

Chris Whealy

Inside Web Dynpro for Java

 HOCHSCHULE
LIECHTENSTEIN
Bibliothek

Galileo Press

Bonn • Boston

Contents

Author's Apology — Preface to the Second Edition.....	21
---	----

Part I: Background

Introduction..... «..

1.1	Why Did SAP Decide to Build Their Own Web Development Framework?.....	29
1.2	What Is Web Dynpro?.....	30
1.3	Causes of Poor Web Dynpro Implementations.....	32
1.4	How Web Dynpro Differs from Other Web Development Tools.....	32
1.5	The Fundamental Difference Between Web Dynpro and Other Web Development Tools.....	34
1.6	A Quick History Lesson.....	35
1.6.1	The Original Design Concept.....	35
1.6.2	The Web Dynpro Implementation.....	36
1.7	The Design Philosophy Behind Web Dynpro.....	38
1.7.1	Design Influences.....	38
1.7.2	Why Do I Need an SAP-Supplied Server To Run a Web Dynpro Application?.....	39
1.8	The Influence of MVC on Web Dynpro.....	40
1.8.1	The User's Mental Model.....	40
1.8.2	A Top-Down Approach.....	41
1.8.3	Interacting with Business Objects.....	41
1.8.4	Simplifying the Editor.....	42
1.8.5	The Essence of the MVC Design Concept.....	43
1.8.6	The Data Consumer-Generator Concept in Web Dynpro.....	45
1.9	The Influence of SAP's Requirements on Web Dynpro.....	45
1.9.1	The Consequences of a Declarative Approach to UI Design.....	45
1.9.2	Decoupling Data Presentation from Data Processing ...	46
1.10	The Influence of Focus Groups on Web Dynpro.....	47
1.11	What Do I Need To Understand To Be Successful with Web Dynpro?.....	47
1.11.1	Using Web Dynpro to Reproduce the User's Mental Model of the Data.....	48

- 1.11.2 Understanding the Data Generator-Consumer Concept.....48
- 1.12 Is Writing a Web Dynpro Program Just a Matter of Writing Models, Views, and Controllers?48

Part II: The Fundamental Concepts

2 The Web Dynpro Component Concept 53

- 2.1 A Typical MVC Architecture.....53
- 2.2 The Web Dynpro Component Concept.....55
 - 2.2.1 How Does a Web Dynpro Component Alter Reenskaug's Original Design? 56
 - 2.2.2 Why Bother with Components?.....57
- 2.3 Inside the Web Dynpro Component.....58
- 2.4 What Do the Different Parts of the Component Do?.....60
 - 2.4.1 The Programmatic Parts of a Component.....60
 - 2.4.2 The Visual Parts of a Component.....62
- 2.5 Inside a Web Dynpro Controller.....65
 - 2.5.1 Architecture of Non-Visual Controllers.....65
 - 2.5.2 Architecture of Visual Controllers.....72
 - 2.5.3 Controller Interaction.....78
- 2.6 Communication Between Controllers.....79
 - 2.6.1 Usage Declarations79
 - 2.6.2 Context Mapping.....80
 - 2.6.3 Data Binding.....81
 - 2.6.4 Summary of the Roles Played by the Different Controllers.....82
- 2.7 Web Dynpro Applications or "Where Does My Component Sit in the Big Scheme of Things?".....83
 - 2.7.1 Accessing Web Dynpro Functionality from a Browser.....83
 - 2.7.2 Simple Application Structure.....84
 - 2.7.3 More Realistic Application Structure.....84
 - 2.7.4 Application Properties.....85
 - 2.7.5 Application Parameters.....86
 - 2.7.6 Start Up and Shut Down.....86
- 2.8 Managing Web Dynpro Components.....87

