September 2010

ANWENDUNGS-INTEGRATION IN UNTERNEHMENSPORTALEN

Methoden und Herausforderungen

Juri Urbainczyk
A:gon Solutions GmbH

Abstract

Portale sind als Kommunikations- und Prozessplattformen heute unverzichtbar. Ihre Leistung entfalten sie jedoch erst durch die Integration von Diensten und Anwendungen. Dieses Dokument beschreibt die Grundprinzipien und die wesentlichen Methoden der Anwendungsintegration und stellt die Konsequenzen für die Anwendungen dar. Als Entscheidungshilfe für die Wahl einer Integrationsmethode wird eine Kriterienmatrix vorgestellt. Einen weiteren Schwerpunkt bildet das Thema Single-Sign-On, das eine Kernanforderung für Portale ist.

Überblick

Ein Portal ist eine Webanwendung, die den Benutzern strukturierten und personalisierten Zugriff auf die Applikationen, Dienste und Daten des Unternehmens ermöglicht [DEF]. Es handelt sich also – einfach gesprochen – um eine Infrastruktur zur Frontend-Anwendungsintegration. An dieser Stelle liegt eine wesentliche Unterscheidung zu EAI (Enterprise Application Integration), die eine Integration auf Ebene der Anwendungslogik anstrebt. Die Präsentation des Portals im Web setzt sich dabei aus Portlets [PORTLET] zusammen, kleinen fachlich und technisch definierten Einheiten, die zusammengenommen eine Portalseite bilden.

Das Portal selbst ist lediglich ein Container, eine Hülle, deren Wert sich durch die in ihm verfügbaren Ressourcen ergibt. Nur durch die Integration von Diensten und Anwendungen kann Mehrwert mit einem Portal generiert werden. Daher ist die Integration [AI] die wichtigste Aufgabenstellung beim Aufbau und bei der Pflege eines Portals. Aufwand und Komplexität der Anwendungsintegration werden jedoch immer wieder unterschätzt. Produkthersteller erwecken oft den Eindruck, dass diese Aufgabe durch das Portalprodukt bereits gelöst sei, da es entsprechende Werkzeuge mitbringe. Tatsächlich aber muss jedes Portalprojekt die Probleme der Anwendungsintegration für das jeweilige Umfeld erneut verstehen und lösen.

Portalprojekte können entscheidend beschleunigt und auch verbilligt werden, wenn die Anwendungsintegration vereinfacht und stärker standardisiert werden kann. Um das zu erreichen, ist eine gute Kenntnis der vorhandenen Methoden und notwendigen Aufgaben ebenso erforderlich, wie eine Analyse des aktuellen Umfelds im jeweils individuellen Projekt. Denn nicht jede Technologie ist für alle Anforderungen stets die richtige Wahl.

Daher beschreibt dieses Dokument die Ziele, Methoden und Aufgaben der Anwendungsintegration detailliert und versucht sie miteinander zu verknüpfen. Dabei basiert der Artikel auf Projekten zur Portaleinführung und Anwendungsintegration bei der BMW AG und der Zeppelin Baumaschinen GmbH, bei denen sowohl Kaufsoftware (TIBCO) als auch Opensource Software (Jetspeed, Liferay [LR] Versionen 4 und 5) und Eigenentwicklungen auf J2EE-Basis eingesetzt wurden.

Warum Anwendungsintegration?

Warum sollte man überhaupt Anwendungen in ein Portal integrieren? Im Allgemeinen wird ein Portalprojekt erst gestartet, wenn bereits eine umfangreiche Anwendungslandschaft existiert – und die funktioniert auch ohne Portal. Tatsächlich aber gibt es eine Reihe guter Gründe für eine Portalintegration, welche im Folgenden vorgestellt werden, wobei kein Anspruch auf Vollständigkeit besteht.

