

SAP NetWeaver 精要丛书

卡尔·凯斯勒(Karl Kessler)
彼得·蒂勒特(Peter Tillert)
帕纳尤特·多布林科(Panayot Dobrikov)

等著



SAP

应用服务器的JAVA编程 (影印版)

JAVA Programming with the SAP Web Application Server

東方出版社

 PRESS

SAP

应用服务器的JAVA编程

JAVA Programming with the SAP Web Application Server(影印版)

卡尔·凯斯勒(Karl Kessler)
彼得·蒂勒特(Peter Tillert)
帕纳尤特·多布林科(Panayot Dobrikov) 等著

東方出版社



PRESS

责任编辑: 吴秋淑
封面设计: 左 涛

图书在版编目(CIP)数据

SAP 应用服务器的JAVA编程/凯斯勒 等著;
北京:东方出版社, 2005.6
ISBN 7-5060-2228-1
I. S… II. 凯… III. ①计算机②软件 VI. F97
中国版本图书馆CIP数据核字(2005)第025787号

SAP 应用服务器的JAVA编程 (影印版)

卡尔·凯斯勒(Karl Kessler)
彼得·蒂勒特(Peter Tillert) 等著
帕纳尤特·多布林科(Panayot Dobrikov)

出版发行 东方出版社
地 址 北京朝阳门内大街166号 邮政编码 100086
电 话 010—82665724 (编辑部) 010—82665462 (发行部)
010—82679079 (生产部)
网 址 <http://www.erptraining.cn>
经 销 新华书店
印 刷 山东新华印刷厂临沂厂
开 本 787×1092毫米 1/16 版 次 2005年6月第1版
印 张 32.50 印 次 2005年6月第1次
字 数 600 000 定 价 80.00元

版权所有 侵权必究 印装出错 负责调换

简明目录

第1章 SAP NetWeaver	17
1.1 企业软件平台的必要性	17
1.2 作为一个集成平台的SAP NetWeaver	19
1.3 SAP NetWeaver 组件	23
1.4 展望：SAP NetWeaver 作为企业服务架构的平台	32
第2章 SAP NetWeaver 开发者工作室： 特性、工具和“视角”	35
2.1 用户接口	35
2.2 工作场所、项目和开发对象	37
2.3 特性	39
2.4 体系结构	48
2.5 工具和视角	50
第3章 SAP NetWeaver 开发者工作室： 循序渐进开发示例应用程序	65
3.1 雇员管理示例应用程序	66
3.2 开始使用SAP NetWeaver 开发者工作室	67
3.3 定义数据模型	68
3.4 访问表数据	75

3.5 定义业务逻辑	81
3.6 创建部署描述	86
3.7 创建基于JSP的 Web 应用	93
3.8 定义和部署完整的J2EE应用	100
3.9 把业务逻辑作为Web服务来提供	103

第4章 SAP Web应用服务器中的JAVA保持 107

4.1 Open JDBC	107
4.2 SAP Web应用服务器的运行时保持性架构 ...	108
4.3 JAVA 字典	112
4.4 开发一个示例应用程序	115
4.5 Open SQL/SQLJ	119
4.6 企业级 JAVABeans ——容器管理的持久性	131
4.7 使用JAVA 数据对象编程	143
4.8 适用于专家的持久性	164

第5章 SAP 应用服务器中的 Web服务 167

5.1 Web 服务框架	168
5.2 标准化Web 服务	169
5.3 提供Web服务—— 服务器端	171
5.4 在公众目录中提供Web 服务	194
5.5 使用Web服务—— 客户端	202

第6章 Web Dynpro: 开发用户界面 213

6.1 使用视图设计器	213
6.2 使用Adobe 技术的互动窗体	220

6.3 具有多视图的应用	223
6.4 生成和定制源代码：使用事件句柄	230
6.5 动态改变用户界面	238
6.6 Web Dynpro 组件	242
6.7 控件之间的通信	249
6.8 总结	251

第7章 Web Dynpro:开发业务应用 255

7.1 Web Dynpro调用一个Web服务 —— 通过五个步骤建立一个应用	256
7.2 Web Dynpro 控件和其接口	278
7.3 在前端和后端之间的Contexts和数据流	290
7.4 使用数据类型信息	308

第8章 SAP NetWeaver JAVA 开发架构： 组件模型和服务 321

8.1 大型软件项目的特性	321
8.2 SAP NetWeaver Java 开发架构包含的元素	326

第9章 SAP NetWeaver Java 开发架构： 循序渐进开发示例应用程序 371

9.1 雇员案例应用程序	372
9.2 开始使用SAP NetWeaver JDI	373
9.3 定义一个数据模型	377
9.4 提供对数据表和业务逻辑的访问	384

