Groupware Competence Center

Universität Paderborn

IBM Workplace Business Component

Entwicklung einer Business Component auf Basis der IBM Workplace API

Prof. Dr. L. Nastansky

Projekt Office Systeme

Wintersemester 2004 / 2005

Betreuer:

Dipl. Wirt. Inf. Ingo Erdmann

Vorgelegt von:

Carsten Böhle
(6220060)

Christoph Danne
(6219631)

Viktor Dück
(6219895)

Inhaltsverzeichnis:

Abkürzungsverzeichnis

C2A	Click to Action
CACI	Collaborative Application Component Interface
EAR	Enterprise Application Archive
EJB	Enterprise Java Bean
HTML	Hyper Text Markup Language
J2EE	Java 2 Platform Enterprise Edition
JSP	Java Server Page
LDAP	Lightweight Directory Access Protocol
RDBMS	Relationales Datenbank Management System
RSS	Rich Site Summary
UI	User Interface
URL	Uniform Resource Locator
WAR	Web Application Archive
WAS	WebSphere Application Server
WPS	WebSphere Portal Server
WSAD	WebSphere Application Developer
WSDL	Web Services Description Language
XML	Extensible Markup Language

Abbildungsverzeichnis

1. IBM Workplace

1.1. Einleitung

Workplace ist eine Sammlung von Produkten und Technologien, die das Verteilen von Anwendungen, die gemeinsame Datenhaltung sowie die Zusammenarbeit im Team unterstützen. IBM stellt an die Plattform den ehrgeizigen Anspruch, die Produktivität von Menschen bei ihrer täglichen Arbeit zu erhöhen. Dazu erweitert Workplace die bestehende Produktpalette sowohl auf der Server- als auch auf der Client Seite.

Zwar existieren bereits einige Anwendungen für Workplace, doch hängt der Nutzen einer solchen Plattform entscheidend von den Möglichkeiten ab, ihre Funktionalität durch eigene Applikationen zu erweitern. Aus diesem Grund sollen im Rahmen dieser Projektarbeit zunächst die Möglichkeiten der Anwendungsentwicklung für die Workplace Plattform evaluiert werden. In einem weiteren Schritt soll dann exemplarisch eine Anwendungskomponente (sog. Business Component) entwickelt werden.

Workplace Anwendungen sollen auf mehreren Clients genutzt werden können. Grundsätzlich stehen eine Web basierte Portalumgebung und ein Rich Client zur Auswahl, wobei der Rich Client zu Beginn dieser Projektarbeit noch nicht öffentlich verfügbar war. Die folgenden Ausführungen beziehen sich auf die Entwicklung von Business Components, die für die Darstellung die Workplace Portalumgebung nutzen. Große Teile der Beschreibungen, die nicht die Darstellungsform betreffen, lassen sich aber auf die Entwicklung für den Rich Client übertragen.

1.2. Workplace Server Architektur

Die Workplace Server Architektur baut auf vorhandenen Server-Produkten auf. Auf der untersten Ebene stellt ein WebSphere Application Server (WAS) die Infrastruktur zur Ausführung von Java 2 Enterprise Edition (J2EE) Anwendungen zur Verfügung und bietet somit insbesondere eine Ausführungsumgebung für Enterprise Java Beans (EJBs), die in Workplace- Applikationen oft für das Abbilden der Anwendungslogik genutzt werden.

Auf dem WAS baut der WebSphere Portal Server (WPS) auf. Dieser bietet die Infrastruktur für Portlet-basierte Anwendungen und ist für die Darstellung der Komponenten zuständig. Auch können hier die Teile der Applikationslogik angesiedelt werden, die keine J2EE Ausführungsumgebung benötigen. Dies können z.B. Java Beans für Dienste wie Session Handling und Caching sein.

Neben diesen Schichten steht eine relationale Datenbank (RDBMS), die die zentrale Datenhaltung übernimmt und über definierte Schnittstellen das Auslesen und Manipulieren dieser Daten erlaubt.

Der Workplace Server fügt sich in diese Struktur ein und erweitert die vom WAS und WPS zur Verfügung gestellten Dienste. Dabei liegt der Fokus auf der Bereitstellung von kollaborativen Funktionen, die bisher überhaupt nicht oder nur aufwendig zu implementieren waren. Teams können einzelne Anwendungen gemeinsam nutzen, um so innerhalb der Gruppe Diskussionen zu führen oder auf einer gemeinsamen Datenbasis zu arbeiten. Diese Teams können dynamisch beim Erstellen der Anwendung definiert und später geändert werden. Weiterhin wird die Kommunikation durch Mail- und Instant Messaging Dienste unterstützt.

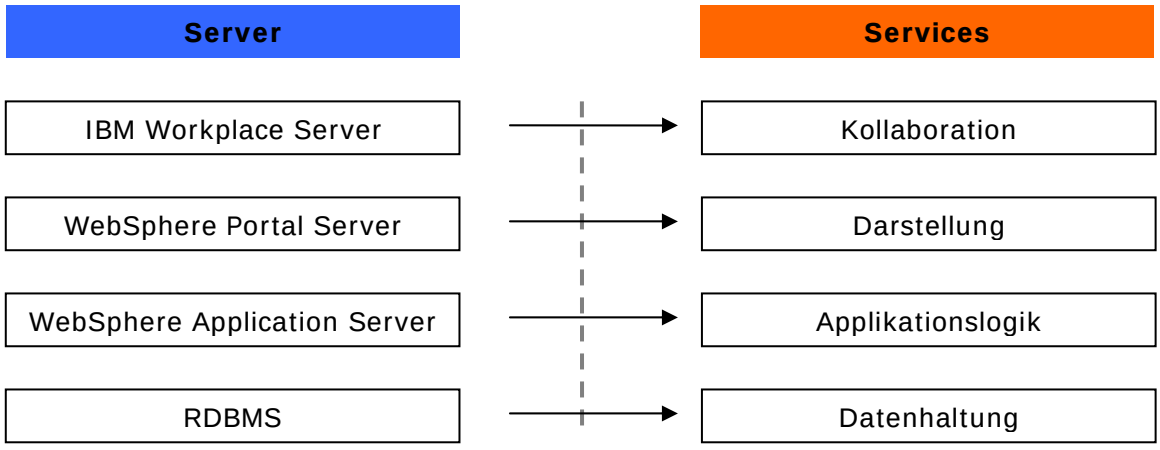

Abb. 1 Workplace Server-Architektur

1.3. Business Components und Anwendungen

Business Components sind die Kernkomponenten von Workplace-Anwendungen. Jede Business Component kapselt eine gewisse Funktionalität, die in Applikationen genutzt werden kann. Workplace stellt bereits einige häufig verwendete Komponenten zur Verfügung, wie z.B. Mail, Diskussionsforen, Kalender und Suchfunktionen, um nur einige zu nennen. Diese können nach Belieben kombiniert und zu Anwendungen zusammengesetzt werden. Eine Anwendung ist somit eine Komposition mehrerer Business Components, die für die Nutzung im Web Client auf Seiten organisiert werden. Um den mehrfachen Einsatz einer Anwendung zu vereinfachen, können Templates erstellt werden, die alle Informationen über die Zusammensetzung der Anwendung beinhalten. Mit Hilfe eines Template können dann beliebig viele Instanzen dieser Anwendung erzeugt werden.

