程思想（容器—组件） …… 5

1.3　Java EE 的架构 …… 7

1.3.1　Java EE 的技术框架 …… 7

1.3.2　Java EE 的优势 …… 9

1.4　开发工具与环境搭建 …… 9

1.4.1　NetBeans IDE 工具介绍 …… 9

1.4.2　NetBeans IDE 的安装 …… 10

第 2 章　JSP …… 12

2.1　JSP 概述 …… 12

2.1.1　JSP 简介 …… 12

2.1.2　JSP 的工作原理 …… 13

2.1.3　JSP 实例 …… 14

2.2　JSP 脚本 …… 16

2.2.1　JSP 脚本的基本形式 …… 16

2.2.2　对象的声明 …… 17

2.2.3　输出表达式 …… 20

2.2.4　注释的使用 …… 21

2.3　指令与动作组件 …… 22

2.3.1　page 指令 …… 22

2.3.2　include 指令 …… 25

2.3.3　动作组件 …… 27

2.4　内置对象 …… 32

2.4.1　常用的内置对象 …… 32

2.4.2　内置对象的作用范围 …… 49

2.5　表达式语言——EL …… 50

2.5.1　基本语法 …… 50

2.5.2　隐式对象 …… 53

2.6　JSP 的标签 …… 57

2.6.1　标签简介 …… 57

2.6.2　标准标签库 JSTL …… 57

2.6.3　自定义标签 …… 59

2.7　思考与练习 …… 62

第 3 章　JavaBean …… 63

3.1　JavaBean 概述 …… 63

3.1.1　JavaBean 简介 …… 63

3.1.2　JavaBean 的特征 …… 64

3.1.3　JavaBean 的特征实现 …… 65

3.1.4　创建一个 JavaBean 文件 …… 67

3.2　JavaBean 在 JSP 中的应用 …… 70

3.2.1　JSP 的标签 …… 70

3.2.2　调用的基本形式 …… 72

3.2.3　JavaBean 与 JSP 的参数传递 …… 73

3.2.4　JavaBean 的生命周期 …… 75

3.3　思考与练习 …… 80

第 4 章　Servlet …… 81

4.1　Servlet 概述 …… 81

4.1.1　Servlet 简介 …… 81

4.1.2　Servlet 的工作原理与生命周期 …… 82

4.1.3　创建第一个 Servlet …… 83

4.1.4　web.xml 文件 …… 86

4.2　请求与响应 …… 86

4.2.1　处理表单的参数 …… 87

4.2.2　Header 与初始化参数 …… 90

4.2.3　发送非网页文档 …… 93

4.2.4　转发与重定向 …… 94

4.3 会话跟踪 …… 96
4.3.1 Cookie …… 96
4.3.2 URL 参数传递与重写 …… 99
4.3.3 Session …… 101
4.3.4 Servlet 的上下文 …… 105
4.4 过滤器 …… 108
4.4.1 过滤器简介 …… 108
4.4.2 创建过滤器 …… 109
4.5 侦听器 …… 114
4.5.1 侦听器的工作原理 …… 114
4.5.2 创建侦听器 …… 116
4.6 思考与练习 …… 119
第 5 章 JDBC …… 120
5.1 JDBC 概述 …… 120
5.2 搭建 JDBC 环境 …… 121
5.2.1 在 MySQL 中创建数据 …… 121
5.2.2 添加 JDBC 驱动 …… 124
5.3 连接数据库 …… 125
5.3.1 建立连接 …… 125
5.3.2 简单查询 Statement …… 125
5.3.3 带参数查询 PreparedStatement … 126
5.3.4 使用存储过程 …… 127
5.3.5 向数据库中插入数据 …… 129
5.4 数据的更新和删除 …… 130
5.4.1 数据的更新 …… 130
5.4.2 数据的删除 …… 131
5.5 两种结果集的使用 …… 132
5.5.1 ResultSet 类 …… 132
5.5.2 RowSet 接口 …… 135
5.6 思考与练习 …… 145
第 6 章 MVC 与框架 …… 146
6.1 MVC 模式概述 …… 146
6.1.1 MVC 模式简介 …… 146
6.1.2 MVC 模式基础 …… 146
6.1.3 MVC 模式的作用 …… 148
6.1.4 Java EE 中的 MVC …… 150
6.2 框架的概念 …… 152
6.2.1 框架概述 …… 152
6.2.2 框架和设计模式的关系 …… 152
6.2.3 框架的作用 …… 153