9.5 创建基于JSP的用户界面	385
9.6 创建并测试完整的J2EE应用程序	388
9.7 使开发对象在架构中集中可用	389

第10章 SAP NetWeaver Java 开发架构： 配置与系统管理 393

10.1 SAP NetWeaver Java 开发架构的配置	393
10.2 SAP NetWeaver Java 开发架构的系统管理	418

第11章 SAP Web 应用服务器的架构 439

11.1 SAP Web 应用服务器的全貌	440
11.2 簇生命周期管理	448
11.3 可伸缩性和高可用性	455
11.4 远程调试	466
11.5 运行时体系结构概览	468

第12章 SAP Web 应用服务器的维护 483

12.1 登录	483
12.2 监控	485
12.3 系统管理	492
12.4 性能分析	498

附录

关于作者	501
索引	505

Contents

Preface	13
1 SAP NetWeaver	17
1.1 The Necessity of a Platform for Enterprise Software	17
1.1.1 Motivation for Introducing a Technological Platform	17
1.1.2 SAP Basis as an Example of a Successful Technology Platform ...	17
1.1.3 Architectural Features of SAP Basis	17
1.2 SAP NetWeaver as an Integration Platform	19
1.2.1 Integration within a System	19
1.2.2 Standards Enable Integration	20
1.2.3 Example of an Integration Scenario: Invoice Verification	21
1.2.4 Component Architecture of Invoice Verification	21
1.2.5 Requirements for an Integration Platform	22
1.3 SAP NetWeaver Components	23
1.3.1 People and Information Integration	23
1.3.2 Process Integration	28
1.3.3 Application Platform	30
1.4 Prospects: SAP NetWeaver as a Platform for the Enterprise Services Architecture	32
2 SAP NetWeaver Developer Studio: Features, Tools, and Perspectives	35
2.1 User Interface	35
2.2 Workspace, Projects, and Development Objects	37
2.3 Features	39
2.3.1 Integrating the Java Development Infrastructure	39
2.3.2 Server Integration in the Developer Studio	43
2.4 Architecture	48
2.5 Tools and Perspectives	50
2.5.1 Development Configuration Perspective	51
2.5.2 Dictionary Perspective	53
2.5.3 J2EE Perspective	55
2.5.4 Web Dynpro Perspective	57
2.5.5 Web Service Perspective	60
2.5.6 DTR Perspective	61

3 SAP NetWeaver Developer Studio: Step by Step to the Example Application 65

3.1	Employee Example Application	66
3.2	First Steps in the SAP NetWeaver Developer Studio	67
3.2.1	Starting the Developer Studio	67
3.2.2	Settings Under Windows Preferences	68
3.3	Defining the Data Model	68
3.3.1	Creating a Dictionary Project	69
3.3.2	Defining the Employee Table	70
3.3.3	Deploying the Table	73
3.4	Accessing Table Data	75
3.4.1	Creating an EJB Module Project	75
3.4.2	Defining the Employee Entity Bean	76
3.4.3	Creating a Data Transfer Object Class	79
3.5	Defining the Business Logic	81
3.5.1	Creating Session Bean EmployeeServices	81
3.5.2	Implementing the Bean Class	83
3.6	Creating Deployment Descriptions	86
3.6.1	Creating Descriptions in the ejb-jar.xml	87
3.6.2	Creating Server-Specific Deployment Descriptions	91
3.6.3	Creating the Java Archive	93
3.7	Creating a JSP-Based Web Application	93
3.7.1	Creating a Web Module Project	94
3.7.2	Implementing the User Interface with JSP	95
3.7.3	Descriptions in the Deployment Descriptor web.xml	98
3.7.4	Creating the Web Archive	100
3.8	Defining and Deploying the Entire J2EE Application	100
3.8.1	Creating an Enterprise Application Project	101
3.8.2	Creating Descriptions for application.xml	101
3.8.3	Creating a Data Source alias	102
3.8.4	Creating and Deploying the Enterprise Application Archive (EAR)	102
3.9	Providing Business Logic as a Web Service	103
3.9.1	Creating a Web Service	103
3.9.2	Deploying the Web Service	105
3.9.3	Calling the Web Service in the Test Environment	105