2. Business Component Entwicklung

2.1. Nützliche Vorkenntnisse

Wie bereits in Abschnitt 1.2 erwähnt, baut die Workplace-Architektur insbesondere auf dem WebSphere Portal Server auf. Kenntnisse im Bereich der Portlet Entwicklung auf Basis der IBM Portlet API[1] sind für Business Components sehr hilfreich, da die Benutzerschnittstelle für den Web Client als Portlet implementiert wird. Auch das Verständnis grundlegender Konzepte wie z.B. der Model View Control Architektur (MVC) ist hilfreich, um das Zusammenspiel der verschiedenen Komponeten nachvollziehen zu können. All dies ist sehr ausführlich nachzulesen bei Rodriguez (2004).

Die Portlet-Entwicklung umfasst das Erstellen diverser Anwendungskomponenten auf Basis verschiedener Technologien. Diese spielen auch bei der Erstellung eigener Business Components eine Rolle.

Hier sind insbesondere die stateless Enterprise Java Beans (EJBs) von Interesse, da sie oft genutzt werden, um die Anwendungslogik von Business Components abzubilden.

Weiterhin kommen für die Generierung der Portletinhalte oft Java Server Pages (JSPs) zum Einsatz. Diese Technologie erlaubt das Einbetten von Java Code-Fragmenten in HTML Code, um dynamische Inhalte zu generieren. Die nötigen Informationen sowohl zu EJBs als auch zu JSPs sind bei Green (2002) nachzulesen.

2.2. Architektur einer Business Component

Applikationen auf Basis der J2EE Plattform besitzen in der Regel eine mehrschichtige Architektur, bei der jeder Schicht bestimmte Verantwortlichkeiten zugewiesen werden[2]. Diese Architekturen sind erprobt und eignen sich für die Entwicklung flexibler und skalierbarer Anwendungen, weshalb Workplace eine ähnliche Architektur für Business Components definiert.

[1] Diese ist nicht gleichzusetzen mit der vielleicht bekannteren JSR 168 Java Portlet Specification, vgl. Hepper (2003)

[2] Vlg. Dale 2002, Kapitel „Distributed Multitiered Applications"

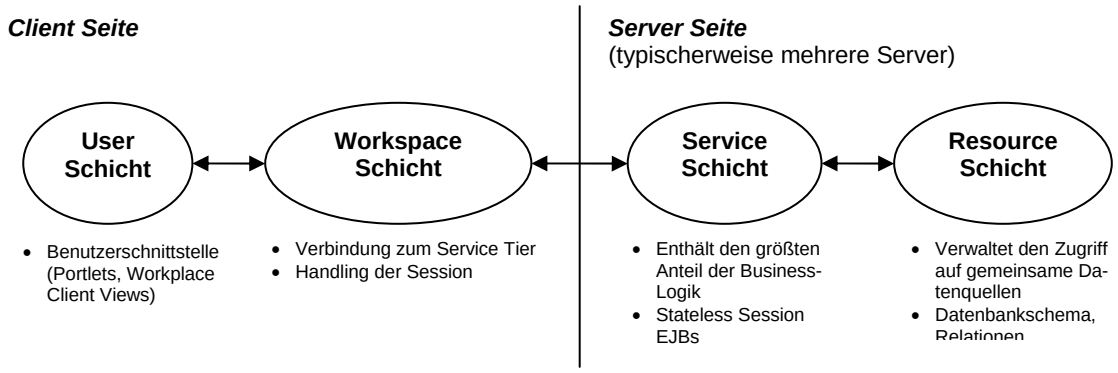

Abb. 2 Architektur einer Business Component[3]

In der User-Schicht werden alle Komponenten zusammengefasst, die für die Interaktion mit dem Benutzer verantwortlich sind. Wir betrachten hier die Darstellung der Anwendung als Portlet, so dass in der User Schicht hauptsächlich JSP Elemente anzutreffen sind. Eine alternative Umsetzung wäre die Implementierung als Workplace Client View für die Darstellung im Rich Client.

Die Workspace-Schicht ist ebenfalls auf der Client-Seite angesiedelt und fungiert als Delegationsschicht zwischen dem User Interface (UI) und der Logik. Sie dient dazu, Details über die Art der Implementierung der Service-Schicht vor den UI-Komponenten zu verstecken, indem sie alle Aufrufe der User Schicht weiterdelegiert. Somit erreicht man eine lose Kopplung der beiden Seiten und die User-Schicht benötigt keine Informationen darüber, wo die Service-Schicht angesiedelt ist und wie sie implementiert wird. Weiterhin kann die Workspace-Schicht auch eigene Funktionalität anbieten, indem sie clientseitig Dienste wie z.B. Caching realisiert. Die Implementierung dieser Schicht erfolgt in der Regel durch normale Java-Klassen bzw. Java Beans. Sowohl User- als auch Workspace-Tier können bei Verwendung eines Rich Clients auf der Clientseite, bei Entwicklung für eine Web Umgebung auf dem Portal Server, implementiert werden.

Die Service-Schicht bildet die eigentliche Logik ab, die dem Benutzer von der Business Component bereitgestellt wird. Dies wird zur Sicherung einer hohen Skalierbarkeit der Anwendung oft durch stateless session EJBs realisiert, theoretisch sind anwendungsabhängig aber auch andere Implementierungen wie z.B. Web Services denkbar. Allen Implementierungsarten ist gemein, dass sie eine verteilte Ausführung erlauben und session-unabhängig sind, also keine Informationen über Anfragen von Clients über die Dauer der Anfrage hinaus speichern.

Als Resource-Schicht bezeichnet man alle Komponenten, die eine persistente Datenspeicherung für die Business Component ermöglichen. Diese Schicht umfasst somit einerseits die

[3] Vgl. Groeger / Gorman (2004)

eigentlichen Datenspeicher wie z.B. relationale Datenbanken, aber auch die Infrastruktur, um auf diese Daten zuzugreifen und sie zu modifizieren. Dies könnte z.B. eine JDBC-Schnittstelle sein.

2.3. Das Workplace API Toolkit

Für die Entwicklung von Business Components wird ein Application Programming Interface (API) zur Verfügung gestellt, das Schnittstellen und einige Services anbietet, um die Einbindung in die Workplace Plattform zu ermöglichen.

Wir wollen im Folgenden einen kurzen Überblick über die Komponenten des Workplace API Toolkits geben und werden auf einige in Kapitel 3 noch näher eingehen, indem wir ihre Implementierung bzw. Nutzung im Rahmen der von uns erstellten Business Component erläutern. Für eine ausführliche Beschreibung sei auf den Workplace API Toolkit User's Guide (IBM, 2004) und die dazugehörige Dokumentation der Java Klassen verwiesen.

Die Mail Messaging Service Provider Interfaces (SPI) bieten eine Sammlung von Klassen und Schnittstellen, die es erlauben, Erweiterungen zu den Maildiensten von Workplace zu entwickeln. Durch diese Klassen kann der Header und der Inhalt einer Mail vor der Zustellung untersucht und ggf. modifiziert werden. Dies könnte beispielsweise für Spam-Filter oder Virenscanner genutzt werden.