6.3 主流框架介绍 …… 154
6.3.1 Struts 框架 …… 154
6.3.2 Hibernate 框架 …… 155
6.3.3 Spring 框架 …… 155
6.3.4 JSF 框架 …… 155
6.4 思考与练习 …… 156
第 7 章 Hibernate 框架 …… 157
7.1 框架简介 …… 157
7.1.1 Hibernate 框架简介 …… 157
7.1.2 POJO 简介 …… 158
7.1.3 Hibernate 的核心接口 …… 159
7.2 Hibernate 对象关系映射 …… 162
7.2.1 对象关系映射的基本概念 …… 162
7.2.2 基本类映射过程 …… 163
7.2.3 关系映射类型 …… 164
7.3 创建一个 Hibernate 项目 …… 167
7.3.1 Hibernate 项目开发的一般步骤 …… 167
7.3.2 Hibernate 项目实例 …… 168
7.4 Hibernate 逆向工程 …… 174
7.5 思考与练习 …… 180
第 8 章 Struts2 框架 …… 181
8.1 Struts2 框架简介 …… 181
8.1.1 Struts2 的发展历程 …… 181
8.1.2 Struts2 的工作原理 …… 183
8.1.3 Struts2 的软件包 …… 184
8.1.4 Struts1. x 和 Struts2. x 框架对比 …… 185
8.2 创建 Struts1. x 项目 …… 186
8.2.1 在 NetBeans 环境下创建 Struts1. x 项目 …… 186
8.2.2 Struts1. x 配置文件解析 …… 189
8.3 创建一个 Struts2 项目 …… 197
8.3.1 Struts2 项目的创建 …… 197
8.3.2 Struts2 项目文件解析 …… 199
8.4 创建 Struts2 自定义项目 …… 202
8.5 思考与练习 …… 207
第 9 章 Spring 框架 …… 208
9.1 Spring 简介 …… 208
9.1.1 Spring 的内部结构 …… 208

9.1.2 Spring 的工作原理 …… 209
9.1.3 依赖注入的方式 …… 212
9.2 IoC 的主要组件 …… 213
9.2.1 通过一个例子来了解 IoC …… 213
9.2.2 Bean …… 215
9.2.3 BeanFactory …… 216
9.2.4 ApplicationContext …… 220
9.3 Spring MVC …… 221
9.3.1 Spring MVC 的工作原理 …… 221
9.3.2 创建一个 MVC 项目 …… 222
9.3.3 配置自己的页面文件 …… 225
9.4 思考与练习 …… 227
第 10 章 基于 MVC 模式的论坛发布系统的设计与实现 …… 228
10.1 项目概述 …… 228
10.2 概要设计 …… 228
10.3 详细设计与编码实现 …… 229
10.3.1 数据库的设计 …… 229
10.3.2 创建数据访问公共模块 …… 230
10.3.3 登录模块 …… 231
10.3.4 用户注册 …… 235
10.3.5 用户发帖 …… 235
10.3.6 用户回帖 …… 240
10.3.7 用户管理 …… 243
10.3.8 身份认证 …… 243
第 11 章 基于 Struts 的校园兼职信息网的后台管理设计 …… 247
11.1 项目概述 …… 247
11.2 概要设计 …… 247
11.2.1 系统架构设计 …… 247
11.2.2 数据库设计 …… 248
11.2.3 功能模块设计 …… 249
11.3 详细设计与编码实现 …… 249
11.3.1 用户登录 …… 249
11.3.2 职位信息发布 …… 253
11.3.3 职位信息管理 …… 257
参考文献 …… 260

第1章　Java EE概述

本章主要介绍Java EE的基础概念，以及与之相关的一些组件和技术。通过对Java EE体系的宏观性介绍，阐述企业级软件开发的基本体系结构。通过对容器—组件的介绍，阐述Java EE的编程思想及优势。

本章的重点内容为Java EE的编程思想、体系结构，以及它的容器与组件。

1.1　Java EE的由来

1.1.1　软件开发的发展历程

计算机软件开发相对于其他传统的生产领域，是一个比较年轻的行业。但是回顾计算机软件开发的历史，不难发现，其实它与其他传统生产领域中的技术发展走了相似的道路。虽然计算机技术本身是先进的、具有开创性的，但是如果抽象到策略和全局的层面，它与传统的工业制造的发展历程非常相似，甚至是仍在按照常规的发展路线进行。所以，它离人们的距离并不遥远。