- Das Portal soll alle verteilten Anwendungen, Dienste und Daten an zentraler Stelle zusammenführen und einen eindeutigen, **gemeinsamen Einstieg** bieten. Aus diesem Ansatz leitet sich das Wort „Portal“ selbst ab.
- Mit einem Portal braucht der Benutzer sich nur einmal mit einem Login und einem Kennwort anzumelden und ist für alle integrierten Anwendungen authentisiert („**Single-Sign-On**“). Selbst für vormals offene Anwendungen gibt es nun einen **minimalen Zugriffsschutz**.
- Die **personalisierte Portalnavigation** zeigt nur die Ressourcen an, für die der Benutzer auch berechtigt ist. Dadurch lassen sich Aufgaben und Prozesse effizienter gestalten.
- Im Portal ist der Benutzer zu jedem Zeitpunkt eindeutig authentisiert. Das ermöglicht die Bereitstellung von individuell zugeschnittenen, **personalisierten Inhalten**. Beispielsweise kann nach dem Login eine Nachricht angezeigt werden, die nur für diesen Benutzer oder nur für Benutzer mit dieser speziellen Aufgabe relevant ist. Der Benutzer hat zudem die Möglichkeit die Oberfläche des Portals nach seinen eigenen Wünschen zu gestalten, also z.B. die Positionierung der Portlets im Portal zu verändern und für sich abzuspeichern.
- Die Anwendungen sind im Portal unter dem gleichen Submenü oder Reiter zu finden, so dass sich für den Benutzer fast den Eindruck einer **gemeinsamen Oberfläche und Navigation** ergibt. Dieser Eindruck lässt sich verstärken, indem man mehrere Anwendungen in verschiedenen Portlets auf der gleichen Portalseite anzeigt. Häufig bieten Anwendungen mehr als nur einen Einstieg (die sich im Allgemeinen durch unterschiedliche URLs repräsentieren). Diese Einstiegspunkte können im Portal aufgebrochen und dem fachlichen Prozess entsprechend sinnvoll angeordnet werden.
- Einzelne **Services kann das Portal zentral zur Verfügung** stellen. Sie werden dann von den integrierten Anwendungen über standardisierte Schnittstellen genutzt („Portalinfrastruktur“). Der Zugriff auf die dahinterliegende Infrastruktur, welche die benötigten Daten bereitstellt, wird vom Portal gekapselt. Die Attribute des Benutzers lassen sich z.B. beim Portal-Login im HTTP-Header ablegen und können dann von jeder Anwendung dort ausgelesen werden.
- Um den Prozess für die Benutzer durchgängiger zu gestalten, können bei der Portalintegration die einzelnen Anwendungen vom Look&Feel her aneinander angeglichen werden („**Re-Branding**“). Um dies zu erreichen sind jedoch komplexe Integrationsverfahren nötig (s.u.). Der dadurch generierte Aufwand ist durch diese Anforderung allein sicherlich nicht zu rechtfertigen.

- Portale bieten erweiterte Möglichkeiten, wie z.B. Portlet-Portlet-Kommunikation, wodurch eine Client-seitige Integration der Anwendungen untereinander ermöglicht wird. So lassen sich z.B. Daten unter den Anwendungen austauschen und Benutzer benachrichtigen. Auf diese Weise kann man das Portal in Richtung **Workflow-Management-System** ausbauen. Es sei jedoch angemerkt, dass dazu Entwicklungsaufwand notwendig ist, denn die Kommunikation ist nicht konfigurativ durch reine Portal-Bordmittel erreichbar (wie manche Werbung suggeriert).
- Viele Portalprodukte bringen bereits viele kleine Anwendungen als Portlets mit. Dazu gehören z.B. **Web 20 und Social Services sowie Document Management Systeme**. Diese sind leicht änderbar und somit flexibel an die individuellen Anforderungen anpassbar. Mit vergleichsweise einfachen Anpassungen lassen sich z.B. Qualitätssicherungsschritte einbauen.

Grundprinzipien der Anwendungsintegration

Unabhängig von den fachlichen Anforderungen und den technischen Möglichkeiten gibt es mehrere grundlegende Eigenschaften, welche von der Anwendungsintegration erfüllt werden müssen. Sie leiten sich aus den allgemeingültigen Anforderungen ab, den Aufwand – und damit die Kosten – möglichst gering zu halten, Folgeaufwand zu vermeiden und somit die Administration und die Weiterentwicklung aller beteiligten Systeme nicht mehr als notwendig zu erschweren. Folgende grundlegenden Eigenschaften werden gefordert:

1. Die Integration sollte nur zur einer *schwache Kopplung zwischen Applikation und Portal* führen. Eine starke Kopplung bedeutet eine hohe gegenseitige Abhängigkeit und würde daher die ungestörte Weiterentwicklung gefährden. Des Weiteren soll die Abhängigkeit vom verwendeten Portalprodukt gering gehalten werden, um einen späteren Wechsel offen zu halten. Ein gutes Indiz für schwache Kopplung ist es, wenn die integrierte Anwendung *sowohl im Portal als auch außerhalb* aufrufbar ist und sich in beiden Situationen gleich verhält. Obwohl es sinnvoll sein kann, den Endbenutzern nur noch den Zugriff über das Portal zu erlauben, muss die Anwendung auch ohne Portal lauffähig sein, allein um Entwicklung, Test und Wartung zu erlauben.
2. Am Portalprodukt sollten *möglichst wenige oder gar keine Änderungen* vorgenommen werden. Bei jedem Versionswechsel des Portalprodukts (und die sind mittelfristig unvermeidbar) wäre es nämlich zwingend, alle Änderungen nachzuziehen.
3. Die Lösung sollte *so einfach wie möglich* sein. Komplexität ist zu vermeiden, um die Integration einfach und flexibel durchführen zu können und die Administration handhabbar zu halten. Zudem wird die Fehlerwahrscheinlichkeit deutlich reduziert und der Aufwand für die Einarbeitung verringert.
4. Der initiale *Aufwand für die Integration ist so gering wie möglich* zu halten. Szenarien, die einen unrealistisch hohen Aufwand bei der Anpassung der Anwendungen fordern, sind nicht zukunftsfähig, da sie eine große Hürde für die Integration erzeugen.

5. Eine mehrfache Pflege von Daten und *Redundanz von Daten und Logik sollte vermieden werden*. Denn damit steigen die Fehleranfälligkeit und direkt auch der Aufwand bei der Administration.
6. Die *Last auf dem Server des Portals* darf durch die Integration nicht übermäßig ansteigen, um eine vernünftige Performance zu gewährleisten. Insbesondere ist es kritisch zu sehen, wenn alle Zugriffe der Clients vollständig über den Server laufen.
7. Der *Integrationsprozess* muss beschrieben werden und er muss im Rahmen des heutigen und zukünftigen Umfelds durchführbar sein. Das beste Integrationsverfahren nützt nichts, wenn keine Mitarbeiter zu Verfügung stehen, die es kennen und umsetzen können.
8. *Zukünftige Entwicklungen des Portalprodukts* müssen berücksichtigt werden. Einerseits kann man so unnötigen Aufwand bei der Implementierung vermeiden, wenn entsprechende Erweiterungen bereits vom Hersteller angekündigt werden. Andererseits kann man vermeiden, in eine Sackgasse zu geraten, wenn z.B. der Produkthersteller bestimmte Alternativen bereits heute ausschließt.
9. Das *Berechtigungskonzept* des Portals muss zur vorhandenen Infrastruktur und zu den bereits existierenden Prozessen beim Berechtigungsmanagement passen. Ist dies nicht der Fall, droht hoher Aufwand beim „Verbiegen“ der Möglichkeiten des Portals oder sogar das Scheitern der fachlichen Ziele des Portalprojekts.

Integrationsmethoden

Die Einführung und der Aufbau eines Portal führen zwangsläufig zu der Frage, wie man die vorhandenen (und die noch kommenden) Anwendungen technisch in das Portal integrieren kann. Und in jedem Fall muss eine auf das jeweils individuelle Umfeld zugeschnittene Lösung gefunden werden. Im Folgenden werden die bekannten Integrationsmethoden mit ihren Vor- und Nachteilen vorgestellt. Im Anschluss findet sich eine Entscheidungsmatrix für die Auswahl der passenden Methode.

Methode 1: Direkter Aufruf

Diese Methode wird auch häufig als *URL-Integration* oder *Integration by Link* bezeichnet. Hierbei wird aus dem Portalmenü oder aus einer Portalseite direkt die Anwendung aufgerufen. Der Klick auf den entsprechenden Menüeintrag im Portal führt dann zum Öffnen der Anwendung in einem neuen oder dem aktuellen Browserfenster.