Ähnlich bieten die Instant Messaging SPIs die Möglichkeit, die bestehenden Messaging-Funktionen zu erweitern. Ein typisches Anwendungsbeispiel für den Einsatz dieser SPIs wären Anwendungen zum Aufzeichnen („Loggen“) von Gesprächen oder Übersetzung des Textes vor der Zustellung an den Empfänger.

Workplace erweitert die bestehenden JSP Bibliotheken, die durch den Portal Server bereitgestellt werden. Dies sind im Wesentlichen zwei JSP Tags, mit denen sich Dienste für die Kollaboration zwischen Teilnehmern einer Workplace Umgebung einfach implementieren lassen.

Das so genannte Person Tag erzeugt „people awareness“. Dies bedeutet, dass zu einem gegebenen Namen Funktionen wie „Email schreiben“ oder „Personenprofil einsehen“ aufgerufen werden können. Weiterhin kann der online Status der Person eingesehen werden um ggf. direkt über die Instant Messaging Dienste Kontakt zu der Person aufzunehmen. Für das direkte Verfassen einer Email reicht es aus, im entsprechenden JSP Tag die Adresse und den anzuzeigenden Namen des Adressaten anzugeben. Sollen die erweiterten Funktionen wie „Personenprofil einsehen“ oder „Instant Messaging starten“ genutzt werden, müssen in dem Workplace Adressverzeichnis die nötigen Informationen zu der Person hinterlegt sein.

Weiterhin bieten die Workplace JSP-Bibliotheken das so genannte Online Center Tag an, das Dienste für so genannte „presence awareness“ bereitstellt. Der Einsatz dieses Tags auf einer Portal Seite bietet dem Benutzer die Möglichkeit, seinen Online Status festzulegen, d.h. sich zur Nutzung von Instant Messaging Diensten zu verbinden. Weiterhin können hier Einstellungen zur Nutzung dieser Dienste vorgenommen werden. Dieses Tag ist notwendig, damit die Messaging Funktionen des Person Tag korrekt funktionieren und ist standardmäßig bereits in die Portalseiten integriert. Werden jedoch in Applikationen weitere Portlets in Popup Fenstern dargestellt, so muss das Online Center Tag erneut eingebunden werden, um ein korrektes Arbeiten des Person Tag zu gewährleisten.

Der weiterer interessanter Bestandteil der Workplace API sind die Collaborative Application Component Interfaces (CACIs). Diese Schnittstellen können von Entwicklern genutzt werden, um die Business Component in die Workplace Umgebung einzubetten und spezielle Funktionen dieser Plattform zu nutzen. Je nach Bedarf können ausgewählte Schnittstellen in einer EJB implementiert werden.

Das `Lifecycle` Interface deklariert eine Methode, die aufgerufen wird, wenn die Business Component einem Template hinzugefügt wird oder ein Template mit der Business Component zu einer Anwendung instanziiert wird. Analog existiert eine Methode für das Entfernen der Komponente. Dies bietet die Möglichkeit, von der Business Component benötige Ressourcen zu erstellen und zu löschen. Weiterhin muss beim Anlegen eine Kennung zurückgegeben werden, die dazu dient, die Instanz der Business Component später eindeutig zu identifizieren, um z.B. die passenden Informationen aus einer Datenbank auszulesen und um beim Entfernen der Komponente die zu löschenden Ressourcen zu identifizieren. Falls eine Business Component keine speziellen Ressourcen benötigt, braucht dieses Interface nicht implementiert zu werden. Allerdings wird für die Implementierung aller anderen CACIs auf die eindeutige Kennung der Komponenten zurückgegriffen, so dass die Implementierung einer der Schnittstellen immer auch die Implementierung des `Lifecycle` Interfaces erforderlich macht.

Jede Business Component verwaltet Informationen darüber, welche Benutzer- und Benutzergruppen welche Zugriffsrechte auf die von ihr verwalteten Ressourcen haben. Das `Membership` Interface dient dazu, die Business Component darüber in Kenntnis zu setzen, dass den zugeordneten Gruppen Mitglieder hinzugefügt wurden oder aus ihnen Mitglieder entfernt wurden. Die Business Component kann somit Benutzern den Zugriff auf die Ressourcen erlauben oder verweigern.

Gegebenenfalls möchte man eine Business Component in mehreren Anwendungen einsetzen, aber jeweils unterschiedliche Konfigurationen oder Standardvorgaben auf Anwendungsebene definieren. Dies lässt sich über das `Templateable` Interface realisieren. Eine Implementierung dieser Schnittstelle bietet die Möglichkeit, zu einer Business Component beim Erstellen einer neuen Anwendung aus einem Template Variablen zu setzen, um so z.B. Voreinstellungen vorzunehmen, die nur für diese bestimmte Anwendung gedacht sind.

Das `Sensor` Interface bietet lediglich die Möglichkeit, vom Workplace Server gesammelte Informationen über die Ressourcen einer Business Component auszulesen. Hierzu gehören die Daten der Erstellung, letzte Änderung und letzte Verwendung sowie die Größe der verwendeten Ressourcen.

Das `Transactional` Interface liefert durch eine einfache Methode Informationen darüber, ob die Implementierungen der durch das `Lifecycle` und `Membership` Interfaces deklarierten Methoden globale Transaktionen unterstützen[4].

[4] Für nähere Informationen zu globalen Transaktionen siehe Green (2002), Kapitel „Transactions“ (http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Transaction2.html)

3. RSS Reader Business Component

3.1. Intention

Neben der Einarbeitung in die Möglichkeiten der Anwendungsentwicklung für die Workplace-Plattform ist es Ziel dieser Projektarbeit, die gewonnenen Kenntnisse umzusetzen und exemplarisch eine Business Component zu entwickeln. Auffälligerweise gab es unter allen Anwendungen, die bereits mit Workplace ausgeliefert werden, keine Möglichkeit, beliebige RSS Newsfeeds anzuzeigen. Zwar existieren Business Components zur Darstellung von Nachrichten, allerdings kann der Benutzer hier lediglich aus einer vorgegebenen Menge an Seiten auswählen und hat nicht die Möglichkeit, eigene Nachrichtenquellen in Form von RSS Feeds anzugeben.

Deshalb wurde im Rahmen dieses Projektes ein RSS Reader für die Verwendung in der Workplace-Umgebung entwickelt. Dieser besteht aus zwei Business Components (RSSReader und RSSContent), die Bestandteil einer Anwendung sein können und miteinander interagieren.

3.2. Komponenten und Architektur

Der größte Teil der Funktionalität wird durch die RSSReader Business Component bereitgestellt, während RSSContent lediglich die Darstellung der Nachrichtenseiten innerhalb der Portalumgebung übernimmt. Beide Komponenten haben die in Abschnitt 2.2 beschriebene Architektur, um sich in die Workplace Umgebung einzufügen. Die RSSContent Business Component implementiert allerdings nicht alle Teile dieser Architektur, da diese für ihren Verwendungszweck nicht notwendig sind. Am Beispiel RSSReader hingegen lassen sich Implementierungen der verschiedenen Schichten wieder finden, weshalb wir unsere Ausführungen in diesem Kapitel hierauf beschränken.

