

Praxisleitfaden SAP XI – Programmierung

Marcus Banner, Halil-Cem Gürsoy, Heinzpeter Klein

Inhalt

Einleitung	3	Aktivieren der ABAP-Mapping-Funktion in SAP XI	22
Inhalt des Heftes	4	Anlegen einer ABAP-Mapping-Klasse	24
Danksagung	4	ABAP-Mapping mit der iXML-Library	25
1 Architektur der SAP NetWeaver Exchange Infrastructure	5	Einbindung des ABAP-Mappings ins XI-Repository	25
1.1 SAP NetWeaver	5	Customizing des XI-Directory	29
1.2 Process Integration mit SAP XI	6	3.2 Java-Mapping	34
1.3 Architektur von SAP XI	6	Implementierung der Java-Mapping-Klasse	34
1.4 XI-Landschaftstopologie	7	Testen der Mapping-Klasse	38
2 Grundlagen der SAP NetWeaver Development Infrastructure	9	3.3 Grafischer Mapping-Editor	39
2.1 Architektur der SAP NetWeaver Development Infrastructure	10	4 Entwicklung eines Adapters	45
Konzeption der NWDI	10	4.1 Theoretische Grundlagen	46
Bestandteile der NWDI	11	XI-Adapter-Framework	46
Komponentenmodell	12	J2EE Connector Architecture	46
Ablauf der Entwicklung im SAP NetWeaver Developer Studio	13	Besonderheiten bei SAP XI	47
2.2 Tipps zur Installation des SAP NetWeaver Developer Studio	14	4.2 Vorbereiten des Partner Connectivity Kit	48
2.3 Umgebung zur Adapterentwicklung	15	Benutzerrechte	48
Eclipse	15	aai.properties	50
Subversion	16	Aktivierung des CPA-Cache	50
Subclipse	17	Anpassungen im XI-Adapter	50
Apache Ant	17	4.3 Entwicklungsumgebung vorbereiten	50
Java Development Kit	18	XI-Bibliotheken einbinden	50
3 Mapping-Programmierung in SAP XI	19	Muse-Bibliotheken in Eclipse einbinden	51
3.1 ABAP-Mapping	19	Jabber-Server und -Client	52
Abzubildendes Szenario	19	Apache Ant	53
		Ant-Task zur SDA-Erzeugung	54
		Remote-Deployment mithilfe der SDM-API	54

4.4	Deployment der Muse-Bibliotheken in den SAP NetWeaver AS	54	[Paketname]-dd.xml	81
4.5	Anlegen eines neuen Eclipse-Projekts	57	log-configuration.xml (SDA)	82
4.6	Projekt versionieren	60	CPA-Cache-Metadatei	82
4.7	Adapterskelett der SAP	60	4.12 Zusammenbau (Build) und Deployment des Adapters	83
4.8	Refactoring der Quelltexte und Deskriptoren	62	4.13 Upload der CPA-Cache-Metadaten	85
4.9	Implementierung der Jabber-Kommunikation	63	4.14 Testen im Partner Connectivity Kit	85
4.10	Integration der Kommunikation in den Adapter	71	4.15 Adaptermodul	89
	Implementierung von javax.resource.cci.ConnectionFactory	72	Implementierung	89
	ConnectionManager	73	Testen des Adaptermoduls	92
	ManagedConnectionFactory	73	Anwendungsmöglichkeiten eines Moduls	93
	ManagedConnection	75	5 SLD-API	95
	Verbindungsobjekt	76	5.1 Konfiguration des Java-IDE	95
	Interaktion	77	5.2 Implementierung des CIM-Clients	97
	Weitere Anpassungen an den Quelldateien	77	5.3 Testen des CIM-Clients	98
4.11	Deployment-Deskriptoren	78	6 XI-Anwendungen debuggen	101
	connector-j2ee-engine.xml	78	6.1 Debuggen von ABAP-Komponenten	101
	log-configuration.xml (RAR)	79	6.2 Debuggen von Java-Komponenten	102
	ra.xml	79	7 Abkürzungsverzeichnis	105
	application.xml	80	Index	107
	application-j2ee-engine.xml	80		

Einleitung

Dieses Heft ergänzt und erweitert die Themen, die im *Praxisleitfaden SAP XI 3.0 – Administration*¹ wegen des anders gelagerten Schwerpunkts nicht ausführlich beschrieben wurden. Es kann sowohl als eigenständiger Praxisleitfaden zur Programmierung als auch als ideale Ergänzung zu jenem Heft verstanden werden. Wir legen diesmal den Schwerpunkt auf das Thema Entwicklung. Wie im Praxisleitfaden zur Administration bewegen wir uns wieder sehr nah an einem Praxisbeispiel, das Sie mit relativ wenig Aufwand mit einer Minimalinstallation selbst nachvollziehen können

Diese Minimalinstallation für einen Entwickler besteht aus dem SAP NetWeaver Developer Studio (NWDS), basierend auf der Entwicklungsumgebung Eclipse, dem SAP Partner Connectivity Kit (PCK) für die Adapterentwicklung, Subversion und Subclipse für die Versionsverwaltung² und Apache Ant als unentbehrlichem Helfer im Hintergrund. Zu jedem der genannten Entwicklungswerkzeuge finden Sie im Heft eine ausführliche und verständliche Erklärung der Funktionsweise sowie die Webadresse, unter der Sie die Werkzeuge aus dem Internet kostenlos herunterladen können.

Im September 2005 erschien der *Praxisleitfaden SAP XI 3.0 – Administration*. Damals erwähnten wir im Vorwort, dass sich SAP XI 3.0 im Vergleich zur Version 2.0 deutlich gesteigert und mit steigendem Support-Package-Level (SP-Level) auch das Potenzial hätte, sich zu einem starken Konkurrenten bestehender Anbieter von Enterprise Application Integration (EAI) zu entwickeln. Nach einem Jahr mit guten und teilweise weniger guten SP-Updates hat sich unsere Vermutung bestätigt.

SAP XI 3.0 mit SP 16 ist mittlerweile eindeutig ein ernst zu nehmendes Konkurrenzprodukt zu bekannten EAI-Produktpaletten wie SeeBeyond, das vor Kurzem von Sun übernommen wurde, oder IBM WebSphere.

Dennoch setzt sich SAP XI bei größeren Firmen, die sich vor dem Erscheinen von SAP XI 3.0 bereits für ein EAI-Tool entschieden hatten, teilweise nur sehr langsam durch. Hier spielen Bestands- und Investitionsschutz eine Rolle. Bei Firmen, die sonst sehr viel mit SAP-Produkten arbeiten, stellen solche Überlegungen oft die größten Hindernisse dar. Aus technischer Sicht gibt es bei diesen Firmen mittlerweile meist keine sinnvollen Gründe mehr, die gegen einen Einsatz von SAP XI 3.0 und für den Einsatz einer Drittsoftware im EAI-Umfeld sprechen würden.