4 Java Persistence in the SAP Web Application Server 107

4.1	Open JDBC for Java	107
4.1.1	Features of Open JDBC for Java	108

4.2	Persistence Infrastructure of the SAP Web AS at Runtime	108
4.2.1	Vendor JDBC	109
4.2.2	Native JDBC	109
4.2.3	Statement Pooling	109
4.2.4	SQL Trace	110
4.2.5	Table Buffering	111
4.2.6	Managing Database Connections	111
4.3	Java Dictionary	112
4.3.1	Defining Database Tables in the SAP NetWeaver Developer Studio	113
4.4	Developing an Example Application	115
4.4.1	Example Project Management Scenario	115
4.4.2	Definition and Deployment of the Tables	117
4.4.3	Implementation of the Auxiliary Classes	118
4.4.4	Implementation of the Session Bean and Deployment of the Application	119
4.5	Open SQL/SQLJ	119
4.5.1	Open SQL/SQLJ: Basic Principles	120
4.5.2	Implementing the Project Management Scenario with Open SQL/SQLJ	121
4.6	Enterprise JavaBeans—Container-Managed Persistence	131
4.6.1	Implementing the Project Management Scenario Using EJB CMP	132
4.7	Programming with Java Data Objects	143
4.7.1	JDO Concepts	144
4.7.2	Preparing the JDO Project	145
4.7.3	Programming the Example Classes	145
4.7.4	Modifying Bytecode with the Enhancer	149
4.7.5	Object-Relational Mapping of Persistent Classes to the Database	153
4.7.6	Programming the Application Logic	156
4.7.7	Deploying the Application	163
4.8	Persistence for Experts	164

5 Web Services in the SAP Web Application Server 167

5.1	The Web Service Framework	168
5.2	Standardizing Web Services	169
5.3	Providing a Web Service—The Server Side	171
5.3.1	Creating a Web Service	173
5.3.2	Testing a Web Service with the Web Service Home Page	181
5.3.3	Securing the Web Service	183
5.3.4	Restrictions for Web Service End Points	192
5.3.5	Supported Types in Web Service End Points	193

5.4	Providing Web Services in Public Directories	194
5.4.1	Publishing a Web Service in a Registry	200
5.4.2	Creating a Business Service	201
5.5	Consuming a Web Service—The Client Side	202
5.5.1	Calling the WSDL Description of the Web Service	202
5.5.2	Generating a Web Service Proxy	203
5.5.3	Implementing the Client Application	205

6 Web Dynpro: Developing User Interfaces 213

6.1	Working with the View Designer	213
6.1.1	Views and Layouts	213
6.1.2	Local Data: The Context of a View	216
6.1.3	Binding Data	217
6.2	Interactive Forms with Adobe Technology	220
6.2.1	The Adobe Forms Designer in the Developer Studio	220
6.2.2	Further Scenarios with Interactive Forms	221
6.3	Applications with Multiple Views	223
6.3.1	Data Transfer Between Multiple Views Using Mapping	223
6.3.2	Arranging Views in the Navigation Modeler	225
6.3.3	Round Trips and Actions	226
6.3.4	Navigation from One View to Another	228
6.4	Generated and Custom Source Code: Working with Event Handlers	230
6.4.1	Event Handler for Actions	230
6.4.2	Other Types of Event Handlers	232
6.4.3	API of the Web Dynpro Runtime Environment	233
6.4.4	Simple Code Examples for the Web Dynpro API	233
6.5	Changing the User Interface Dynamically	238
6.5.1	Creating Context Elements Dynamically	238
6.5.2	Creating UI Elements Dynamically	239
6.5.3	Binding UI Elements to Actions	241
6.5.4	Statically and Dynamically Created Elements	242
6.6	Web Dynpro Components	242
6.6.1	Components and Applications	243
6.6.2	Components as Reusable Units	244
6.6.3	Example of a Reusable Component	246
6.7	Communication Between Controllers	249
6.7.1	Usage Relationships	249
6.7.2	Communication Using Events	250
6.8	Summary	251
6.8.1	The Metamodel of Web Dynpro	252
6.8.2	Outlook	254