Wir benennen im Folgenden die Komponenten der Software mit den Namen, die die entsprechenden Projekte im Quellcode tragen, um eine einfache Zuordnung zu gewährleisten.

Abbildung 3 gibt einen Überblick über Komponenten und Architektur der RSSReader Business Component.

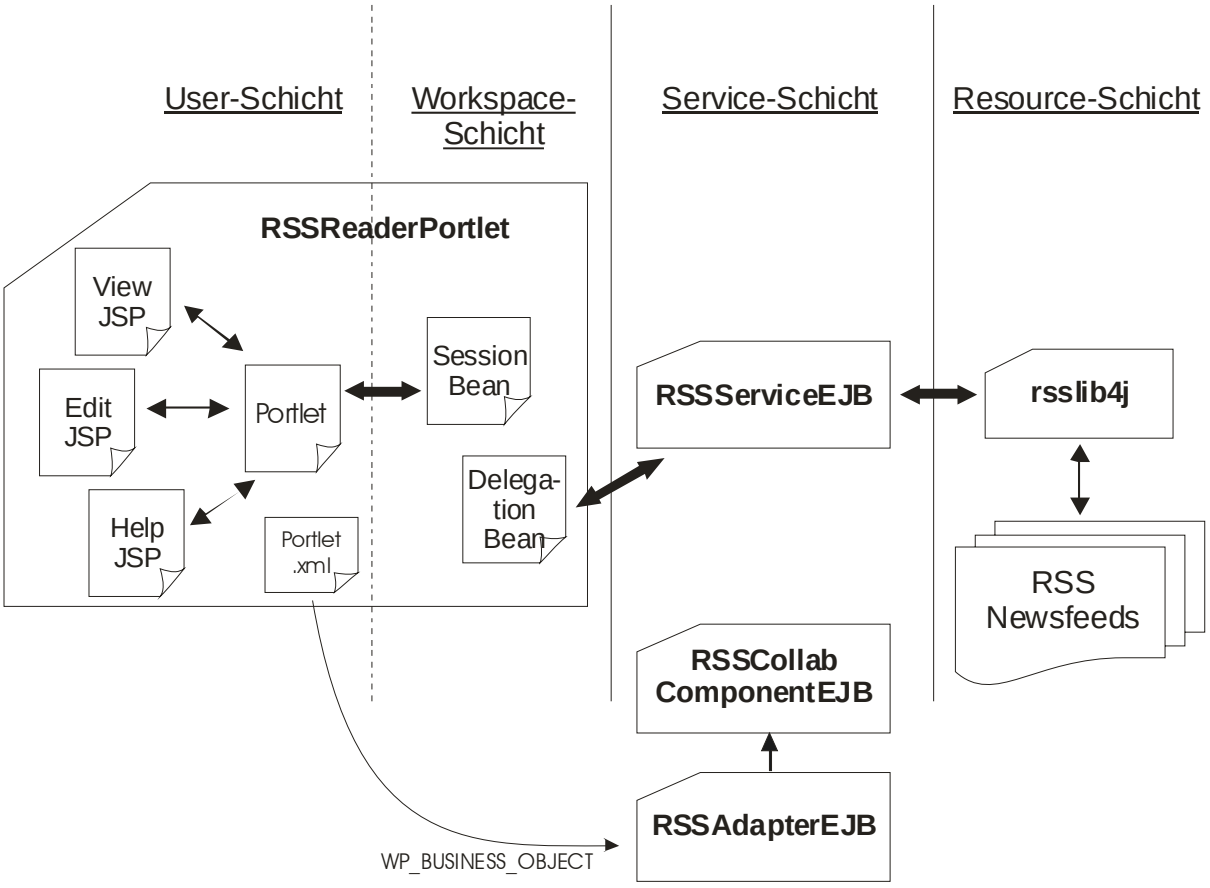

Abb. 3 Architektur der RSSReader Business Component

Die für den Benutzer sichtbaren Elemente sind alle in der Komponente `RSSReaderPortlet` zusammengefasst. Diese besteht aus mehreren JSPs zur dynamischen Generierung des HTML Codes für die verschiedenen Anzeigemodi des Portlets (View, Edit, Help). Die Klasse `RSSReaderPortlet` verarbeitet die im Portlet auftretenden Ereignisse und wählt je nach Benutzer-Interaktion dynamisch die entsprechende JSP zur Generierung der Inhalte aus. Die JSPs dieser Komponente stellen im Kontext unserer Business Component die User-Schicht dar. Diese Schicht ist die einzige Möglichkeit der Benutzer-Interaktion und in diesem Fall als Portlet implementiert, so dass die Benutzung über einen Web Browser erfolgt.

Die Workspace-Schicht wird durch Klassen realisiert, die ebenfalls in der Komponente `RSSReaderPortlet` angesiedelt sind. Um die Performanz beim Neuladen der Seite nicht negativ zu beeinflussen, werden alle Inhalte der Nachrichten-Channels sessionabhängig zwischengespeichert. Ein Neuladen der Seite löst also keine Aktualisierung der Inhalte aus, da dies bedingt durch das häufige Neuladen der Portalseite die Performanz negativ beeinflussen würde. Eine Aktualisierung der jeweiligen Inhalte wird vorgenommen, wenn ein neuer Channel hinzugefügt wird oder der aktuell angezeigte Channel gewechselt wird. Eine manuelle Aktualisierung des aktuellen Channels ist ebenso möglich.

Hauptaufgabe der Business Component ist das Auslesen der Daten eines RSS Newsfeeds zu einer vom Benutzer festgelegten URL. Diese Anwendungslogik ist in der Komponente `RSSServiceEJB` gekapselt und dort in Form einer stateless session EJB implementiert. Dieses Modul bietet beliebig vielen Clients unabhängig voneinander den Dienst an, zu einer gegebenen URL die Daten aus der RSS Datei auszulesen und in ein spezielles Objekt gekapselt zurückzugeben. Dies ist in unserem Fall die Implementierung der Service-Schicht.

Eine Resource-Schicht in der in Abschnitt 2.2 beschrieben Form wurde bei uns nicht vollständig implementiert. Dies ist auf die Tatsache zurückzuführen, dass für unsere Anwendung die benötigten Daten die Inhalte der RSS Feeds sind, die laufend aus dem Internet abgerufen werden. Dies macht eine persistente Datenhaltung in einer gesonderten Datenbank überflüssig. Die Resource-Schicht muss aber neben der eigentlichen Datenspeicherung auch die Schnittstellen für den Zugriff auf diese Daten bereitstellen. Der RSS Reader verwendet hierzu die offene Java Bibliothek rsslib4j[5]. Diese wird dazu genutzt, RSS Feeds auszulesen und in eine Form zu übersetzen, die von der `RSSServiceEJB` weiterverarbeitet werden kann. Die einzige Information, die über die Ausführung einer Session hinaus gespeichert wird, ist die Konfiguration der RSS Channel zu einer Instanz der Business Component. Somit ist es möglich, dass ein Benutzer mehrere unabhängig voneinander konfigurierbare Portlets hat. Diese Informationen werden allerdings direkt nach dem Hinzufügen eines Channels in eine zentrale Konfigurationsdatenbank des WPS geschrieben.