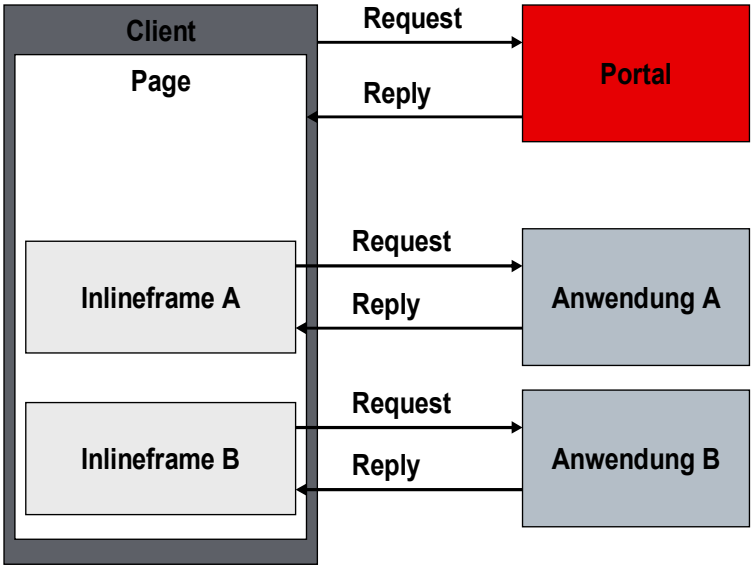

Abbildung 1: Inlineframe

Diese Integrationsform ist einfach und führt zu keinerlei Abhängigkeiten zwischen Portal und Anwendung. Die Anwendung muss lediglich durch eine URL aufrufbar sein. Da nach dem Aufruf der Anwendung die weitere Benutzerinteraktion direkt in der Anwendung geschieht, ist die Belastung des Servers, auf dem das Portalprodukt läuft, sehr gering. Viele Integrationsziele (z.B. Prozessunterstützung) sind mit dieser Methode jedoch nicht realisierbar. Für Anwendungen, die meist das volle maximierte Browserfenster benötigen (z.B. Reports), ist dies dennoch die Methode der Wahl.

Methode 2: Inlineframe

Viele Portalframeworks bieten IFRAME-Portlets für die Integration an. Das IFRAME-Portlet wird zur Ausführungszeit als Inlineframe in HTML gerendert [IF]. Mit einem Inlineframe kann man ein weiteres Dokument in ein vorhandenes einbetten. So haben Inlineframe eine feste Größe, enthalten aber einen eigenen Scrollbar, wodurch man beliebig umfangreiche Seiten innerhalb des Inlineframes darstellen kann. Was im Inlineframe angezeigt wird, ergibt sich durch dessen SRC Attribut, das eine URL enthält. In unserem Fall ist dies die Start-URL der zu integrierenden Anwendung.

Ein großer Vorteil dieser Methode ist, dass so die Anwendung auch optisch in den Rahmen des Portals eingebettet werden kann. Da der Inlineframe einen echten „virtuellen Browser“ zur Verfügung stellt, funktioniert das auch bei den meisten Anwendungen problemlos. Darüber hinaus ist es möglich, mehrere Inlineframes auf einer Seite darzustellen uns somit mehrere Anwendungen in einen gemeinsamen, optischen Kontext zu stellen (s. Abbildung 1). Anwendungen dürfen allerdings keinen Code enthalten, der eine Darstellung im gesamten Browserfenster erzwingt und damit den Kontext des Portals zerstören würde (z.B. _TOP als Ziel einer URL). Abgesehen von diesem Punkt kann eine Anwendung i.Allg. ohne Änderungen in den Inlineframe integriert werden. Weitere Abhängigkeiten zwischen Portal und Anwendung ergeben sich ebenfalls nicht. Allerdings gibt es auch keine weitere Kopplung, z.B. keine Kommunikation mit dem Portal über den Aufruf der initialen URL hinaus. Deshalb kann von einer einfachen Integrationsmethode gesprochen werden. Da das Rendering des Inlineframes im Browser (also auf dem Client) stattfindet, ist die Belastung des Servers ebenfalls sehr niedrig.

Die Inlineframes wurden erst recht spät durch Browser unterstützt, nämlich mit Internet Explorer 5.1 und Netscape 6.0. Daher weicht die Implementierung der Inlineframes auf unterschiedlichen Browsern leicht voneinander ab, was u.A. zu Problemen mit den Umlauten führen kann. Zu beachten ist auch, dass ein Neuladen der Portalseite (z.B. durch ein Maximieren des Portlets) auch ein Neuladen des Inlineframes erzwingt. In diesem Fall wird allerdings die hinterlegte Start-URL erneut geladen, was – je nach Anwendung – zu einem Kontext- und Datenverlust führen kann. Vor allem Seiten, die Applets enthalten, werden im Inlineframe wohl nicht funktionieren, da mit jedem Neuladen der Seite auch eine neue Instanz des Applets erzeugt wird. Timeouts sind ebenfalls ein Ärgernis für Inlineframes, da bei der Nutzung der Applikation im Frame ein Timeout im umgebenden Portal auftreten kann [OPT].