3 Desirftng a Web Dynpro Application«»».*.^,**-«-. 89

3.1	The SAP Component.Mpdel.....,,-,.....	89
3.1.1	Development Component Naming Conventions.....	91
3.1.2	Child Development Components.....	92
3.1.3	Software Components.....	93
3.1.4	Full Component Hierarchy.....	93
3.1.5	Developing Without the NWDI.....:	94
3.2	Architectural Concepts of the NWDI].....,.....	95
3.2.1	The Design Time Repository (DTR).....	96
3.2.2	The Component Build Service (CBS).....	97
3.2.3	The Change Management Service (CMS).....	97
3.3	Building Web Dynpro Applications Using the NWDI.....:	98
3.3.1	A Frequently Asked Question.....	99
3.3.2	Sharing Web Dynpro Functionality Between Multiple DCs.....	100
3.4	Preparation and Planning.....	103
3.4.1	The Rule of the-Seven Ps.....	104
3.4.2	You Get What You Pay For.....	104
3.4.3	There Is No Substitute for Good Training and Experience.....	104
3.4.4	Oh and by the Way, Can You Just Add this Functionality? !.....	105
3.4.5	The Partner Over-Sells Themselves.....	105
3.4.6	Functional Scope.....	106
3.4.7	Realistic Deadlines and Trained Project Staff.....	107
3.4.8	What's the Value of Not Making a Mistake?.....	108
3.4.9	Access to Business Functionality.....	110
3.4.10	Planning the User Interface: Creating a Story Board. . .	111
3.4.11	Analysis of the Business Process from Various Perspectives.....	111
3.5	Design.....	113
3.5.1	Avoid Monolithic Coding Structures.....	114
3.5.2	Avoid Excessive Fragmentation.....:	114
3.5.3	Getting the Balance Right.....	115
3.5.4	Component Hierarchy.....	115
3.5.5	Using Model Components in the Component Hierarchy.....7.....	120
3.5.6	Component Interface Definitions.....	122
3.5.7	Configuration Requirements.....	125
3.5.8	Localization Requirements.....	126

3.5.9	Location of Business Processing.....	127
3.6	Implementation.....	129
3.6.1	Management of Child Component Instances.....	129
3.6.2	Intercomponent Communication.....	129
3.6.3	Internal Data Storage Within Web Dynpro Components.....	135
3.6.4	Centralized Functionality.....	137
3.6.5	Correct Use of Standard Hook Methods.....	138
4.1	Phase Model Processing and Component Instantiation.....	143
4.2	Web Dynpro Hook Methods.....	143
4.3	The Relationship Between Phase Model Processing and Web Dynpro Windows.....	144
4.4	Phase Model Processing.....	145
4.4.1	Transport Data to Data Container.....	145
4.4.2	doApplicationStateChange; The Floating Step — Part 1.....	145
4.4.3	Validate and Transport Data.....	147
4.4.4	Call System Event Handlers.....	149
4.4.5	Call Service Event Handlers.....	149
4.4.6	Application Event Handlers.....	151
4.4.7	doBeforeNavigation.....	152
4.4.8	Navigation and View Initialization.....	153
4.4.9	Dynamic UI Manipulation.....	155
4.4.10	Client Specific UI Validation.....	156
4.4.11	doPostProcessing.....	157
4.4.12	doApplicationStateChange:The Floating Step- Part 2.....	158
4.4.13	Response Rendering.....	159
4.5	Summary.....	159

Part III: Basic Development

5 The Context at Design Time

5.1	Nodes.....	163
5.2	Attributes.....	164
5.3	Node and Attribute Data Types.....	164
5.4	Context Attribute Data Types that Can Supply Data to UI Elements.....	167