In diesem Sinne wollen wir Ihnen in diesem Praxisleitfaden einige wertvolle Tipps und Tricks für die Entwicklung innerhalb von SAP XI mit auf den Weg geben.

Einer der Schwerpunkte dieses Heftes liegt auf der Entwicklung eigener XI-Adapter. Dieses Thema wollten wir aufgrund der sehr geringen und recht ungenügenden Dokumentation und Unterlagen, die es hierzu offiziell und inoffiziell gibt, unbedingt in dieses Heft integrieren. Wir wissen aus eigener Erfahrung, dass viele SAP-Berater, die sich mit SAP XI beschäftigen und dort auch entwickeln wollen, aus dem ABAP-Umfeld kommen und daher oft Probleme mit der Java-Welt und ihren Entwicklungsumgebungen und Tools haben. Wir haben versucht, dieser Problematik gerecht zu werden und die Beispiele so zu formulieren, dass jemand mit Java-Grundkenntnissen sie nachvollziehen kann. Wir bitten die geübteren Java-Entwickler unter unseren Lesern, die teilweise recht ausführlich beschriebenen Vorgänge in den Entwicklungsumgebungen und den teilweise absichtlich recht einfach gehaltenen Java-Code zu überlesen.

¹ Banner, Marcus; Klein, Heinzpeter: *Praxisleitfaden SAP XI 3.0 – Administration*. SAP-Heft 21. Galileo Press, Bonn 2005.

² Analog können Sie auch CVS (Concurrent Versions System) oder SAP DTR (Design Time Repository) verwenden.

Wir hoffen, dass wir die Entwicklung einiger neuer XI-Adapter anstoßen und einige Entwickler demnächst im Web begrüßen dürfen! Den Quellcode für den Beispieladapter stellen wir Ihnen auf der Webseite zu diesem Heft unter <http://www.sap-hefte.de/1202> zum Download zur Verfügung.

Inhalt des Heftes

Das vorliegende Heft beschreibt die Grundlagen für den Aufbau und Ablauf der SAP-Adapterentwicklung mit Java sowie verschiedene Mapping- und Debugging-Möglichkeiten.

Zu Beginn gehen wir in **Kapitel 1** kurz und prägnant auf die SAP NetWeaver Exchange Infrastructure, und in **Kapitel 2** auf Architektur und Infrastruktur der SAP NetWeaver Development Infrastructure (NWDI) ein. Im weiteren Verlauf von Kapitel 2 folgt die Beschreibung der SAP-Client-Installation für das SAP NetWeaver Developer Studio (NWDS). Die Erklärung zu Change Management Services (CMS) und Design Time Repository (DTR) fällt hier relativ kurz aus, da statt ihrer andere Werkzeuge für die Entwicklung eingesetzt wurden. Diese können konform zum SAP-Standard verwendet werden. Trotzdem wollen wir Ihnen die integrierte SAP-Lösung mit CMS und DTR nicht ganz vorenthalten. Es folgt die detaillierte Beschreibung der Toolkette für die Adapterentwicklungsumgebung (Eclipse, Subclipse, Ant, JDK).

Nach der Beschreibung dieser Grundvoraussetzungen folgt ab **Kapitel 3** die Beschreibung des XI-Mappings für ABAP und für Java. Ausführlich und mit einfachen Beispielen aus der Praxis werden diese beiden Möglichkeiten erklärt.

In **Kapitel 4** beschreiben wir, wie ein Adapter mit dem PCK entwickelt wird. Dort schildern wir praxisnah die häufigsten Schwierigkeiten bei der Entwicklung, bieten Lösungsmöglichkeiten an und steigen tief in die Funktionsweise ein. Schon nach kurzer Einarbeitungszeit ist es so geübteren ABAP- und Java-Entwicklern möglich, Kundenanforderungen in eigenen Anwendungen umzusetzen.

Weitere Spezialthemen wie SLD-API (Einbinden von Java-Klassen in die XI-Umgebung) und Debugging unter SAP XI werden ausführlich in **Kapitel 5** und **Kapitel 6** beschrieben.

Danksagung

Marcus Banner möchte sich an dieser Stelle bei seiner Familie für ihr Verständnis bei der Erstellung dieses Werks bedanken: Ihr beide seid mein Ein und Alles!

Heinzpeter Klein möchte sich bei seinen Freunden und seiner Familie für die Toleranz und das Verständnis für sein Fortbleiben bedanken – sie haben mich lange nicht mehr gesehen.

Dr. Halil-Cem Gürsoy bedankt sich bei seiner Familie für die sehr große Geduld und Unterstützung sowie bei der CDI AG und ihren Mitarbeitern für die Unterstützung durch Infrastruktur und Denkanstöße.

Nun möchten wir Ihnen viel Spaß beim Lesen wünschen, verbunden mit der Hoffnung, dass Ihnen dank dieses Heftes viel mühsames Ausprobieren zu den dargestellten Themen erspart bleibt!

bieten, außerhalb eines Application Servers zum Einsatz zu kommen. Da ein XI-Adapter aber immer in einem *Managed Environment*, dem SAP NetWeaver Application Server deployt wird, ist diese Implementierung nicht zwingend erforderlich. Sie kann aber dennoch durchgeführt werden.

Die Anforderungen bezüglich des *Transaktionsmanagements* sind ebenfalls etwas eingeschränkt: Von den drei vorgesehenen Transaktionstypen NoTransaction, LocalTransaction und XATransaction wird letztere von SAP XI nicht unterstützt. Der Adapter muss also entweder NoTransaction oder LocalTransaction unterstützen. Falls LocalTransaction eingesetzt wird, müssen folgende Randbedingungen beachtet werden: Die Methoden `ManagedConnection.getLocalTransaction()` und `Connection.getLocalTransaction()` müssen implementiert werden, und ebenso die Schnittstellen `javax.resource.spi.LocalTransaction` und `javax.resource.spi.ConnectionEventListener`.

Auch bezüglich des Deployments, also des »Transports« des Adapters auf das XI-System, müssen einige Abweichungen gegenüber der JCA-Spezifikation beachtet werden: Während das Deployment eines RAR (*Resource Adapter Archive*) unterstützt wird, ist das für ein EAR (*Enterprise Application Archive*) im Gegensatz zur JCA-Spezifikation nicht möglich. Das liegt daran, dass die JCA-Spezifikation in diesem Fall eine Bündelung mit einer Applikation (die in der EAR ebenfalls enthalten ist) vorsieht, wobei dann der Adapter nur für eben diese Applikation sichtbar sein darf (siehe JCA 1.0, Abschnitt 10.2). Innerhalb von SAP XI deployen wir aber keine Applikationen bzw. die Applikation, die auf den Adapter zugreift, ist das AF. Das würde bedeuten, dass in solch einem Fall der Adapter gebündelt mit dem AF in einer EAR deployt werden müsste. Stattdessen können Sie das SAP-spezifische Format SDA (*Software Delivery Archive*) verwenden, das neben der RAR zusätzliche Informationen über den Adapter enthält. Dieses SDA kann dann wie gewohnt über den Software Deployment Manager (SDM) deployt werden. Die SAP empfiehlt diesen Weg und auch wir werden in unserem Beispiel ein SDA verwenden.