7 Web Dynpro: Developing Business Applications 255

7.1	Web Dynpro Calls a Web Service—Five Steps to an Application	256
7.1.1	Importing the Model for the Employee Web Service	259
7.1.2	Binding a Custom Controller to the Model	263
7.1.3	Defining the Context Mappings	267
7.1.4	View Layout and Data Binding	269
7.1.5	Implementing Controllers	271
7.2	Web Dynpro Controllers and Their Interfaces	278
7.2.1	Model View Controller Model	278
7.2.2	Controller Concept	280
7.2.3	Generated and Generic Controller APIs	285
7.2.4	Shortcut Variables wdControllerAPI and wdComponentAPI	286
7.3	Contexts and Data Flow Between Backend and Frontend	290
7.3.1	Context Concept	291
7.3.2	Typed and Generic Context APIs	293
7.3.3	Context Programming Basics	296
7.3.4	Supply Functions	298
7.3.5	Calculated Context Attributes	301
7.3.6	Web Dynpro Models and the Common Model Interface	303
7.3.7	Accessing the Business Logic with the Common Model Interface	305
7.3.8	Differences Between Model Nodes and Value Nodes in the Context	306
7.4	Using Data Type Information	308
7.4.1	Data Type Interfaces	310
7.4.2	Local Dictionary	310
7.4.3	Using Input Helps	311
7.4.4	Dynamic Data Types	316
7.4.5	Structures and Strengths of Adaptive RFC Models	317

8 SAP NetWeaver Java Development Infrastructure: Component Model and Services 321

8.1	Special Characteristics of Large Software Projects	321
8.1.1	Example of a Typical Development Process Without Central Infrastructure	322
8.1.2	Software Logistics in the Java Development	324
8.2	Elements of the SAP NetWeaver Java Development Infrastructure	326
8.2.1	SAP Component Model	327
8.2.2	Design Time Repository	340
8.2.3	Component Build Service (CBS)	353
8.2.4	Change Management Service (CMS)	359

8.2.5	Software Logistics in the Development Process with the SAP NetWeaver Java Data Infrastructure	366
-------	--	-----

9 SAP NetWeaver Java Development Infrastructure: Step by Step to the Example Application 371

9.1	The Employee Example Application	372
9.2	First Steps with the SAP NetWeaver JDI	373
9.2.1	Preparations by the Administration of the SAP NetWeaver JDI	374
9.2.2	Importing the Development Configuration	375
9.3	Defining a Data Model	377
9.3.1	Creating a Dictionary DC Project	377
9.3.2	Defining the Employee Table	381
9.3.3	Defining Public Parts of the Development Components	381
9.3.4	Building and Deploying Development Components	383
9.4	Providing Access to Table Data and Business Logic	384
9.4.1	Creating an EJB Module DC	384
9.5	Creating a JSP-Based User Interface	385
9.6	Creating and Testing the Entire J2EE Application	388
9.7	Making Development Objects Centrally Available	389
9.7.1	Checking in Source Files and Activating All DCs	389
9.7.2	Releasing Changes	390

10 SAP NetWeaver Java Development Infrastructure: Configuration and Administration 393

10.1	Configuration of the SAP NetWeaver Java Development Infrastructure	393
10.1.1	Java Development Landscape	393
10.1.2	Setting Up a SAP NetWeaver Java Development Infrastructure	397
10.2	Administration of the SAP NetWeaver Java Development Infrastructure	418
10.2.1	Product Definition in the SLD	418
10.2.2	Namespace Prefix	419
10.2.3	Preparing a Track	423
10.2.4	Development in the Java Development Infrastructure	429
10.2.5	Consolidation Phase	435
10.2.6	Assembling the Software and Quality Assurance	435
10.2.7	Delivery to the Customers	437