5.5	Terminology Concerning Nodes and Attributes.....	168
5.6	What's the Difference Between Value Nodes and Model Nodes?.....	169
5.6.1	Value Nodes.....	169
5.6.2	Model Nodes.....	170
5.7	The Most Important Node Properties to Understand.....*	171
5.7.1	Node Cardinality.....	172
5.7.2	Node Selection Cardinality.....	174
5.7.3	The Singleton Property.....	175
5.8	Should Node Names Be Singular or Plural?.....	177
5.9	General Naming Conventions.....!	178
5.10	Identifying a Specific Node Attribute.....	179
5.11	Calculated Attributes.....	180
5.11.1	Implementing a Getter Method.....	182
5.11.2	Implementing a Setter Method.....	183
5.11.3	Coding Principles for Getter and Setter Methods.....	185
5.12	Internal Context Mapping.....	186
5.13'	External Context Mapping.....	187
5.14	Supply Functions.....	189
5.15	Dispose Functions.....	194
5.16	Coding Principles for Supply and Dispose Functions.....-	200
5.17	Recursive Context Nodes.....	200
5.17.1	Creating Recursive Context Nodes.....	201
5.17.2	Mapping Recursive Context Nodes.....	203
5.17.3	Restrictions on the Use of Recursive Nodes.....	204
5.18	Fundamental Principles of the Context.....	205

Context Structure at Runtime ...-«.....*.....» 207

6.1	Generated Classes.....	207
6.1.1	The Typed Context API.....;	207
6.1.2	Background to the typedAccessRequired Flag.....	209
6.1.3	The Byte Code Minimizer.....	210
6.1.4	Generated Classes and Context Mapping.....	211
6.2	Working with Value Nodes.....	213
6.2.1	Accessing the Context Through the Typed API.....	213
6.2.2	Accessing a Node Element.....	214
6.2.3	Accessing all Elements in a Node.....	215
6.2.4	Creating a New Node Element.....	216
6.2.5	Treating the Node Lineltems as a Singleton Node.....	218
6.2.6	Treating the Node Lineltems as a Non-Singleton Node.....	221

6.2.7	Accessing the Context Through the Generic API.....	224
6.3	Working with Model Nodes.....	230
6.3.1	Adding Elements to a Model Node Collection.....	230
6.3.2	Binding Elements to a Model Node Collection.....	232
6.4	Working with Recursive Nodes.....	233
6.5	Using the WDCopyService.....	233
7.1	How are UI Elements Stored Within a View Controller?.....	237
7.2	Editing a View Layout.....	238
7.2.1	Adding UI Elements to the View Layout.....	238
7.2.2	Editing Existing UI Elements on the View Layout.....	239
7.3	Putting Data on the Screen.....	239
7.3.1	Data Binding.....	239
7.3.2	Binding UI Properties to the Context.....	240
7.3.3	Controlling UI Element Behavior from the Context.....	242
7.3.4	Impact of Data Binding on the Coding.....	243
7.4	Avoiding a Possible Source of Confusion.....	244
7.5	Layout Managers.....	246
7.5.1	Flow Layout.....	247
7.5.2	Row Layout.....	247
7.5.3	Matrix Layout.....	248
7.5.4	Grid Layout.....	249
7.5.5	Layout Manager Properties.....	251
7.5.6	Principles for the Efficient Use of layout Managers.....	253
7.6	Composite UI Elements.....	254
7.6.1	A Simple Composite UI Element.....	254
7.6.2	Composite, Composite UI Elements.....	254
7.6.3	Binding Table UI Elements to the Context.....	255
7.6.4	Runtime Behavior of a Table UI Element.....	256
7.7	UI Element Events and View Controller Actions.....	258
7.7.1	Declaration and Association with a UI Element Event.....	258
7.7.2	Implementation.....	260
7.7.3	Action Types.....	261
7.7.4	Event Types.....	262
7.7.5	Coding Generated by an Action Declaration.....	263
7.8	Parameter Mapping.....	264
7.8.1	Multiple UI Element Events Handled by One Action Event Handler.....	267