4.2 Vorbereiten des Partner Connectivity Kit

Nach all der Theorie wollen wir nun endlich selbst Hand anlegen und einen eigenen Adapter entwickeln. Dabei werden wir das Rad nicht neu erfinden, sondern verwenden – wie von SAP empfohlen – den Beispieladapter, der bei jeder Installation des Partner Connectivity Kit (PCK) im Lieferumfang enthalten ist, als Ausgangspunkt und modifizieren diesen nach unseren Wünschen.

Die Entwicklung werden wir mithilfe des PCK durchführen. Das PCK dient einerseits der Anbindung von Systemlandschaften an XI, ohne auf diesen ebenfalls ein vollständiges SAP XI installieren zu müssen (»Partnerszenario«) sowie als Entwicklungsumgebung für Adapter.

Wir gehen daher davon aus, dass Sie Zugriff auf ein SAP NetWeaver-System mit einem PCK besitzen (wir verwenden die Version 3.0 SP 14).

Benutzerrechte

In einem ersten Schritt müssen die Benutzerrollen für die Adapterentwicklung im J2EE-Server eingerichtet bzw. konfiguriert werden. Hierzu öffnen Sie den J2EE Engine Visual Administrator. Sie finden diesen auf Windows-Systemen unter `<j2eeDir>\admin`, wobei `<j2eeDir>` das Verzeichnis des J2EE-Containers ist und z. B. `C:\usr\sap\J2E\JCO0\j2ee` lauten kann.

Sobald Sie die Batch-Datei `go.bat` ausführen, startet der Visual Administrator, und Sie werden mit einer Login-Aufforderung begrüßt (siehe Abbildung 4.3).

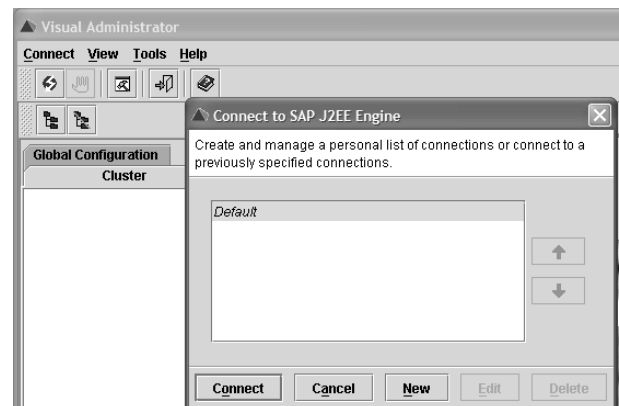


Abbildung 4.3 Login-Maske des J2EE Engine Visual Administrator

Nachdem Sie sich mit dem System verbunden haben, können Sie sich sofort in die Verwaltung für Benutzerrechte begeben, um dem Benutzer, unter dem die Adapterentwicklung erfolgen soll, die benötigten Rechte zu geben. Hierzu wählen Sie im linken Navigationsmenü **Services • Security Provider • Policy Configurations**.

Hiernach können Sie durch die Auswahl der entsprechenden Komponente im mittleren Bereich und der Auswahl des Reiters **Security Roles** Ihrem Benutzer die entsprechenden Rechte zuweisen (siehe Abbildung 4.4). Tabelle 4.1 zeigt die zu vergebenden Rechte.

Komponente	Rolle
sap.com/com.sap.af.app*AdapterFramework	<i>Xi_af_adapter_monitor</i>
sap.com/com.sap.af.app*CPACache	<i>Xi_af_cpa_invalidate</i>
sap.com/com.sap.af.app*CPACache	<i>Xi_af_cpa_monitor</i>
sap.com/com.sap.af.app*CPACache	<i>Xi_af_cpa_schemaupload</i>
sap.com/com.sap.af.ms.app*MessagingSystem	<i>Xi_af_receive</i>
sap.com/com.sap.af.soapadapter*XISOAdapter	<i>Xi_adapter_soap_message</i>

Komponente	Rolle
sap.com/com.sap.sap.xi.mdt*mdt	<i>Display</i>
sap.com/com.sap.sap.xi.mdt*mdt	<i>Modify</i>
sap.com/com.sap.sap.xi.mdt*mdt	<i>Payload</i>
sap.com/com.sap.sap.xi.pck*aii_ib_sbeans.jar	<i>administer</i>
sap.com/com.sap.sap.xi.pck*pck	<i>singlesignon</i>
sap.com/com.sap.sap.xi.pck*pck	<i>Support</i>

Tabelle 4.1 Rollen, die dem PCK-Benutzer zugeordnet werden müssen

Falls mehrere Personen an der Entwicklung beteiligt werden sollen, empfiehlt es sich, vorher eine Benutzergruppe, z. B. *SAP_XI_PCK_ADMIN* anzulegen und die Benutzer dieser Gruppe zuzuordnen. Hiernach reicht es völlig aus, der Gruppe die Rechte zu geben.

Schließen Sie die Applikation noch nicht: Im weiteren Verlauf werden Sie die Administrationsoberfläche nochmals benötigen.

Weitere Details bezüglich der benötigten Rechte können Sie im Übrigen den SAP-Hinweisen 746328 und 792456 entnehmen.