3.3. Collaborative Components

Wie in Abschnitt 2.3 beschrieben, kann eine Business Component die Collaborative Application Component Interfaces implementieren. In diesem Fall wird sie Collaborative Component genannt. Um ausgewählte CACIs zu implementieren, wird die Business Component um zwei EJBs erweitert.

Die `RSSCollabComponentEJB` übernimmt die eigentliche Implementierung dieser Interfaces. Dabei ist es wichtig zu beachten, dass das Remote Interface der EJB um diese Interfaces erweitert wird, während die eigentliche Bean Klasse lediglich die zugehörigen Methoden implementiert.

Da sich die in der öffentlichen API definierten CACIs von den von Workplace intern verwendeten Interfaces unterscheiden, benötigen wir die `RSSAdapterEJB`, die diese Aufrufe ent-

[5] Informationen zu dieser Bibliothek sind unter http://devzone.stealthp.org/cms/index.php?page=RSSLib4J zu finden

sprechend übersetzt. Diese EJB besteht aus einigen Klassen, die von IBM mit dem Workplace API Toolkit ausgeliefert werden.

Um die Komponenten zu verbinden, muss das Portlet einer Business Component in seiner `portlet.xml` Konfigurationsdatei einen Parameter WP_BUSINESS_OBJECT enthalten, der über den JNDI Namen auf die Adapter EJB verweist. Diese wiederum hat einen EJB-Verweis auf die `RSSCollabComponentEJB`. Beim Hinzufügen des Portlets zu einer Anwendung lokalisiert der Workplace Server so die Adapter EJB und kann darüber auf die Implementierung der CACIs zugreifen.

Der RSSReader implementiert in der Klasse `CollabComponentBean` die Interfaces `Lifecycle`, `Templateable` und `Transactional`, um beim Erstellen einer Anwendung, die den RSSReader beinhaltet, die Möglichkeit zu bieten, Standard-Newschannel einzutragen. Leider ist es uns nicht gelungen, dieses Feature vollständig umzusetzen, da die Methoden der CACIs offenbar nicht korrekt vom Workplace Server aufgerufen werden. Wir vermuten, dass nicht dokumentierte Neuerungen im Workplace Server 2.5 beta hierfür verantwortlich sind, da wir auch an anderen Stellen undokumentierte Änderungen feststellen mussten, die wir allerdings selbst lokalisieren und entsprechend behandeln konnten.

3.4. Workplace JSP Tags – Das Person Tag

Eine wichtige Erweiterung des WPS durch Workplace sind die in Kapitel 2.3 beschriebenen JSP Tag Libraries. Der RSSReader benutzt das Person Tag, um zusätzliche Informationen und Möglichkeiten zur Kontaktaufnahme zu den Autoren der Newsfeeds zu bieten. Ein Klick auf den Namen des Autors öffnet ein Menü, in dem der Benutzer zwischen den verschiedenen Funktionen wählen kann.

Ist zu einem News-Artikel die Email Adresse des Autors angegeben, so werden hier immer die Funktionen „Email senden“ und „In Adressbuch aufnehmen“ angeboten. Workplace versucht aber auch, den Autoren durch seine Email Adresse im internen Adressbuch zu lokalisieren. Wenn dies gelingt, wird statt der Email Adresse der Name des Autors aus dem Adressbuch ausgelesen und angezeigt. Außerdem kann somit der Online Status der bestreffenden Person ausgemacht und angezeigt werden. Das Menü wird um Funktionen zur direkten Kontaktaufnahme erweitert.

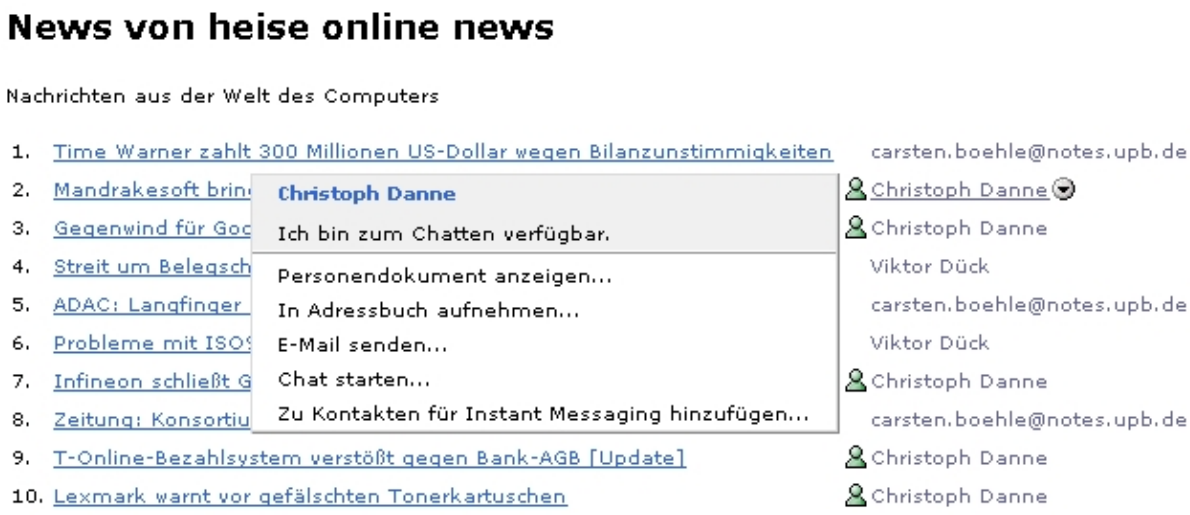

Abb. 4 Funktionen des Person Tag

Diese Funktionalität ist insbesondere interessant, wenn Firmen oder Universitäten interne Nachrichten über RSS Feeds veröffentlichen möchten und die Autoren somit im internen Adressbuch (z.B. ein LDAP Verzeichnis) eingetragen sind.

Workplace bietet Entwicklern die Möglichkeit, diese Formen der „People awareness“ in ihren Anwendungen zu benutzen. Hierzu ist ein spezielles JSP Tag notwendig, das recht einfach auf die entsprechenden Seiten eingefügt werden kann. Um die oben dargestellte Funktionalität zu erhalten, wird die JSP Tag Bibliothek `people.tld` benutzt, um Zugriff auf das Person Tag zu erhalten. In der `RSSReaderPortletView.jsp` finden sich dazu folgende Einträge:

```
<%@ taglib uri="/WEB-INF/tld/people.tld" prefix="pa"%>
...
<pa:person value='<%=item.getAuthor()%>' valueType="EMAIL"></pa:person>
```

Die RSS Spezifikation 2.0 sieht vor, zu jedem Artikel die Email Adresse des entsprechenden Autors mit zu übermitteln[6]. Da dies allerdings bisher in nur wenigen Channels umgesetzt ist und die angegebenen Autoren auch nicht in unserem Adressbuch enthalten wären, haben wir zu Test- und Demonstrationszwecken ein eigenes RSS Feed geschrieben und den Artikeln entsprechende Autoren zugewiesen[7].