- 7.8.2 Further Decoupling of the UI..... 269
- 7.8.3 Advanced Parameter Mapping Example..... 271
- 7.9 Generic UI Services'..... 272
 - 7.9.1 Simple Value Selector..... 273
 - 7.9.2 Extended Value Selector..... 273
 - 7.9.3 Extending the List of Values Shown in a Simple or
Extended Value Selector..... 275
 - 7.9.4 Object Value Selector..... 276
 - 7.9.5 Implementing an Object Value Selector..... 277
- 7.10 Using Messages and the Message Manager..... 278
 - 7.10.1 Location of Messages on the Screen 278
 - 7.10.2 Accessing the Message Manager..... 279
 - 7.10.3 Reporting a Simple Message..... 279
 - 7.10.4 Locale-Dependent Text at Runtime..... 280
 - 7.10.5 Reporting a Message Held in the Message Pool..... 281
 - 7.10.6 Relating Messages with UI Elements..... 282
 - 7.10.7 Locale-Dependent Text Defined in the Java
Dictionary..... 283
 - 7.10.8 Defining Placeholders Within a Message Text..... 285
 - 7A0.9 Process Control Using the Message Manager..... 286

- 8.1 Internationalization..... 287
- 8.2 Externalization..... 287
- 8.3 Web Dynpro i18n Concept..... 288
- 8.4 S2X: SAP's Use of the XLIFF Standard..... 289
- 8.5 Storing Language-Specific Text in XLF Files..... 290
- 8.6 Translating XLF Files..... 291
- 8.7 Use of the S2X Editor Within NWDS..... 291
- 8.8. Editing Message Pool XLF Files..... 292
- 8.9 Runtime Locale Determination..... 294

Part IV: Advanced Development

9 Dynamic Context Manipulation

- 9.1 Dynamic Addition of an Unmapped Context Node..... 297
 - 9.1.1 Before the Coding Details Are Discussed, Though 298
 - 9.1.2 Now for the Coding..... 299
 - 9.1.3 Dynamic Creation of a Hierarchy of Context Nodes. . . . 302

- 9.2 Dynamic Addition of a Mapped Context Node303
 - 9.2.1 Technique 1.....306
 - 9.2.2 Technique 2.....307
 - 9.2.3 Method replicateNodeInfoO.308

- 10.1 Working with Tree UI Elements.....311
 - 10.1.1 Types of Tree UI Element.....311
 - 10.1.2 Building a Tree Whose Depth Is Known at Design Time312
 - 10.1.3 Building a Tree Whose Depth Is Unknown at Design Time.....312
 - 10.1.4 Building a Context; Node Suitable, for a Tree UI Element.....313
 - 10.1.5 Making a Tree Node Interactive.....315
 - 10.1.6 Properties of Tree UI Elements.....318
- 10.2 Efficient Use of Actions to Enable and Disable UI Elements319
- 10.3 Programmatic Assignment of Actions to UI Element Objects 320
- 10.4 Accessing Existing UI Elements in a View Layout.....322
 - 10.4.1 Accessing UI Elements by Name.....323
 - 10.4.2 Accessing UI Elements Generically.....324
- 10.5 Introduction to Dynamic View Generation.....331
- 10.6 The Principles of Dynamic View Construction332
 - 10.6.1 Is There a Simpler Way of Doing It?.....333
 - 10.6.2 The Fundamental Principles of Dynamic View Layout Construction.....334
- 10.7 Dynamic Construction of a UI Element Hierarchy.....335
 - 10.7.1 Implementation Example.....336
 - 10.7.2 Implementation of the Method wdDoModifyViewO ... 337
 - 10.7.3 Implementation of the Method showNodeO.....340
 - 10.7.4 Implementation of the Method showNodeAsColumnsO.....343
 - 10.7.5 Implementation of the Method showNodeAsTableO ... 344
 - 10.7.6 Implementation of the Method getUIElementO.....346
- 10.8 Implementing an OVS.....348
 - 10.8.1 Basic Concept.....349
 - 10.8.2 Assigning the OVS Extension to a Context"Attribute 350
 - 10.8.3 OVS Processing.....352
 - 10.8.4 OVS Implementation.....353