对于初学Java EE的学习者，完全没必要把计算机软件开发技术想得太过玄妙。这些软件开发技术的原理都来自传统的工作方法。通过学习本节知识，希望读者能对软件开发的发展方向和原理有一个清晰、完整的认知。只有这样，才能在以后学习软件开发的道路上，对那些纷繁复杂、不断推陈出新的软件技术，不会觉得太过迷茫和畏惧。只要看透了本质，就能知道自己应该学习什么。

仅从计算机行业的发展来看，计算机软件的发展受到了市场需求和硬件的推动与制约，同样，软件的发展也推动了市场需求和硬件的发展。目前，比较主流的软件技术的发展历程大致可分为3个不同阶段：第一个阶段是软件技术发展的早期（20世纪40年代至60年代），作者称之为个人作品阶段；第二个阶段是结构化程序及高级语言大发展时期（60年代至80年代），作者称之为作坊生产阶段；第三个阶段是从80年代到现在，是软件工程技术及面向对象语言的大发展时期，作者称之为软件开发的工业革命。

1. 软件开发的个人作品阶段

这个阶段人们对于计算机的概念比较陌生，它产生的原因也是为了解决某个问题。第一台通用计算机ENIAC于1946年在美国宾夕法尼亚大学诞生，其发明人是美国人莫克利（JohnW. Mauchly）和艾克特（J. Presper Eckert）。设计它的目的，是为美国国防部的某个科研项目进行弹道计算。

在那个时期，纯粹的计算机软件开发还没有出现。为了自己的科研项目，科研人员设计出了计算机硬件；为了控制这些硬件，又设计出了可以运行在其上的程序，这就是最早的计算机程序。而且这些程序的作用往往只是计算几个数学模型，应用的范围非常窄，主要集中在科研与工程计算方面，处理的对象大都是数值型数据。

到这个阶段的末期1956年，在J. Backus的领导下为IBM机器研制出了第一个实用高级语言Fortran及其翻译程序。此后，又有多种高级语言问世，从而使程序设计和编制的功效大为提高。另一方面，高级语言更为接近人类语言的语法结构，便于理解和学习，降低了学习计算机语言的门槛，从而在客观上让更多的人可以学习并使用它。

计算机软件初期发展的阶段和人类原始社会对工具的使用非常相似。原始社会，人们使用的工具并没有被专业化地从其他东西中分离出来，原始人为了完成一个事情，随手找到的木棍、石片等经过简单处理就成为工具。而这个工具也仅仅是为了这个人的这件事情而作，其应用范围也很有限。

2. 结构化程序及高级语言大发展时期

由于计算机的硬件成本越来越低，体积越来越小，计算能力越来越强，人们对计算机的应用需求也开始急速增长。从以前的科学实验，到商业领域的应用，市场对计算机软件开发提出了前所未有的要求。

结构化程序设计成为当时一个流行的开发理论。这期间，诞生了如Pascal、Ada等一系列的结构化语言。这些语言具有较为清晰的控制结构，与之前的高级程序语言相比有一定的优势。此外，高级语言的功能也发生了一定的分化，出现了专门应对数据模型控制的程序语言，即数据库语言。这使得在应用方面，程序语言的能力有了进一步提升，所能实现的业务也更加广阔。

但进入到这个阶段的后期，软件危机的爆发，让人们似乎看不到软件开发的未来。由于软件项目内容越来越复杂，技术上的要求越来越多样化、专业化。而软件公司的开发水平仍然处在一个个人作坊的水平，没有一个客观标准的开发方法，也没有一个科学合理的项目管理方式。大部分的项目开发都是以开发人员的个人经验来进行设计和实现的，最后的结果是投入的人越多，开发的效率越低，开发的周期却越来越长。

这个阶段的发展和人类社会生产力水平的发展也非常相似。进入封建社会，人类的生产力与原始社会比起来有了极大的提升，生产工具越来越专业化，所能做的内容几乎涵盖了各个方面。但这个时期的制造业规模不大，也没有办法应对极大数量和大规模的制造需求，产品的生产效率不高。