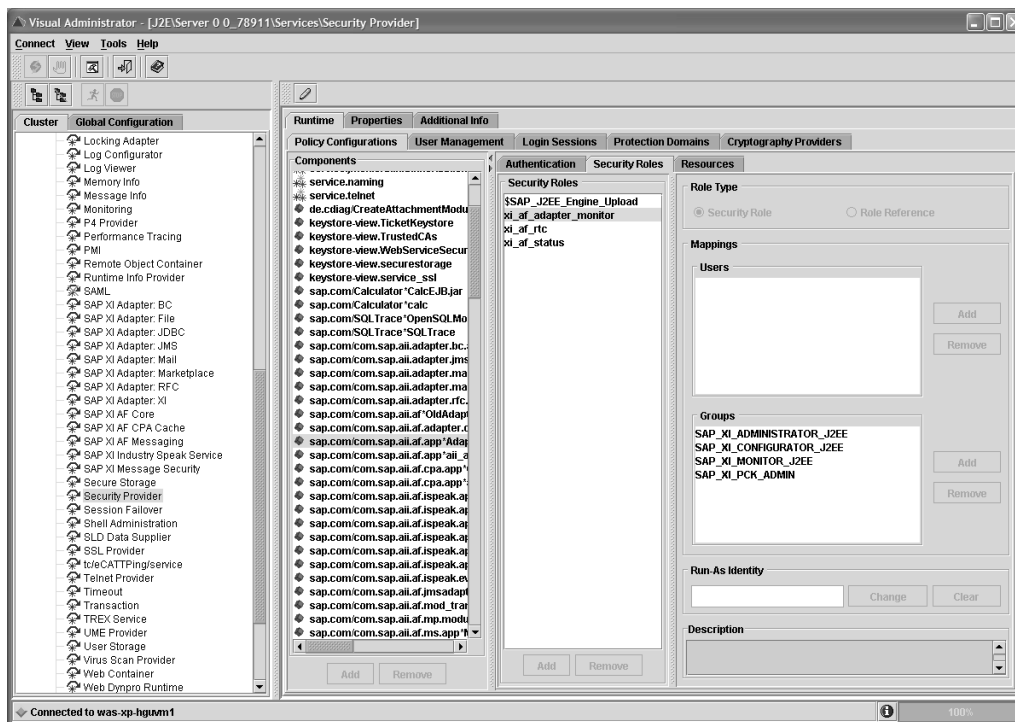


Abbildung 4.4 Konfiguration der Rollen für den PCK-Benutzer mithilfe des J2EE Engine Visual Administrator

```

com.sap.aii.ib.client.properties = com.sap.aii.ib.client.*,com.sap.aii.connect.*,com.sap.aii.
ib.server.*,com.sap.aii.docu.*,com.sap.aii.pck.*
### Connections
com.sap.aii.connect.directory.contextroot = pck
com.sap.aii.connect.directory.name      = was-xp-hguvm1
com.sap.aii.connect.directory.httpport  = 50000
com.sap.aii.connect.directory.rmiport   = 50004
### Appl name
com.sap.aii.ib.client.applicationname.directory = sap.com/com.sap.xi.pck/
com.sap.aii.ib.client.login.languages = EN,DE
### No locking for PCK
com.sap.aii.ib.server.lockauth.activation = false
com.sap.aii.pck.server.compiler.classpath_resolverServiceName = classpath_resolver
com.sap.aii.docu.url = http://was-xp-hguvm1:50000/pck/docOnline/DOCU
com.sap.aii.docu.languages = EN,DE

```

Listing 4.1 Beispiel für die Datei `aii.properties`

`aii.properties`

Jetzt müssen Sie die Datei `aii.properties` erstellen und in das Verzeichnis `<serverDir>` kopieren. Bei `<serverDir>` kann es sich z. B. um das Verzeichnis `/usr/sap/J2E/JC00/j2ee/cluster/server0` handeln. Listing 4.1 zeigt ein Beispiel. Beachten Sie, dass Sie den Rechnernamen und die Ports anpassen müssen.

Aktivierung des CPA-Cache

Nun müssen Sie den CPA-Cache aktivieren. Auch dies geschieht mithilfe des J2EE Engine Visual Administrator. Hierfür wechseln Sie im Administrator in den Service **SAP XI AF CPA Cache** und passen die Parameter wie in Tabelle 4.2 an.

Parameter	Wert
cacheType	<i>PCK</i>
SLDAccess	<i>False</i>

Tabelle 4.2 CPA-Cache aktivieren

Abbildung 4.5 zeigt die Änderungen der Parameter für den CPA-Cache. Die Änderungen gegenüber den Standardeinstellungen werden kursiv dargestellt

Anpassungen im XI-Adapter

Einige Parameter in der Konfiguration des XI-Adapters müssen auch noch angepasst werden. Gehen Sie wie zuvor in den J2EE Engine Visual Administrator und wählen Sie den Service **SAP XI Adapter XI** aus. Dort passen Sie die Parameter wie in Tabelle 4.3 gezeigt an.

Parameter	Wert
<code>xiadapter.internal</code>	<i>devModePCK</i>
<code>xiadapter.isconfig.url</code>	<i>http://[Server name]:[J2EE Engine Port]/MessagingSystem/receive/AFW/XI</i>
<code>xiadapter.isconfig.user</code>	<i>[Name des Benutzers]</i>
<code>xiadapter.isconfig.password</code>	<i>[Kennwort des Benutzers]</i>

Tabelle 4.3 Anpassung der XI-Adapter-Parameter

4.3 Entwicklungsumgebung vorbereiten

Die von uns vorgeschlagenen Komponenten für die Entwicklung eines Adapters haben wir bereits im Abschnitt 2.3 näher beschrieben. Wir gehen davon aus, dass Sie ein *Java Development Kit* sowie *Eclipse* auf Ihrem Entwicklungssystem installiert haben. Zur Sicherheit, z. B. um auf einen älteren Versionsstand zurückgreifen zu können, sollten Sie auch ein Versionierungssystem verwenden.

XI-Bibliotheken einbinden

Folgende *Java-Bibliotheken* (JAR-Dateien) benötigen Sie für die Entwicklung eines Adapters. Je nach Bedarf können weitere hinzukommen:

- ▶ `<serverDir>/bin/ext/com.sap.aii.af.lib/aii_af_trace.jar`
- ▶ `<serverDir>/bin/ext/com.sap.aii.af.lib/aii_af_cci.jar`
- ▶ `<serverDir>/bin/ext/com.sap.aii.af.lib/aii_af_mp.jar`
- ▶ `<serverDir>/bin/ext/com.sap.aii.af.lib/aii_af_ms_api.jar`
- ▶ `<serverDir>/bin/ext/com.sap.aii.af.lib/aii_af_ms_spi.jar`