11 Navigation and Error Handling 359

- 11.1 Navigation Processing..... 359
 - 11.1.1 The View Assembly..... 360
 - 11.1.2 Outbound Navigation Plugs..... 360
 - 11.1.3 Inbound Navigation Plugs..... 361
 - 11.1.4 Navigation Processing..... 361
 - 11.1.5 Re-Entrant Navigation Links..... 365
- 11.2 Suspend and Resume Plugs..... 368
 - 11.2.1 Declaring Suspend and Resume Plugs..... 369
 - 11.2.2 Suspend and Resume Plug Processing..... 372
 - 11.2.3 Requirements for an External Website..... 373
 - 11.2.4 Configuring Time-Out Values..... 374
- 11.3 Error Handling..... 376
 - 11.3.1 The Input and Output Halves of the Phase Model Processing..... 377
 - 11.3.2 Centralized Error Handling..... 378
 - 11.3.3 Error Handling Across a Component Hierarchy..... 382
 - 11.3.4 Principles for Handling Errors Trapped Before Navigation..... 384
 - 11.3.5 Principles for Handling Errors Trapped After Navigation..... 385

12 Web Dynpro Components and Polymorphism

- 12.1 Web Dynpro Components and Polymorphism..... 387
- 12.2 Pop-Up Windows..... 392
 - 12.2.1 Single Round-Trip Pop-Up Window..... 393
 - 12.2.2 Multiple Round-Trip Pop-Up Windows..... 397
 - 12.2.3 Opening an External Window..... 400
- 12.3 Application and URL Properties..... 403
 - 12.3.1 Application Properties..... 403
 - 12.3.2 URL Properties..... 404

- 13.1 A Brief History Lesson..... 407
 - 13.1.1 The Origins of RFC..... 407
 - 13.1.2 Java Connector and the Adaptive RFC Layer..... 408
 - 13.1.3 The Enterprise Connector..... 410
 - 13.1.4 The Common Model Interface..... 412

13.2	General Introduction to BAPIs.....	413
13.2.1	What Is a BAPI?.....	413
13.2.2	ABAP Function Module Interface.....	414
13.3	Custom-Written RFC Modules.....	417
13.4	Introduction to the Adaptive RFC Layer.....	418
13.5	Creating Adaptive RFC Models.....	421
13.6	Structuring a Model.....	425
13.7	Explanation of Generated Model Classes.....	426
13.7.1	Model Naming Convention.....	427
13.7.2	Executable and Non-Executable Model Objects.....	427
13.7.3	Model Structures.....	428
13.7.4	The Model Dictionary.....	429
13.7.5	Model Object Hierarchies and Relation Roles.....	431
13.8	Web Dynpro Models and Development Components.....	433
13.9	Using Model Objects in a Web Dynpro Controller.....	434
13.9.1	Model Objects and the Context.....	434
13.9.2	The Relationship Between the Contents of a Model Object and Its Corresponding Context Model Node	436
13.10	Altering a Model Object's Logical System After It Has Been Created.....	437
13.11	A Simple Example Using Context Model Nodes at Runtime. . .	439
13.11.1	Preparing the Context at Design Time.....	440
13.11.2	Component Controller Coding.....	441
13.11.3	View Controller Coding.....	450
13.11.4	Look at What Has Not Been Done.....	457
13.12	Adapting to Changes in an RFC Interface.....	459
14.1	Connecting to an SAP System.....	467
14.1.1	Logon Parameters.....	467
14.1.2	Language.....	467
14.1.3	Client.....	468
14.2	Identifying the Correct Backend System.....	469
14.2.1	Logical JCo Destinations.....	469
14.2.2	Prerequisites for Creating a Technical System.....	470
14.2.3	Creating a Technical System.....	471
14.2.4	Creating a JCo Destination.....	472
14.3	TheJCo Pool Concept.....	474
14.3.1	Pool Allocation.....	474
14.3.2	Pool Naming Conventions.....	475
14.3.3	Pool Definition.....	475