3.5. Click to Action

Click to Action (C2A) ist eine Technik, die Interaktion zwischen verschiedenen Portlets ermöglicht, sofern diese sich im Portal auf der gleichen Seite befinden. Bestimmte Elemente

[6] Vgl. RSS 2.0 Spezifikation unter http://blogs.law.harvard.edu/tech/rss

[7] Das RSS Newsfeed ist unter der URL http://homepages.upb.de/cboehle/workplacenews.xml erreichbar.

einer Seite werden mit einem Menü versehen, das es erlaubt, Aufrufe und Parameter an andere Portlets zu senden. Der WPS unterstützt den Einsatz von C2A und erlaubt somit auch die Nutzung in der Workplace-Umgebung.

Jedes Portlet speichert die von ihm akzeptierten Datentypen in einer WSDL-Datei, die dem Standard zur Beschreibung von Webservices folgt[8]. Beim Laden einer Portalseite untersucht der Property Broker des WPS die WSDL Dateien der enthaltenen Portlets.

Durch spezielle JSP Tags können Portlets definieren, von welchem Datentyp ihre Ausgabewerte sind. Der Property Broker vergleicht beim Laden einer Seite die Angaben aus diesen JSP Tags mit den Informationen aus den WSDL Dateien und generiert bei Übereinstimmungen ein Menü, durch das die Methoden der Zielportlets aufgerufen werden können.

Die Auswahl eines Menüpunktes führt dazu, dass die Methode `actionPerformed` des Zielportlets aufgerufen wird. Hier kann nun das Zielportlet die übermittelten Parameter aus dem Request auslesen und behandeln. Damit muss das Zielportlet nicht zwischen intern ausgelösten Ereignissen und Aufrufen über C2A unterscheiden. Alle Aufrufe werden in der `actionPerformed` Methode gleich behandelt und es ist theoretisch auch möglich, dass C2A und Benutzerinteraktion im Zielportlet selbst Events vom gleichen Typ auslösen. Eine Anleitung zur Integration von C2A Technologie in Portlets findet man bei Roy-Chowdhury (2002).

Bisher öffnet sich im RSSReader beim Klicken auf eine Schlagzeile der entsprechende Artikel in einem neuen Browser-Fenster. Eine wünschenswerte Alternative ist es, die Seiten innerhalb des Portals anzuzeigen, damit der Benutzer nicht zwischen der Nachrichten-Übersicht und den eigentlichen Artikeln wechseln muss und so beim Lesen einzelner Artikel trotzdem die Übersicht über alle Nachrichten behält. Dies lässt sich durch C2A realisieren. Dazu wurde eine weitere Business Component namens RSSContent entwickelt, deren Portlet es erlaubt, die Webseite zu einer übergebenen URL zu laden und darzustellen.

Abb. 5 Click to Action Menü

[8] Für die Spezifikation der WSDL siehe http://www.w3.org/TR/wsdl

Das RSS Reader Portlet bettet in der Seite zur Anzeige der Nachrichten ein spezielles JSP Tag mit der folgenden Syntax ein:

```
<%@ taglib uri="/WEB-INF/tld/c2a.tld" prefix="c2a" %>
….
<c2a:encodeProperty type="plainurl" namespace="http://gcc.upb.de/lwp/c2a"
name="url" value="<%=item.getURL()%>">….</c2a:encodeProperty>
```

Der Property Broker prüft nun beim Laden des RSS Readers, ob auf der gleichen Seite ein Portlet verfügbar ist, das in seiner WSDL Datei spezifiziert, Daten vom Typ `plainurl` verarbeiten zu können. Dies ist beim RSS Content Portlet der Fall:

```
<xsd:simpleType name="plainurl">
```

In dieser Datei ist auch spezifiziert, was beim Eingehen eines solchen Aufrufes geschehen soll, in diesem Fall wird die Methode `actionPerformed` mit dem Event `loadURL` aufgerufen.

```
<portlet:action name="loadURL" type="simple" caption="laden" description=""
/>
```

4. Auswertung und Ausblick

Wie bereits aus den vorherigen Abschnitten hervorgeht, sind Business Components recht komplexe Anwendungskomponenten, deren Entwicklung Kenntnisse verschiedener Technologien voraussetzt. Dies macht das Erstellen neuer Business Components ohne die nötigen Vorkenntnisse zu einer nicht trivialen Aufgabe.

Allerdings baut Workplace ausschließlich auf Standard-Technologien auf, die vielen Software-Entwicklern bereits bekannt sein dürften. Damit wird ihnen einerseits der Einstieg in die neue Plattform erleichtert und außerdem die Grundlage für die Portierung bestehender J2EE Anwendungen geschaffen.

Die von Workplace verwendete Architektur ist bereits erprobt und bietet die Grundlage für skalierbare und erweiterbare Anwendungen. Komplexe Anwendungen können hierdurch relativ einfach auf mehrere Server verteilt werden. Weiterhin bieten der Workplace Server und das Workplace Toolkit Möglichkeiten, Anwendungen um kollaborative Funktionen zu erweitern. Diese Funktionen waren bei der Portlet Entwicklung vorher gar nicht oder nur aufwendig zu realisieren. Ein Beispiel hierfür sind die Dienste, die durch das Person Tag wie in Abschnitt 3.4 beschrieben hinzugefügt werden. Außerdem bietet Workplace die Möglichkeit, viele kleine Komponenten in einer Umgebung zu integrieren und sie interagieren zu lassen.

Momentan zeigte sich die Entwicklung von Business Components noch relativ aufwendig. Dies ist unter anderem darauf zurückzuführen, dass es bisher kaum Erfahrung im Bereich der Anwendungsentwicklung für die Workplace-Plattform gibt und entsprechend wenig technische Informationen öffentlich verfügbar sind. Weiterhin bietet der WSAD zwar Unterstützung für die Entwicklung von Portlets und EJBs, jedoch nicht für Workplace-spezifische Elemente. So muss der Entwickler sich in aufwendiger Detailarbeit um die Realisierung der vorgegebenen Business Component Struktur kümmern und kann sich nicht allein auf die Implementierung der Funktionalität konzentrieren.

Für die Zukunft ist zu erwarten, dass die kommenden Versionen des WSAD stärkere Unterstützung für die Entwicklung von Workplace Anwendungen bieten werden. Außerdem ist für Mitte 2005 der IBM Workplace Designer angekündigt, der ähnlich wie der Domino Designer aufgebaut ist und durch Hilfsmittel zur Maskengestaltung sowie Unterstützung von Scriptsprachen eine einfache Anwendungsentwicklung ermöglicht[9].