11	The Architecture of the SAP Web Application Server	439
11.1	SAP Web Application Server Landscape	440
11.1.1	Cluster Design Concepts	441
11.1.2	Components of the Cluster	442
11.1.3	Cluster Landscape Options	446
11.2	Cluster Lifecycle Management	448
11.3	Scalability and High Availability	455
11.3.1	Request Processing and Load Balancing	456
11.3.2	High Availability	459
11.3.3	Session Failover	463
11.4	Remote Debugging	466
11.5	Runtime Architecture Overview	468
11.5.1	The Java Kernel	470
11.5.2	Infrastructure Layer	472
11.5.3	Classloading and Isolation	474
11.5.4	Deployment Service and J2EE Containers	481
12	Supportability of SAP Web Application Server	483
12.1	Logging	483
12.2	Monitoring	485
12.2.1	JMX Infrastructure	485
12.2.2	Monitors	488
12.2.3	Adding New Content in the Monitoring Framework	490
12.2.4	Accessing Monitoring Data from an External Client	490
12.3	Administration	492
12.3.1	Administration Infrastructure	492
12.3.2	Possibilities to Extend Visual Administrator	495
12.3.3	Providing a User Interface for a Newly Installed Service	496
12.3.4	Adding a New Managed Object in the Administration Tree	496
12.3.5	Other Administrative Tools	497
12.4	Performance Analysis	498
12.4.1	Application Tracing	498
12.4.2	Single Activity Trace	499
12.4.3	SQL Tracing	499
	The Authors	501
	Index	505

Preface

SAP NetWeaver is based on the SAP Web Application Server—the technological basis of all modern SAP solutions. Without the SAP Web Application Server (SAP Web AS), no Enterprise Portal, Exchange Infrastructure, Business Information Warehouse, or SAP enterprise software would be able to run. SAP Web AS is the platform on which applications can be developed and operated. The success of SAP's software depends largely on the efficiency and robustness of SAP Web AS.

As we reflect over the past decades, SAP's implementation of the application server at the end of the 1980s and the beginning of the 1990s was revolutionary. SAP's competitors—in so far as they existed at the time—were building two-tier applications. Some competitors moved the entire business logic to the frontend, creating the extremely cost-intensive *fat client*. This was difficult to scale as numbers of users increased, since every user process was assigned to a shadow process in the database. Other SAP competitors moved the business logic to stored procedures running directly in the database, but soon learned that with stored procedures an application couldn't be run on a database platform without recoding the entire application.

SAP's solution to this problem was to insert a separate tier between the frontend and the database: the *application tier*. This was when the application server—as we know it today—was born—even outside the context of SAP.

The application server from the days of R/3 still had all the features of a modern application server. It abstracted data from the underlying operating system and database platform, and optimized access to database resources using a sophisticated transaction and update concept. It provided a platform-independent programming language (ABAP) that was tailored to meet the requirements of professional enterprise software.

Many ideas from the world of Java, such as generating bytecode interpreted by a virtual machine, have been state of the art in the ABAP world for a long time. Another advantage of Java was that the development and runtime environments were interlinked. There's probably no other system in which you can perform all development work without ever having to leave the system. As the system compiles itself incrementally after changes, you no longer have the long generation times that were standard for classic C and Java development.

The swift success of the R/3 System on the market led to the application server being widely distributed. At the same time, the Internet and the World Wide Web (WWW) took off at an incredible speed. Proprietary worlds were no longer in demand; instead, people wanted open standards that were adopted by the Internet community. And the primary standard was the Java programming language. Even though its core wasn't designed for business applications, it was popular because it was simple and platform-independent. Consequently, Java became quickly widespread (largely because of the Internet).

But not all concepts from those early years had any chance of success: Downloading entire applications on request from the Internet onto simple devices still remains an illusion. Endeavors to define a Java-based application server that different software providers could implement were taken more seriously, although it wasn't until well-known providers (including SAP) actually began to implement the J2EE standard that Java had any real opportunity for success in the field of enterprise software.

However, implementing the standard came at a price. You had to implement many functions that were not imperative for business applications. Conversely, the J2EE standard could not sufficiently define many requirements essential for a business application, primarily the user interface, which is fundamental for business applications because an application's acceptance by end users depends largely on the user interface. This is where SAP's many years of experience with enterprise software came into play. Even in the earliest versions of SAP's software, the user dialog was the focal point. This tradition is currently supported by Web Dynpro for SAP Web AS. Web Dynpro represents an application's user interface running in the SAP Enterprise Portal and, as such, is often the starting point for a development project. Powerful tools such as the SAP NetWeaver Developer Studio (NWDS) help you to design your applications as prototypes, while Web Dynpro supports a highly declarative, model-oriented approach.

In addition to the user interface, elements and issues such as business logic, service-orientation, persistence, scalability, and maintainability are vitally important for the success of an application. A comprehensive chapter is therefore dedicated to each of these issues, starting with **Chapter 1**, which focuses on SAP Web Application Server's positioning within SAP NetWeaver and introduces the most important SAP NetWeaver components.