14.4	Pool Usage at Runtime	477
14.4.1	Correct Design of Web Dynpro Applications	477
14.4.2	Incorrect Design of Web Dynpro Applications	479
14.4.3	Correcting a Poor Model Architecture	480
14.5	Model Scope Type	482
14.6	Calculating JCO Pool Size	484
14.7	Metadata Connections	485
14.7.1	Pool Optimization	485
14.7.2	Metadata Connection Usage	486
14.8	Impact of JCo Connections on the SAP Gateway Process	486
14.9	Relationship Between JCO Destinations and ABAP Sessions	489
14.9.1	Logging on to an SAP System	489
14.9.2	ABAP Sessions	489
14.9.3	Performing ABAP Database Updates Without Causing an Unwanted Session Change	491
14.9.4	Statements to Avoid in an RFC Module	492
14.10	Avoiding the Read-Write-Read Problem	492
	Web Dynpro Naming Placeholders	499
A.1	Web Dynpro Component Architecture	499
A.2	Web Dynpro Controllers	499
A.3	Model Objects	500
A.4	NetWeaver Development Infrastructure Entities	500
A.5	Context Entities	500
A.6	Generic and Composite Abbreviations	501
A.7	Subscripts for Composite Placeholders Using the SAP Recommended Suffixes	502
A.8	AS Java Placeholders	503
	Naming Conventions	505
B.1	General Rules for Naming	505
B.2	Naming Conventions for Coding Entities	505
B.2.1	Applications: $\{n_a\} = \{a\}App$	505
B.2.2	Components: $\{n_c\} = \{c\}Comp$	506
B.2.3	Component Interface Views: $\{n_{civ}\} =$ $\{w\}InterfaceView$	506
B.2.4	Component Usage: $\{n_u\} = \{n_c\}\{p\}Inst$ or $\{cid\}\{p\}Inst$	506
B.2.5	Custom Controllers: $\{n_{cc}\} = \{cc\}Cust$	506
B.2.6	Inbound Plugs: $\{n_{pi}\} = \{p_{in}\}In$	506
B.2.7	Models: $\{n_m\} = \{m\}Model$	507

B.2.8	Outbound Plugs: $\{n_{po}\} = \{p_{out}\}Out$	507
B.2.9	Component Interface Definition: $\{n_{cid}\} =$ $\{cid\}Compl$	507
B.2.10	Component Interface View Definition: $\{n_{civ}\} =$ $\{n_{cid}\}<any_name>$	507
B.2.11	Views: $\{n_v\} = \{v\}View$	507
B.2.12	Viewsets: $S\{n_{vs}\} = \{vs\}Viewset$	508
B.2.13	Windows: $\{w\} = \{n_c\}$ or $\{n_c\}Window$ or $<any_name>Window$	508
C	Exercise tp^Display a File System Using a Recursive Context Node Structure.....	509
C.1	Create a New Web Dynpro DC.....	509
C.2	Create a Web Dynpro Component.....	510
C.3	Create a Java Bean to Represent a Directory Entry.....	510
C.4	Import the Java Bean as a Model.....	512
• C.5	Component Controller Configuration.....	513
C.5.1	Declare Model Usage.....	513
C.5.2	Context Configuration.....	513
C.5.3	Method Declarations and Coding.....	514
C.6	View Controller Configuration.....	517
C.6.1	Context Configuration.....	517
C.6.2	Create an Action.....	517
C.6.3	Editing the View Layout.....	518
C.6.4	View Controller Implementation.....	518
C.7	Create an Application.....	519
C.8	Comment on the Use of Standard SAP Icons.....	520
D	ABAP Coding.....	521
E	Dictionary Structures.....	525
F	The Author.....	527
	Index.....	529