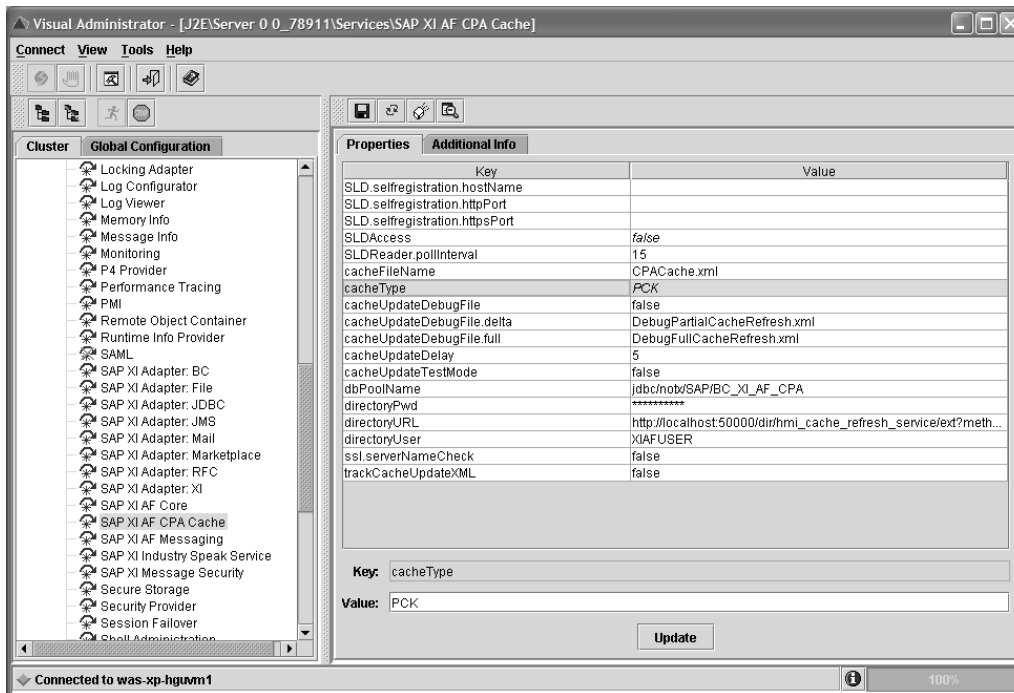


Abbildung 4.5 Änderungen der Parameter für den CPA-Cache

- ▶ <serverDir>/bin/ext/com.sap.aii.util.misc/aii_util_misc.jar
- ▶ <serverDir>/bin/ext/com.sap.xi.util.misc/aii_utilxi_misc.jar
- ▶ <serverDir>/bin/ext/j2eeca/connector.jar
- ▶ <serverDir>/bin/ext/com.sap.guid/guidgenerator.jar
- ▶ <serverDir>/bin/services/com.sap.aii.adapter.xi.svc/aii_adapter_xi_svc.jar
- ▶ <serverDir>/bin/services/com.sap.aii.af.cpa.svc/aii_af_cpa.jar
- ▶ <serverDir>/bin/services/com.sap.aii.af.svc/aii_af_svc.jar
- ▶ alle J2EE-Client-Bibliotheken unter <j2eeDir>/j2eeclient

Diese Bibliotheken müssen Sie später Ihrem Adapterprojekt in Eclipse mitteilen. Um sich das Leben etwas zu erleichtern, empfiehlt es sich, unter Eclipse die Bibliotheken thematisch zu bündeln, um diese später mit wenigen Mausklicks komplett einem Projekt zuordnen zu können. Hierzu öffnen Sie in Eclipse das Menü **Windows • Preferences** und wählen im linken Navigationsfenster den Punkt **Java • Build Path • User Libraries** aus (siehe Abbil-

dung 4.6). Mit einem Mausklick auf **New** erhalten Sie einen Dialog, in dem Sie den Namen der Benutzerbibliothek eingeben können, in unserem Fall z. B. *SAPXIAF* für die XI-AF-JAR-Dateien sowie *SAPJ2EECLIENT* für die J2EE-Client-Bibliotheken. Haben Sie diese vorerst leeren User Libraries angelegt, können Sie sie mit der Maus auswählen und dann über den Button **Add JARs** JAR-Dateien hinzufügen. Fügen Sie die oben aufgezählten Dateien hinzu. Diese User Libraries können nun jedem Projekt in Ihrer Entwicklungsumgebung bekannt gegeben werden; ansonsten müssten Sie ein ähnliches Verfahren für jedes Adapterprojekt durchlaufen.

Muse-Bibliotheken in Eclipse einbinden

Um Nachrichten mit dem Adapter an einen Jabber-Server versenden zu können, benötigen wir eine Bibliothek, die das Jabber/XMPP-Protokoll implementiert. Wir empfehlen Ihnen die *Muse-Bibliothek*, die recht bekannt und stabil ist (<http://open.echomine.org/confluence/display/MUSE/Muse+Home>).

Laden Sie hierzu von der Muse-Homepage die aktuelle Version herunter; für dieses Heft haben wir die Version

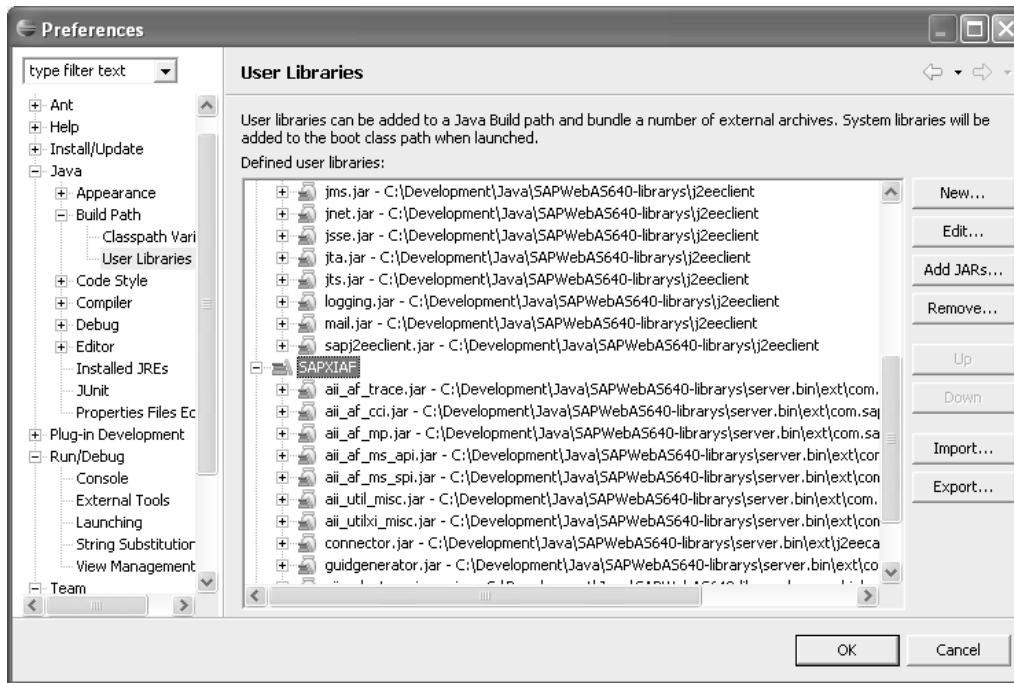


Abbildung 4.6 JAR-Dateien zu User Libraries bündeln

0.81 verwendet. Entpacken Sie die heruntergeladene Zip-Datei in einen dafür vorgesehen Ordner.

Hiernach binden Sie wie oben beschrieben die benötigte Datei *muse.jar* in eine User Library in Eclipse ein.