3. 软件工程技术及面向对象语言的大发展时期

由于在上一个阶段遇到了软件危机，为了解决这个问题，一个新的概念——“软件工程”被提出。那么，软件工程究竟是什么？其实就是一套完整的软件项目的开发方法和评价体系，以及软件项目的项目管理办法。它并不是介绍一个具体项目如何开发，而是抽象出了所有软件开发项目的共性和本质，告诉开发者遇到复杂的应用需求时，应该如何通过一套标准规范的开发步骤，把复杂的问题简单化，从而得出项目开发的结果——软件产品。它的核心其实就是：抽象——分解——标准化。

有一位美国著名的管理学家说过，中餐没有办法出现美国那样知名的快餐企业是业务管理上没有做好。虽然中餐的菜品看起来味道的好坏取决于做饭厨师的功力，但是如果使用现代科学化的管理，一样可以让每一个分店都做出一样味道的菜品。他用的方法就是抽象——分解——标准化。如果把一道鱼香肉丝的加工过程进行抽象处理，可以分解成十几道工序，然后再对每道工序进行严格的量化要求，就可以实现每次做出来的菜都是一个味道，这就是科学管理的威力。

软件开发其实和厨师做菜有一定的相似性，都是以个人主观的判断和设计来进行工作，

而且都要按照一定的规则来进行。因此上面的例子同样也在软件开发领域有着相同的意义。有了软件工程，就可以把人的主观因素降到最低点，通过统一的标准化管理与规范的开发步骤，让每个程序员都可以写出合理的程序。

如果说软件工程是从开发策略上寻求解决软件危机的方式，那么面向对象的程序设计语言的出现则是从高级语言本身出发找到了一个化繁为简的解决方案。以 Java 语言为代表，面向对象的程序设计几乎在 20 世纪末横扫天下，成为解决所有复杂问题的不二之选。客观分析，面向对象语言的最基本原则其实和软件工程是类似的，仍然是抽象——分解——标准化。首先对应用项目进行抽象，所有的业务几乎都变成了类。但这个类是一个静态的描述，如果要让它动起来工作，就要做一个实例化对象。所以，类就像是工业生产中的设计图纸，实例化对象的过程就是按照图纸生产产品的过程。

可以说，在这个时期软件开发进入了前所未有的大发展阶段，一方面软件工程每隔几年就会有更为先进的开发理论产生，促进项目开发的效果提升，另一方面，面向对象程序设计语言的发展，使得软件的能力越来越强。软件的复用性和可移植性大大提高，从而极大地提高了软件项目开发的效率。此外，集成开发环境及中间件技术的发展使得开发工具也有了极大的提升，将开发人员从一些底层的、重复性的程序开发中解放出来，从而腾出更多的时间来进行软件结构和功能上的设计。

这个阶段的发展非常像人类社会的工业文明后的发展时代。一战时期被提出的标准化零件的管理理念不就是软件工程中提到的标准化吗？所有产品被分解成若干个几乎不能再分的零件，然后统一标准，按照设计图纸来进行设计。这不就是面向对象程序设计中的类与对象的关系吗！此外，制造企业中，流水线生产方式的大规模应用，使得产品的投入与产出的效率得到极大提升，这个流水线不就像是软件开发中越来越强大的集成开发环境吗！

所以，通过对软件开发历程的回顾和梳理，可以发现，从本质上看，软件开发的历程正在复制人类社会生产力发展的轨迹。所以，将来软件的发展一定也符合目前社会生产的发展趋势。

在作者看来，其实还应该有第 4 个阶段，这就是互联网及分布式系统的普及，它开创了软件开发的又一个新的起点和方向。目前软件开发方向是进一步构件化，大块头的企业级开发已经有所减少，取而代之的是基于互联网技术的轻量级应用的发展。在具体开发上，每个模块越来越专一，越来越专业；从应用的角度上看，反而是越来越融合，越来越集成。

最典型的代表就是智能手机相关的软件开发。从应用的角度，它集成了无线通信技术、互联网技术、软件通信技术、单片机技术和嵌入式技术等。但从单个模块来看，手机的每个功能都是由一个专业的软件在进行控制。而这些专业的软件就像是一块块积木一样，通过统一标准的接口，可以和其他模块或功能相互交流数据。它们彼此的独立性很高，单独换掉哪一个都不会影响其他模块的工作。