[9] Nähere Informationen über den Workplace Designer findet man unter http://www.ibm.com/developerworks/workplace/workplace-designer.pdf

5. Referenzen

Green, Dale et al. (2002): The J2EE Tutorial, Sun Developer Network, 2003, Aus http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html am 13.03.2005

Groeger Hardy; Gorman, Mel (2004): IBM Workplace programming model, IBM DeveloperWorks, 2004, Aus http://www.ibm.com/developerworks/lotus/library/lwp-appdev2/ am 10.03.2005

Hepper, Stefan (2003): Comparing the JSR 168 Java Portlet Specification with the IBM Portlet API, IBM DeveloperWorks, 2003; Aus: www.ibm.com/developerworks/websphere/library/techarticles/0312_hepper/hepper.html am 10.03.2005

IBM (2004): Workplace API Toolkit User's Guide, Version 1.0 vom Oktober 2004, aus http://www.lotus.com/ldd/lwpapi am 10.03.2005

Rodriguez, Juan et al. (2004): IBM WebSphere Portal V5 A Guide for Portlet Application Development, IBM Redbook, 2004; Aus http://www.redbooks.ibm.com/redbooks/SG246076/ am 10.03.2005

Roy-Chowdhury, Amber et al. (2002): Using Click-to-Action to Provide User-Controlled Integration of Portlets, IBM DeveloperWorks, 2002; Aus: http://www.ibm.com/developerworks/websphere/library/techarticles/0212_roy/roy.html am 28. März 2005

6. Anhang

Eingesetzte Software

- IBM Workplace Services Express 2.0.1
    - Infos:
      http://www.lotus.com/products/product5.nsf/wdocs/workplaceservicesexpresshome
- IBM Worplace Server 2.5 beta mit IBM HTTP Server 2.0.42.1
- IBM WebSphere Studio Application Developer 5.1.2
- IBM WebSphere PortalToolkit 5.0.2.3
    - Download:
      http://www.ibm.com/software/info1/websphere/index.jsp?tab=products/portaltoolkit
- IBM Lotus Workplace Products API Toolkit 1.0
    - Download:
      http://www.lotus.com/ldd/lwpapi

Installation des RSSReaders

Für das Entwickeln und Ausführen von Business Components wird zunächst ein Workplace Server benötigt. Im Rahmen dieses Projektes wurde zunächst mit einer schlankeren Version des Servers namens IBM Workplace Services Express gearbeitet. Zu einem späteren Zeitpunkt wurde der Server neu installiert, dann mit einer Beta-Version des Workplace Servers 2.5.

Für die Entwicklung benötigt man eine Integrated Development Environment (IDE), die das Entwickeln von J2EE Anwendungen sowie von Portlets unterstützt. Wir beschreiben hier den Einsatz des WebSphere Studio Application Developer (WSAD) von IBM.

Installieren Sie den WSAD und anschließend das WebSphere Portal Toolkit. Bei beiden Installationen sind keine besonderen Einstellungenen vorzunehmen.

Kopieren Sie das Workplace API Toolkit (Ordner lwpapi10) an einen beliebigen Ort auf ihrer Festplatte (z.B. C:/Programme/WebSphere Studio/lwpapi10)

Im Folgenden wird die Installation der von uns erstellten Business Component “RSSReader” beschrieben. Die Installation der “RSSContent” Business Component verläuft analog. Da sich diese Dokumentation in erster Linie an Entwickler richtet, werden wir in einem ersten Schritt beschreiben, wie die Sourcen des RSSReaders in den WSAD importiert werden, um sie dann zu kompilieren und die entsprechenden .war und .ear Dateien für das Deployment auf den Servern zu generieren. Alternativ können auch die zur Verfügung gestellten Dateien

RSSReader.ear und RSSReaderPortlet.war benutzt werden, um sie direkt wie unter IV. und V. beschrieben auf die Server zu deployen.

I. Arbeiten mit der RSSReader Business Component im WebSphere Studio

1. Starten Sie WSAD und wählen Sie **Datei → Importieren → Vorhandenes Projekt in den Arbeitsbereich**
2. Klicken Sie auf **Durchsuchen** und wählen sie den Ordner RSSCollabComponentEJB
3. Klicken Sie **Fertigstellen**; das Projekt sollten nun in ihrem Arbeitsbereich angezeigt werden. Ignorieren Sie vorerst Kompilierungsfehler und nicht aufgelöste Abhängigkeiten.
4. Wiederholen Sie diesen Import-Vorgang für alle zum RSSReader gehörenden Projekte (RSSReaderAdapterEJB, RSSReaderEAR, RSSReaderPortlet, RSSServiceEJB)
5. Für das erfolgreiche Kompilieren der Projekte müssen zwei Klassenvariablen (LWPAPI_HOME und WAS_HOME) im WSAD angelegt werden. Wählen Sie **Fenster → Benutzervorgaben** und unter dem Punkt **Java** den Unterpunkt **Klassenpfadvariablen**. Legen Sie eine neue Variable an und geben Sie ihr den Namen LWPAPI_HOME. Die Variable sollte auf das Verzeichnis verweisen, in dem das Workplace API Toolkit liegt (z.B. C:/Programme/WebSphere Studio/lwpapi10).
6. Sie benötigen zum lokalen Kompilieren der Projekte einige jar-Dateien vom WebSphere Application Server und dem Workplace Server. Es empfiehlt sich, von diesen Dateien lokale Kopien anzulegen und eine entsprechende Klassenvariable zu setzen, um die Projekte fehlerfrei kompilieren zu können. Legen Sie auf ihrem Dateisystem einen Ordner an, der als Basisverzeichnis für diese Bibliotheken dient (z.B. C:\Programme\IBM). Legen Sie im WSAD eine neue Klassenvariable mit dem Namen WAS_HOME an, die auf dieses Verzeichnis verweist. Alle benötigten Dateien werden Ihnen mit der RSSReader Business Component im Ordner WAS_libraries zur Verfügung gestellt und müssen somit nicht mehr vom Server heruntergeladen werden. Kopieren Sie alle Unterordner (AppServer, PortalServer und WorkplaceServer) in Ihr WAS_HOME Verzeichnis.
7. Sollten Sie mit ihrer Installation des WSAD noch nie Portlets entwickelt haben, müssen Sie einige zusätzliche Klassenvariablen anlegen. Dies kann der WSAD allerdings automatisch für Sie übernehmen. Wählen Sie **Datei → Neu → Projekt → Portlet Entwicklung → Portletprojekt** und geben Sie einen beliebigen Namen für das Projekt an. Mit Fertig stellen legen Sie das Projekt an und WSAD generiert die nötigen

Variablen. Sie können das Projekt sowie das ggf. automatisch angelegte DefaultEAR sofort wieder löschen, die Klassenvariablen bleiben bestehen.

8. Sie sollten nun über das Menü **Projekt → Alles erneut erstellen** die Projekte kompilieren können. Es wird ggf. noch ein Fehler in der RSSReaderPortletView.jsp angezeigt, dieser kann allerdings ignoriert werden, da die benötigte Bibliothek später bei der Ausführung auf dem Server vorhanden ist.