Jabber-Server und -Client

Neben den Muse-Bibliotheken benötigen wir natürlich noch einen Jabber-Server und einen Jabber-Client. Hier gibt es eine sehr große Auswahl sowohl an kommerziellen als auch an freien Produkten. Eine umfassende Übersicht finden Sie unter <http://www.jabber.org>. Wir empfehlen als Server *Wildfire* von Jive Software, das unter der GNU Public License (siehe <http://www.gnu.org/copyleft/gpl.html>) steht. Sie können Wildfire von der Jive-Homepage herunterladen (<http://www.jivesoftware.org/wildfire/>). Für unser Szenario haben wir die Version 2.6.1 verwendet. Installieren Sie Wildfire, indem Sie der einfachen Beschreibung in der Dokumentation folgen. Nach dem ersten Start von Wildfire sollten Sie sich in die Administrationsoberfläche begeben (standardmäßig unter <http://127.0.0.1:9090/> zu finden) und zwei Benutzer einrichten. Wählen Sie in der Administrationsoberfläche in der oberen Navigation **Users/Groups** und anschließend links **Create New User**. Diese Benutzer werden wir

verwenden, um Nachrichten aus dem Adapter heraus zu versenden beziehungsweise zu empfangen.

Damit Nachrichten an einen Benutzer, der gerade offline ist, nicht verloren gehen, empfiehlt es sich, diese vom Server speichern zu lassen, damit sie dem Empfänger zugestellt werden können, wenn er sich wieder anmeldet (so genannte *Offline-Nachrichten*). Das erreichen Sie, indem Sie unter **Server • Server Settings • Offline Messages** die Optionen **Store** und **Always Store** auswählen.

Als Jabber-Client empfiehlt sich Ψ (sprich: »Psi«), zu finden unter <http://psi-im.org/>. Ψ ist übrigens nicht nur für Windows, sondern auch für sehr viele weitere Plattformen erhältlich. Nach dem Download ist die Installation recht simpel: Starten Sie einfach den Installer und folgen Sie den vorgegebenen Schritten.

Nach dem ersten Start von Ψ müssen Sie Ihre Benutzer einrichten. Hierzu wird zuerst ein *Profile* eingerichtet, das verschiedene Benutzer aufnehmen kann. Nach der Erstellung des Profiles richten Sie die *Jabber Accounts* ein. Verwenden Sie dafür die zwei Benutzer, die Sie vorhin im Wildfire-Server eingerichtet haben. Beachten Sie, dass Sie dem Benutzernamen ein »@[ServerName]« anhängen müssen, damit der Client weiß, auf welchem Server er die Benutzer findet. Abschließend müssen diese

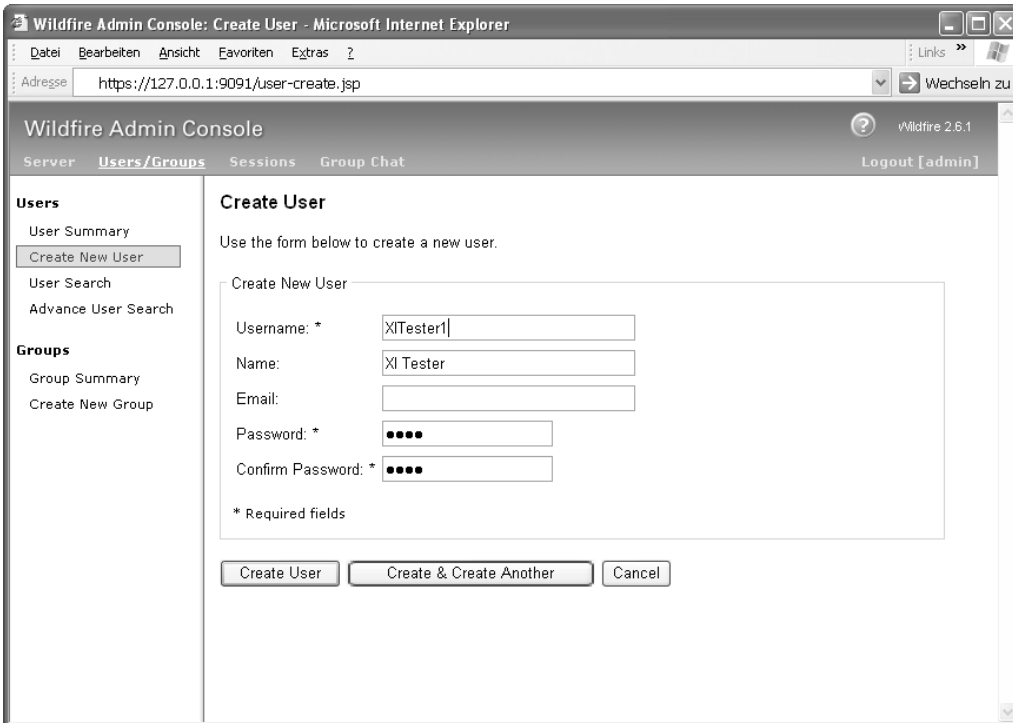


Abbildung 4.7 Neue Benutzer über die Administrationsoberfläche von Wildfire einrichten

Benutzer sich gegenseitig als Kontakt bekannt gegeben werden. Hierbei werden auch Autorisierungsnachrichten ausgetauscht, der Vorgang ist allerdings intuitiv zu bedienen.

Ist dies geschehen, könnte Ihr Ψ wie in Abbildung 4.8 aussehen. Der Jabber-Client Ψ ist in Aktion mit zwei eingerichteten Accounts, und die jeweiligen Kontakte sind auch online.



Abbildung 4.8 Jabber-Client Ψ in Aktion mit zwei eingerichteten Accounts

Alternativ zu einem eigenen Jabber-Server können Sie auch öffentlich zugängliche Server verwenden (eine Liste finden Sie unter <http://www.xmpp.net/>) und sich dort Benutzer einrichten. Allerdings setzt dies für unser Szenario voraus, dass dann unter Umständen Firewalls in Ihrer Domäne konfiguriert werden müssen, um eine Kommunikation zu erlauben. Sie können aber mit Ψ auch einen http-Proxy verwenden. Weiterhin können Sie anstelle von Ψ jeden beliebigen Jabber-Client einsetzen. Eine Übersicht über Clients finden Sie ebenfalls unter <http://www.jabber.org>.

Apache Ant

Das Kompilieren der Quelltexte, Zusammenbauen der Archiv-Dateien und das Deployment wollen wir mithilfe des frei erhältlichen Tools *Apache Ant* durchführen (siehe Kapitel 3).