由于篇幅所限，对于软件开发的发展只能先介绍到这里，作者想通过上面的内容向读者展示出软件的开发设计方法其实并不是特有的，它是符合当前社会生产规律的东西。它未来的发展也一定会继续按照这个轨迹发展下去。

1.1.2 企业级软件项目开发的体系结构

这里首先需要读者明确一个概念，什么是企业级应用程序。这里的企业级是指那些应用

规模巨大，集成了很多应用功能，需要处理巨量数据的软件开发项目。一般来说，企业级软件开发规模应该具有以下几个特点。

1）基于网络的应用。企业级的应用程序，规模应用如此巨大，一定不是集成在一两台机器之上，它的应用应该是基于一个范围更为宽广的网络之上，所以必定要有网络方面的应用需求。

2）巨量的数据集成。在这样的一个应用当中，存放着企业的巨量信息，这些信息会作为数据库中的数据被保存起来。其中还包括之前应用系统的数据资源。

3）高度的安全性。由于是基于网络的应用，而数据又都是至少具有商业级的保密要求，因此，提供一个可靠、安全、稳定的系统是必不可少的特性之一。

4）具备可扩展性。当前的应用发展非常迅速，对于企业级应用软件，所服务的对象也是海量的用户，对于用户群里不断增加的新需求，必须要具备一定的可扩展性来适应用户不断提出的新需求。

企业级软件项目的体系也有一个逐步发展的过程。总的来讲，目前有两个大的方向，一种是C/S体系结构；另一种是B/S体系结构。后一种结构的发展晚于第一种，但后来成为更多企业级软件项目的首选结构。但是这不能说明C/S结构就走到了尽头，目前仍然有很多软件是采用C/S体系结构的。

1. C/S体系结构

C/S体系结构由客户端（Client）和服务器（Server）两部分构成。用户要使用这个系统，首先必须安装它的客户端。通过客户端来完成用户与服务器的交互。在具体处理过程中，客户端负责人机界面的交互及业务控制方面的操作，服务器端主要负责数据的交互和保存。

这样的系统安全性强，通信效率高，能处理大量数据，操作相对简单，交互性强。不过它也有缺点，客户端的块头较大，使用不太方便，必须先安装客户端，可扩展性低；维护与修改工作量较大。

目前最常见的C/S应用软件就是腾讯公司的QQ，除此之外还有手机中安装的各式各样的客户端，都是C/S结构的软件产品。

2. B/S体系结构

B/S体系结构也由两部分组成：浏览器（Browser）和服务器（Server）。这个结构也被人称为“瘦客户端”，主要是与C/S相比。对于浏览器，读者一定非常熟悉，就是泛指在日常上网时浏览网页的浏览器工具。这个浏览器与C/S中的客户端相比体积小了很多。但是从用户的角度来看，似乎它起到了与客户端一样的作用，因此才把它称为瘦客户端。

不过，从内部功能结构上来看，浏览器和C/S中的客户端完全不是一个概念。B/S体系结构中的浏览器是一个工具，通过这个工具，用户可以看到应用软件从服务器发送过来的相关信息。所以浏览器真的只起到了浏览的作用，它仅仅把程序需要传递的界面在浏览器中呈现出来，本身不对数据做任何业务处理和控制。与C/S相比，客户端的部分业务控制功能已经全部放到了服务器端。服务器内部进行了一个分层，应用服务器负责实现业务处理和控制的工作，它可以近似地认为替代了C/S中的客户端部分的功能。数据库服务器负责对数据库的管理和对数据的具体交互。

B/S体系的优点非常明显，首先，它对用户的硬件要求不高，兼容性极强。只要能上网且有浏览器，就可以与服务器进行交互完成应用。其次，维护起来非常容易，因为它在用户

方没有安装任何程序，所以软件自身的修改与升级只需要在服务器端完成即可，对用户没有任何影响，实施起来很方便。此外，基于 B/S 的服务器端存在一个业务应用服务器和一个数据库服务器，这样的分层有效地使程序和数据分离，提高了它们的独立性，每个系统都可以根据需要进行扩展和修改，而将对彼此的影响降到最低。本书要介绍的 Java EE 就是针对 B/S 体系结构提出的解决方案。