II. Generieren der .ear Datei

1. Für das Deployment der Anwendung müssen Sie eine .ear Datei generieren, die alle Komponenten enthält, die auf dem WebSphere Application Server ausgeführt werden. Selektieren Sie zunächst das Projekt RSSCollabComponentEJB mit der rechten Maustaste und wählen Sie **Generieren → Implementierungs- und RMIC Code**. Selektieren Sie das Projekt in der Liste und klicken Sie Fertig stellen.
2. Wiederholen Sie diesen Vorgang für alle EJB-Projekte (d.h. noch für RSSReaderAdapterEJB, RSSServiceEJB)
3. Klicken Sie mit der rechten Maustaste auf das Projekt RSSReaderEAR und wählen Sie **Exportieren → EAR Datei**. Versichern Sie sich, dass als Unternehmensanwendungsprojekt RSSReaderEAR ausgewählt ist. Wählen Sie einen Zielordner unter Beibehaltung des vorgeschlagenen Dateinamens und exportieren Sie das Projekt mit **Fertig stellen**

III. Generieren der .war Datei

1. Klicken Sie mit der rechten Maustaste auf das RSSReaderPortlet Projekt und wählen Sie **Exportieren → WAR Datei**
2. Wählen Sie einen Zielordner unter Beibehaltung des vorgeschlagenen Dateinamens und exportieren Sie das Projekt mit **Fertig stellen**

IV. Deployment der RSSReader.ear auf den WebSphere Application Server

1. Loggen Sie sich als Administrator auf der Administratorkonsole des WebSphere Application Server ein. Dieser ist bei Standard-Installation des Servers über den Port 9044 unter dem Verzeichnis /admin erreichbar (z.B. https://pbwi2lwp.winfo2.uni-paderborn.de:9044/admin/logon.jsp)
2. Wählen Sie in der Navigation **Anwendungen → Neue Anwendung installieren**
3. Wählen Sie unter lokaler Pfad ihre RSSReader.ear Datei aus und klicken Sie **Weiter**

4. Belassen Sie die Standardeinstellungen unter „Vorbereitung der Anwendungsinstallation“ und klicken Sie **Weiter**
5. Belassen Sie bei den folgenden Schritten 1 bis 3 („Installationsoptionen angeben“, „JNDI-Namen für Beans angeben“ und „EJB-Referenzen zu Beans zuordnen“) die Standardeinstellungen und bestätigen Sie jeweils mit **Weiter.**
6. Stellen Sie in Schritt 4 („Module zu Anwendungsservern zuordnen“) sicher, dass bei allen EJB Modulen unter „Server“ der Portal Server (z.B. WebSphere_Portal) ausgewählt ist. Selektieren Sie dazu alle Module, wählen sie in der Listen den Portal Server und klicken Sie auf **anwenden**. Bestätigen Sie mit **Weiter**.
7. Belassen Sie bei den folgenden Schritten 5 und 6 („Sicherheitsaufgabenbereiche zu Benutzern/Gruppen zuordnen“ und „Vergewissern Sie sich, dass alle ungeschützten 2.0-Methoden die richtige Zugriffsschutzebene verwenden“) die Standardeinstellungen und bestätigen Sie jeweils mit **Weiter**.
8. Sie sehen unter Schritt 7 eine Zusammenfassung ihrer Angaben, bestätigen Sie diese mit **Fertig stellen**
9. **Speichern** Sie ihre Änderungen in der Master Konfiguration
10. Wählen Sie in der Navigation **Anwendungen → Enterprise Anwendungen**. In der Liste sollten nun die Anwendung RSSReaderEAR auftauchen. Selektieren Sie diese Anwendung und klicken Sie auf **Starten**.
11. Die Anwendung sollte nun auf dem Application Server laufen. Verlassen sie die Administratorkonsole.

V. Deployment der RSSReaderPortlet.war auf den Workplace Server

1. Die RSSReaderPortlet.war muss auf dem Portal Server deployed werden. Dies geschieht über das Administrationsinterface des Workplace Servers. Loggen Sie sich als Administrator auf dem Workplace Server über Port 80 im Verzeichnis /lwp/workplace ein (z.B. unter http://pbwi2lwp.winfo2.uni-paderborn.de/lwp/workplace)
2. Klicken Sie oben in der Navigation auf **Verwaltung**, dann den Reiter **Portlets** und Menüpunkt **Installieren**.
3. Wählen Sie die Datei RSSReaderPortlet.war aus und klicken Sie **Weiter**.
4. Bestätigen Sie nach dem Hochladen der Datei die Installation mit **Installieren**.
5. Bestätigen Sie die erfolgreiche Installation.

6. Kehren Sie über **Mein Workplace** auf ihre Arbeitsoberfläche zurück. Ziehen Sie auf der rechten Seite die Liste mit verfügbaren Portlets hervor, klicken Sie auf **Hinzufügen** und suchen Sie nach „RSS Reader“. Fügen sie das RSS Reader Portlet der Liste ihrer Portlets hinzu. Nun können Sie das Portlet per Drag und Drop auf ihren Arbeitsbereich ziehen und somit starten.
7. Damit alle Benutzer des Portals das Portlet nutzen können, gehen Sie wie folgt vor: Wählen Sie **Verwaltung → Zugriff → Ressourcenberechtigungen**.
8. Wählen Sie Portlets aus und suchen sie das RSSReader Portlet. Klicken Sie auf das Symbol unter **Zugriffsberechtigung zuordnen.**
9. Klicken Sie unter Benutzer (oder einer anderen Rechteklasse, die Sie den Benutzern des Portals einräumen möchten) auf das Symbol zu **Berechtigungsklasse bearbeiten**
10. Selektieren Sie **all authenticated portal users** und klicken Sie **OK**. Bestätigen Sie alle Eingabemasken mit **OK** bzw. **Fertig**, bis Sie wieder auf der Ebende Ressourcenberechtigungen sind. Nun können alle angemeldeten Benutzer das Portlet finden und ihrem Workplace hinzufügen.

Weitere Ressourcen

Im Folgenden möchten wir auf einige Quellen verweisen, die in der Dokumentation nicht explizit genannt wurden, aber dennoch wertvolle Ressourcen bei der Entwicklung von Business Components darstellen:

Monson, Philip et al. (2005): Building a Component for IBM Workplace, IBM Redpaper Draft Version vom 21.01.2005; Aus http://www.redbooks.ibm.com/redpieces/abstracts/redp3952.html

IBM Lotus Workplace Information Center: Dokumentationsbibliothek zu Workplace. Hauptsächlich Informationen zu Installation und Nutzung; Aus http://publib.boulder.ibm.com/infocenter/lwpphelp/

WebSphere Portal Information Center: Bibliothek zum IBM Portal Server. Informationen zur Portlet Entwicklung findet man im Menü unter „Portlet Development“; Aus http://publib.boulder.ibm.com/pvc/wp/502/ent/en/InfoCenter/

WebSphere Application Server V5.1 System Management and Configuration: IBM Redbook zum WebSphere Application Server. Nützlich für Informationen zu Server Konfiguration und Application Deployment; Aus http://www.redbooks.ibm.com/redbooks/SG246195/

WebSphere Portal API: API der Java Klassen, die vom Portal Server bereitgestellt werden; Aus http://www.ibm.com/developerworks/websphere/zones/portal/portlet/5.0api/WPS/

JavaDoc zum Projekt: Dokumentation der Java-Klassen, die mit den Sourcen zum Projekt zur Verfügung gestellt werden.