Hierzu laden Sie das jüngste stabile Ant-Release von der Ant-Homepage (<http://ant.apache.org>) unter **Downloads • Binary Distributions** herunter. Zurzeit ist dies die Version 1.6.5, aber unser Projekt ist mit jeder Ant-1.6-Version kompatibel. Haben Sie das Zip-Archiv in einen Ord-

ner entpackt, passen Sie anschließend unter Windows XP oder 2000 einige Systemumgebungsvariablen an.

Um eine Umgebungsvariable unter Windows 2000 oder XP anzupassen, gehen Sie in die Systemsteuerung, wählen **System** und anschließend den Reiter **Erweitert** und dort den Button **Umgebungsvariablen** aus. Mit einem Klick auf **Neu** unter **Systemvariablen** erstellen Sie zuerst die Umgebungsvariable `ANT_HOME` (siehe Abbildung 4.9).

Nun gehen Sie analog vor und passen die Systemumgebungsvariable `Path` so an, dass das `bin`-Verzeichnis von Ant ebenfalls in den Suchpfad aufgenommen wird. Dabei können Sie die zuvor gesetzte Umgebungsvariable `ANT_HOME` nutzen: Setzen Sie einfach an das Ende der bisherigen Definition `»%ANT_HOME%\bin«` ein.

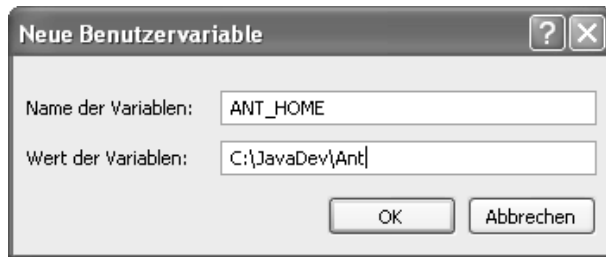


Abbildung 4.9 Erstellen einer Systemumgebungsvariablen unter Windows XP und 2000

Jetzt sollten Sie Ant starten können. Überprüfen Sie es, indem Sie ein Windows-Kommandofenster öffnen (**Start** • **Ausführen**, dann `»cmd«` eingeben) und `»ant -version«` eingeben. Die Ausgabe dürfte dann wie folgt aussehen:

```
C:\>ant -version
Apache Ant version 1.6.5 compiled on June 2
2005
C:\>
```

Ant-Task zur SDA-Erzeugung

Um SDAs ohne das NetWeaver Developer Studio oder das J2EE-Engine-Deploy-Tool zu erstellen benötigen wir einen Ant-Task, der diese Aufgabe für uns erledigen kann. Eine Lösung ist schon im Developer Studio in Form des Tasks `JarSAP` vorhanden.

Die hierfür benötigte JAR-Datei `jar4sap.jar` finden Sie in der Developer-Studio-Installation im Plug-in `com.sap.ide.eclipse.jarsap`, üblicherweise zu finden unter `C:\Pro-`

`gramme\SAP\JDT\eclipse\plugins\com.sap.ide.eclipse.jar-sap\lib`.

Sie werden später den Ort dieser Datei in das Build-Skript für Ant einsetzen müssen.

Remote-Deployment mithilfe der SDM-API

Da wir die benötigten Archive gerne ohne Verzögerungen nach dem Kompilieren und Zusammenbauen auf dem J2EE-Server deployen möchten, wäre ein Deployment über die SDM-Oberfläche recht umständlich. Da wir sowieso schon Ant einsetzen, bietet sich auch hier eine elegante Lösung an: David Beisert, ein bekannter Consultant im NetWeaver-Umfeld, hat in seinem SDN-Blog einen Ant-Task vorgestellt, mit dessen Hilfe wir ein Deployment aus Ant heraus mithilfe der SDM-API vornehmen können (<https://weblogs.sdn.sap.com/pub/wlg/3138>).

Folgen Sie dem Link **Download SDM Deployment Task** und laden Sie die Zip-Datei herunter. Danach benennen Sie die heruntergeladene Datei `sdmAnt.zip` in `sdmAnt.jar` um.

Noch ein wichtiger Hinweis im Zusammenhang mit dem SDM: Falls Sie während des Deployments im späteren Verlauf des Projektes auf Probleme stoßen und die Meldung `»Server <Server-Name> did not accept login request as admin on port <Server Port>«` erhalten, beachten Sie bitte den SAP-Hinweis 941150. In diesem Fall müssen Sie, wie im Hinweis empfohlen, eine aktuelle Version des SDM installieren. Da die Beschreibung einer SDM-Installation den Rahmen dieses Heftes sprengen würde, möchten wir Sie hier auf den SAP-Hinweis 860939 verweisen, in der sehr detailliert auf eine SDM-Installation bzw. ein Upgrade eingegangen wird.

4.4 Deployment der Muse-Bibliotheken in den SAP NetWeaver AS

Bisher haben wir unsere Entwicklungsumgebung aufgesetzt und Vorbereitungen innerhalb des PCK für die Entwicklung vorgenommen. Im Zuge dessen haben wir unserer Entwicklungsumgebung die Muse-Bibliotheken bekannt gegeben. Da der Adapter während der Entwicklung innerhalb des PCK, später aber in einem vollwertigen SAP XI laufen wird, müssen wir die von unserem Adapter benötigten Bibliotheken in den J2EE-Container

Index

A

ABAP-Mapping 19, 101
ABAP-Mapping-Klasse 24
ABAP/4-Debugger 101
ABAP/4-Entwicklungsumgebung 10
Adapter-Framework 46, 71, 72
Adapter-Modul 89
Adapter Engine 46
Adapter Metadata Upload 85
Adapterskelett 60
aai.properties 50
Änderungshistorie 63
Ant 15, 17, 53, 83
 aufrufen 84
Ant-Target 83
Ant-Task 54, 83
Apache Ant 17
application-j2ee-engine.xml 80, 91
application.xml 80
Audit-Log 88, 90
Audit-Nachrichten 77
Ausgangsadapter 31
Automatisches Mapping 42

B

Beisert 54
Beispieladapter 61
Benutzerrechte 48, 49
Branches 60
Breakpoint 101, 103
Build 83
Build-Skript 83
Build-Tool 17
Bytecode 18