📖 由于服务器端又被分为应用服务器和数据库服务器，再加上浏览器，就构成了逻辑上的三层结构。所以 B/S 体系也被称为三层体系结构。目前还有很多文献提出了多层体系结构的说法，但其实质上还是基于三层体系结构的，只不过是对应用服务器又进行了更为详细的划分。

1.2 认识 Java EE

1.2.1 Java EE 简介

Java EE 的全称为 Java Platform Enterprise Edition。它是基于 Java 语言的一种软件设计体系结构，所以它不是一种语言，也不是集成开发环境，而是一种标准中间件体系结构。Java EE 的作用在于能够标准化企业级多层结构应用系统的部署，并且简化开发过程。就好像要盖一座大楼，所采用的是砖混结构还是框架结构一样。

前面介绍过，软件开发的历程实际上就是一个不断标准化、专业化、抽象化的过程。从结构化程序设计到面向对象程序设计，是从语法结构的表达和分析上的抽象过程。从面向对象程序设计到 Java EE 体系结构的提出，是进一步抽象了开发过程中的应用对象，并且对开发过程中各个组件之间的接口进行了统一的、标准化的规范。

一个典型的 Java EE 结构的软件应用系统，从逻辑上划分，包括 3 个层：表示层、业务层及数据持久层。表示层负责的是对客户端的响应，并进行一定的业务控制（转发和指派）；业务层主要负责对业务数据的具体控制和响应，并负责对具体数据发起编辑请求；数据持久层主要负责对数据库系统的控制和管理，业务层的数据请求都需要通过持久层的处理来完成与具体数据库数据的交互。

Java EE 就是通过上述 3 个层面的设计，给出了一个标准的软件架构和设计方案。由于它的影响力巨大，所以在这个体系里不仅有 Java EE 本身设计出来的具体组件支持这个系统，其他厂商的软件产品为了提高自身产品的使用率，也都会遵循 Java EE 的标准来设计相关的组件。所以发展到现在，Java EE 所涉及的内容越来越多，使用到的组件也越来越多。很多初学者都因为其庞大的内容望而却步。

1.2.2 Java EE 的编程思想（容器—组件）

容器—组件的编程思想，其实就是把完成具体功能的工具以组件的形式“装入”一个容器之中。这个“装入”是逻辑上的，具体的实现方式是要求组件对所有数据的接收和发送必须通过容器才能完成。或者反过来理解，客户端发出的请求，是由容器来分发到相关的组件进行处理，处理的结果组件会交给容器，由容器来决定最后的输出方式。

这样做最大的好处是可以用统一的标准来处理数据，并且让容器与系统的其他部分保持

很高的独立性。这就好像是一些单位的传达室，所有部门人员的进出都必须经过传达室。特别是外来人员，必须经过一个特定的手续才可以进入，如果出门携带大件物品还需要通过特定的手续才可以放行。这就是统一标准，而且还要对来往的人员（数据）进行审核，防止发生异常。

读者会发现，其实这个思想是面向对象中类的封装性的一种延续。只不过类封装的是一个特定的数据模式。而容器封装的是更大的一组特定的功能。这仍然是延续了软件开发的发展趋势，不断的抽象——分解——标准化。

Java EE 的容器会给组件提供一些底层的基础功能，这些功能的主要作用是完成组件与外界的数据互交。为了实现组件与容器之间既保持着相对独立的状态，又有比较强大的数据交互能力，这就需要提供一个统一的标准规范。Java EE 就是这个标准规范的提供者。

最常见的 Java EE 的容器是 Web 容器及 EJB 容器。它们所包含的组件会在容器的 Java 虚拟机中进行初始化。组件根据容器提供的标准服务来与外界进行互交。这些服务主要包括命名服务、数据库连接服务、持久化、消息服务、事务支持和安全性服务等。也就是说，大部分底层的基础功能都由容器提供。这样就可以让开发者不用关心底层互交的数据环境而把精力专注于对业务逻辑的设计上，从而大大提高对组件的开发效率。

从实现原理上来看，容器与组件之间的通信除了通过 Java 程序本身的算法完成具体操作外，更为重要的是通过一个部署描述文件来解决容器如何向组件提供服务，提供哪种服务的问题。这个文件是一个用 XML 文件即扩展性标记语言写成的文件。它通过标记语言的形式，详细地描述在应用当中组件需要调用容器的哪种服务，以及它们的名称、参数等。这个部署描述文件就像一个说明书或是地图，系统在工作时会通过部署描述文件来调用响应的服务。

通过一个部署描述文件的方式协调系统之间的工作方式并不是什么创新，日常生活中，这样的部署描述文件比比皆是。例如，新买回家的一个烤箱，厂家会附送一个烘焙菜谱和使用说明书，这是烘焙系统的部署描述文件；新买回家的全自动洗衣机，里面的使用说明书会告诉用户，洗什么样的衣服用什么程序，选择多少水和洗衣液。这些都是部署描述文件所起的作用。

读者一定有这样的生活常识，现实生活中，越是操作和功能复杂的设备，它的说明书就越复杂。同理，在 Java EE 的系统中，实现的是企业级的应用，因此它的部署描述文件的内容是非常复杂而庞大的。当然，为了最大限度地减轻这个文件给编程人员造成的工作负荷，设计人员也在不断地优化它的设计。

Java EE 5 规范推出了支持在组件中实现代码直接对注解的引用，在很大程度上取代了复杂的部署文件中的配置内容。注解的方式是在 JDK 5 的版本中就已经推出的一种机制，它可以在 Java 源代码中通过嵌入元数据的方式配置和部署文件，在实际编程中非常实用。有兴趣的读者，可以自己查看相关资料。本书在后面的章节中对这个技术有所应用，但由于重点不在这里，所以没有介绍原理和方法。

除了支持注释机制，Java EE 6 规范还对配置做了模板化的设置，也就是所谓“异常才配置”。对于组件的属性和行为，系统的容器会按照一个预设的方式自动进行配置，开发人员可以省略具体配置过程。只有当对某个属性或行为做一个特殊的配置时，才需要对部署描述文件进行具体配置。通过上述方式，就会让编写部署配置文件的过程变得容易，从而提高整个系统的开发效率。

1.3 Java EE 的架构

1.3.1 Java EE 的技术框架

一个面向企业级的并且支持分布式的应用开发标准，它的整个结构是非常庞大的。从技术的角度来划分，完整的 Java EE 分成了 4 个部分：组件技术、服务技术、通信技术及架构技术。

读者如果之前看到过一些介绍 Java EE 架构的资料，可能会觉得里面的架构图看起来内容繁多，而且其中大都是不认识的专有名词。这对于初学者来说，往往会让人对学习 Java EE 产生困惑或信心不足。其实，对于大部分学习应用级开发的人来说，架构图中的大部分东西都不是能够轻易地修改和编辑的，它们的作用是支持应用级编程的一些底层工具。大部分时间，开发者都接触不到这个层面的编程或设置。

当然，并不是说这些东西不用了解，而是对于初学 Java EE 的人来说，可以暂时不用考虑这些技术，只专注于与应用级开发相关的技术即可。等到以后对这个系统有了一个比较全面的认识后，就可以向底层的方向学习。这对于拓展一个程序员的能力是非常有好处的。

因此，这里给大家提供一个被作者简化的体系结构图，图中只把读者可能要学习到的技术标明出来。那些暂时接触不到的部分统一用“支持技术”表示。Java EE 体系结构如图 1-1 所示。

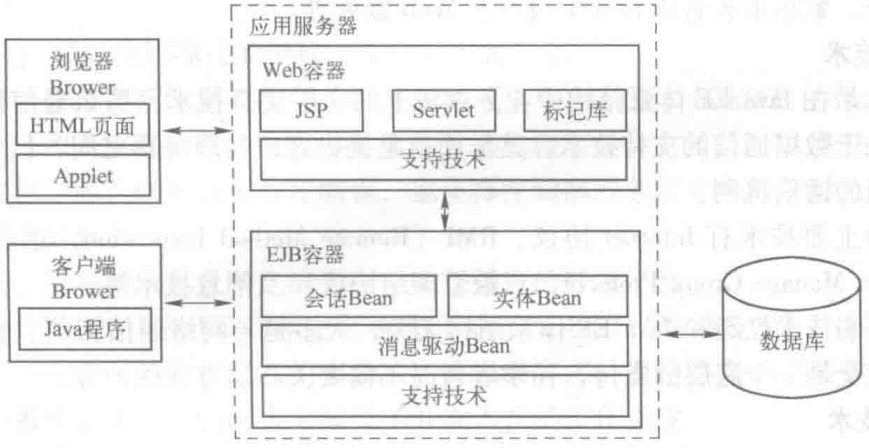

图 1-1 Java EE 体系结构

1. 组件技术

在前面提到的架构的 4 个组成部分中，初学者接触最多的是它的组件技术。组件，顾名思义，是具体完成程序开发过程中的组成部分。所以，这部分主要是指与具体开发相关的工具和技术。