C

C# 9
C++ 9
Central Information Provider 7

Change Management Service (CMS) 12
Chatten 45
CIM-Client 97, 98, 100
CIM-Server 95
ClassLoader 78, 86
com.echomine.jabber.Jabber 68
com.echomine.jabber.JabberContext 66
com.inqmy.lib.xml.StandardDOMWriter 36
com.sap.aii.af.monitor.api.
 AdapterMonitor 73
com.sap.aii.af.mp.module.Module 89
com.sap.aii.af.mp.module.ModuleHome 90
com.sap.aii.af.mp.module.ModuleLocal 90
com.sap.aii.af.mp.module.Module-
 LocalHome 90
com.sap.aii.af.mp.module.Module-
 Remote 90
com.sap.aii.af.ra.cci.NWConnection-
 Factory 73
com.sap.aii.af.service.auditlog.Audit 77
com.sap.aii.af.service.auditlog.Audit-
 MessageKey 77
com.sap.aii.af.service.trace.Trace 69
com.sap.aii.mapping.api.Streamtrans-
 formation 34
com.sap.aii.mapping.api.
 StreamtransformationConstant 34
com.sap.aii.mapping.api.StreamTrans-
 formationException 37
com.sap.engine.interfaces.connector.
 ManagedConnectionFactoryActivation 73
com.sap.sdm.ant.JarSAP 83
Common Client Interface 46
Component Build Service 12
Connection Management 47
Connection Pooling 72, 73, 86
connector-j2ee-engine.xml 78, 80

CPA-Cache 50, 75, 78
CPA-Cache-Metadatei 82
CPA-Cache-Metadaten 85
CVS 15, 16

D

Datenbanktabelle 22
Debug-Perspektive 103
Debugging 101
Debugging-Modus 102
Debugging in ABAP 101
Debugging in Java 102
DeltaV 16
DeltaV-Spezifikation 16
Deployment 53, 54, 83
Deployment-Deskriptoren 61, 62, 78
deprecated 74
Design Time Repository (DTR) 12, 15,
 16, 17
Development Components (DC) 13
Development Objects (DO) 13
Direction.OUTBOUND 78
DOM-Parser 35

E

Eclipse 15, 60
 Eclipse Foundation Inc. 15
Eclipse-Projekt 34, 57
Eingangsadapter 31
ejb-j2ee-engine.xml 90
EJB Module Project 89
Empfängervereinbarung 86
Enterprise Application Archive 48, 80
Enterprise Application Project 91
Enterprise Java Bean 89
Exchange-Infrastructure-Landschafts-
 topologie 7

F

File-Adapter 85

G

GNU Public License 52

Google 45

Google Talk 45

Gosling, James 9

Grafischer Mapping-Editor 39

Groovy 18

H

Hewlett-Packard 45

I

Inbound 82

Inbound Processing 73

InputStream 34

Instant Messaging System 45

Interaktion 77

Interface-Mapping 28

iXML-Funktionen 25

iXML-Library 25

J

J2EE 9, 15

J2EE-Applikation 57

J2EE Connector Architecture 45, 46

J2EE Container 57

J2EE Engine Deploy Tool 55

J2EE Engine Visual Administrator 48

Jabber 45

Jabber-Client 52

Jabber-Kommunikation 63

Jabber-Server 51, 52

Jar-Files 95

Java 9, 15

Entwicklungsumgebung 15

Java-Debugger 102

Java-IDE 95

Java-Mapping 34

Java-Schnittstelle 65

Java-Versionen 9

java.lang.ClassCastException 89

java.lang.Runnable 73

Java Community Process 46

Java Connector Architecture 9

Java Development Infrastructure (JDI) 15

Java Development Kit (JDK) 18, 59

Java Naming and Directory Interface
(JNDI) 74, 82, 90, 92Java Platform Debugger Architecture
(JPDA) 102

Java Specification Request 46

Java Virtual Machine (JVM) 9

javax.resource.cci.Connection 47, 75, 76

javax.resource.cci.ConnectionFactory 47,
71, 72

javax.resource.cci.ConnectionSpec 47

javax.resource.cci.Interaction 72, 77

javax.resource.ResourceException 66

javax.resource.spi.Connection-
EventListener 48javax.resource.spi.ConnectionManager
71, 73

javax.resource.spi.LocalTransaction 48

javax.resource.spi.ManagedConnection
72, 73, 75javax.resource.spi.ManagedConnection-
Factory 72, 73

JEE 9

Jive-Software 52

K

Klassennamen 79

Klassenschnittstellen 24

Komponentenmodell 10, 12

Konfigurationsoberfläche 82

Konstruktordefinition 66

L

Library Project 55

LocalTransaction 48

log-configuration.xml 79, 82, 91

Logging-Kategorie 78

Logmanager 79

Lose Kopplung 12

M

make 15, 17

makefile 18

ManagedConnection 48

Managed Environment 48

Mapping-Klasse 34

Mapping-Programmierung 19

Mapping-Routinen 19

Mapping-Test 102

Metadaten 78, 80

Microsoft 9

Modulentwicklung 89

Muse-Bibliotheken 51

N

Namensgebung in Java 62

Name Service 12

Namespace 78

NetBeans 15

NoTransaction 48

O

Open Source 15

Organize Imports 98

Outbound 82

P

Package Explorer 59, 65

Partner Connectivity Kit 45, 48, 85

Perforce 66

Projekt versionieren 60

Psi 52

Public Part 12

R

ra.xml 61, 79

Receiver-Determination 32

Refactoring 62, 79

Referenz 81

Referenzimplementierung 61

Registrierung der Monitoring-Schnitt-
stelle 73

Remote-Deployment 54

Resource Adapter Archive 48

RFC-Destinationen 20

S

SampleRa.xml 61

SAP-XML-Toolkit 34

SAP NetWeaver 5

SAP NetWeaver Developer Studio
 (NWDS) 13, 14
 Installation 14
 SAP NetWeaver Development Infrastructure (NWDI) 9, 11
 Architektur 10
 SAX-Parser 35
 SDA 18
 SDA-Erzeugung 54
 SDM 18
 SDM-Administrator 56
 SDM-API 54
 SDM-GUI 56
 SDM-Installation 54
 SDM-Repository 57
 SDN Blog 54
 Security Roles 49
 Sender Agreement 33
 Senderschnittstelle 34
 Sendervereinbarung 86
 Sequenzdiagramm 72
 Serialisieren 36
 Serialisierer 37
 Server Provider Interface 47

Service Programming Interface 46
 SIGNATURE 67
 SLD-API 95
 Software Components (SC) 12
 Software Delivery Archive 48
 Software Deployment Manager 48
 Stateless Session Bean 89
 Subclipse 15, 17, 60
 substring 43
 Subversion 15, 16, 60
 Subversion Book 60
 Sun 15
 System Contracts 47
 System Landscape Directory (SLD) 7, 12, 95

T

Tags 60
 TCP/IP-Verbindung 21
 TRACE 66
 Trace 79
 Transaktionsmanagements 48
 Trunk 60

V

Verbindungsmanagement 47
 VERSION_ID 66
 Versionsmanagement 16
 Versionsverwaltungssystem 60

W

WebDAV 16
 Wildfire 52

X

XI-Adapter 50
 XI-Adapter-Framework 46
 XI-Bibliotheken 50
 XI-Directory 29
 XI-Repository 25
 XMPP 45