图 1-1 中除了数据库部分，几乎所有“看得见”的部分都有组件的身影。在客户端部分，浏览器是第三方软件，但是它起到了向服务器发送内容和进行数据互交的作用。此外还有动态部分的体现，Applet 小程序负责这部分工作，并且也会在客户端运行。浏览器与应用服务器构成了 B/S 架构的软件模型。

客户端是指由 Java 语言编写出来的程序，它们不需要应用服务器的 Web 容器进行响应，

可以直接通过与业务层面的控制程序进行连接，完成具体的应用。当然客户端程序是需要在客户的机器上安装后才可以使用的。客户端与应用服务器构成了 C/S 架构的软件模型。

在图 1-1 中的应用服务器，构成了 Java EE 体系结构中最核心的部分。其中 Web 容器中的组件实现了基于 HTTP 协议的 Web 请求与响应。这里的主要技术是 JSP 与 Servlet，JSP 负责面向客户端与浏览器进行互交，Servlet 负责设计的响应与处理指派，主要起控制作用。标记库是一个辅助技术，目的是让程序更加简洁，层次更加明晰。它主要的应用方向大都集中在 JSP 文件中。

EJB 容器中主要包含了对业务逻辑方面的处理，例如会话 Bean、实体 Bean 与消息驱动 Bean。它们的主要作用是响应 Web 容器提供的一些数据业务的请求。它们并不直接面对客户端，也不分析客户端的信息，只是对有关数据的业务请求进行处理。也可以近似地认为它们的作用是对底层数据库的操作。

2. 服务技术

服务，顾名思义，主要作用不是体现在对客户的应用上，而是对内容业务处理提供的支持。就好像是服务产业，如家政公司提供的服务、公交公司提供的服务等，它们本身不产生新的价值，却为社会化大生产提供了有力支持。

在 Java EE 的体系结构中，服务技术主要是为容器与组件之间提供各种支持。因此，在大部分情况下，应用级开发者都感觉不到服务技术的存在，因为大部分是系统已经配置好的，开发者只需要在这个环境之上做应用即可。

这些服务大都集中在图 1-1 中的“支持技术”中。比较常用的有命名服务、部署服务、数据连接服务、数据事务管理和安全服务及 Web 服务等。

3. 通信技术

服务技术是在 Java EE 体系结构中业务逻辑上的底层支持技术，那么通信技术是一个更为底层、有关于数据通信的支持技术，其作用就是提供客户与服务器之间，以及应用服务器内部容器之间的通信机制。

实现它的主要技术有 Internet 协议、RMI（Remote Method Invocation，远程方法调用）、OMGP（Object Manage Group Protocol，对象管理组协议）及消息技术等。

这里的有些技术已经和 Java EE 体系结构无关，大多是与网络通信相关。它们对于应用级开发来说完全是一个底层的支持，初学者暂时不需要关心这方面的内容。

4. 架构技术

架构技术可以说是诞生最晚的一个组成部分，直到 Java EE 6 规范才有了明确的架构标准。这里首先要明确架构是什么？其实这个架构是从软件具体结构实现的角度，从宏观上去分析和设计一个企业级的应用系统，应该遵循的架构标准。

推出这个规范的原因主要是因为 Java EE 之前的规范只是对容器—组件之间的交互方式进行了规范化的设计。而对于整个软件应用系统的构架方面没有给出统一的标准。与此同时，在实际应用中，一些公司在开发项目时，根据自己的经验给出了一些开发的框架，后来很多开发者使用并推广，就形成了如今一些比较知名的设计框架，例如，著名的三大框架 Struts、Hibernate 及 Spring。

基于市场的需求，Java EE 给出了自己的架构标准。目前最主要的架构标准有两个，一个是 JSF（Java Server Faces），它是一种侧重于构建 Web 应用的表示层框架的标准，提供了一种以组件为中心事件驱动的用户界面构建方法。另一个是 JPA（Java Persistence API，Java