Methode 3: Web Clipping

Will man z.B. das Look&Feel der Anwendung im Portal verändern, muss man auf den generierten HTML-Code der Anwendung zugreifen. Solches ermöglicht die Integrationsmethode „Web Clipping“ [WC]. Dabei wird die URL der integrierten Anwendung bereits auf dem Server des Portals aufgerufen und das erzeugte HTML kann somit gefiltert und transformiert werden, bevor es an den Client weitergegeben wird.

Diese Methode bietet eine stärkere Integration als Inlineframes, da nun sowohl die GUI als auch die Daten (z.B. in Formularen eingegebene Werte) der Anwendung vom Portal manipuliert werden können. Auf diese Weise kann z.B. nur ein Ausschnitt der Anwendung im Portal dargestellt werden. Allerdings erkauft man sich auf dieser Weise einige Nachteile: Der gesamte Transformationsvorgang läuft auf dem Server des Portals ab (s. Abbildung 2). Somit läuft jede Kommunikation zwischen Client und Anwendung durch den Server.

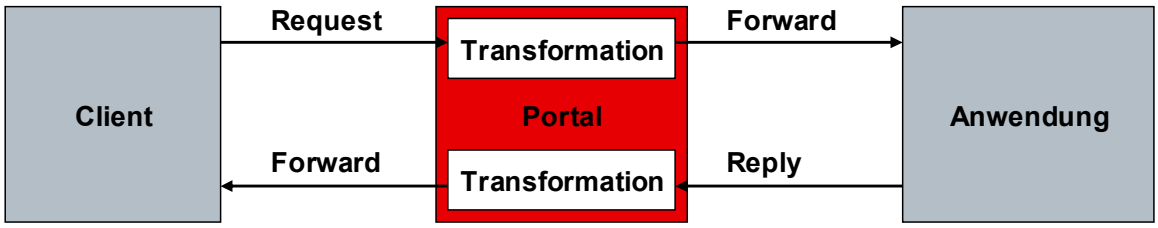

Abbildung 2: Server-Sider Integration

Die Last die auf diese Weise erzeugt wird ist nicht unerheblich und kann zu Performanzproblemen führen. Des Weiteren ist die Transformation des HTML nicht trivial, vor allem wenn der Code über einfache Listen und Formulare hinausgeht. Die Verwendung von <FRAME> Elementen wird meist nicht unterstützt. Auch intensive Nutzung von JavaScript kann die Funktionalität des Web Clipping beeinträchtigen. JavaScript wird nämlich auf dem Client nach dem Transformieren mit dem Web Clipping Portlet ausgeführt. Alle Inhalte, die durch solche Scripts zur Laufzeit erzeugt werden (z.B. dynamisches Aufbauen des HTML-Codes mit document.write oder per DOM), können durch das Clipping nicht kontrolliert werden. Dem Autor ist aktuell kein Web-Clipping-Portlet bekannt, das für eine beliebige Menge von Websites sinnvolle Ausgaben liefert. Webanwendungen, deren Verhalten bekannt und eventuell sogar kontrollierbar ist, können i.A. durchaus mit Web Clipping integriert werden.

Methode 4: Portal-Adapter

Ein Portal-Adapter ist eine Software-Komponente, die als Bindeglied zwischen einem (potenziell generischen) Portlet und der Anwendung dient. Das Portlet besitzt in diesem Architekturmuster eine prinzipiell einfache Struktur und bietet grundlegende Operationen auf bestimmte Datentypen an (z.B. CRUD = Create, Read, Update, Delete). Viele Portalframeworks stellen solche einfachen Portlets zur Verfügung, um Services anzusprechen und die Ergebnisse darzustellen. So lässt sich z.B. einfach eine Suchfunktion für eine Legacy-Anwendung in

das Portal integrieren. Der Portal-Adapter bildet die Operationen des Portlets auf Methoden bzw. Services der Anwendung ab.

Eine Voraussetzung dafür ist dass die Anwendung eine API (Schnittstelle) zur Verfügung stellt, mit der auf die wesentlichen Funktionen der Anwendung zugegriffen werden kann. Idealerweise handelt es sich dabei um eine serviceorientierte Schnittstelle auf Basis von Webservices (s. Abbildung 3). Bei der Integration wird nun nicht mehr die GUI der Anwendung in das Portal integriert, sondern Ihre Geschäftslogik.

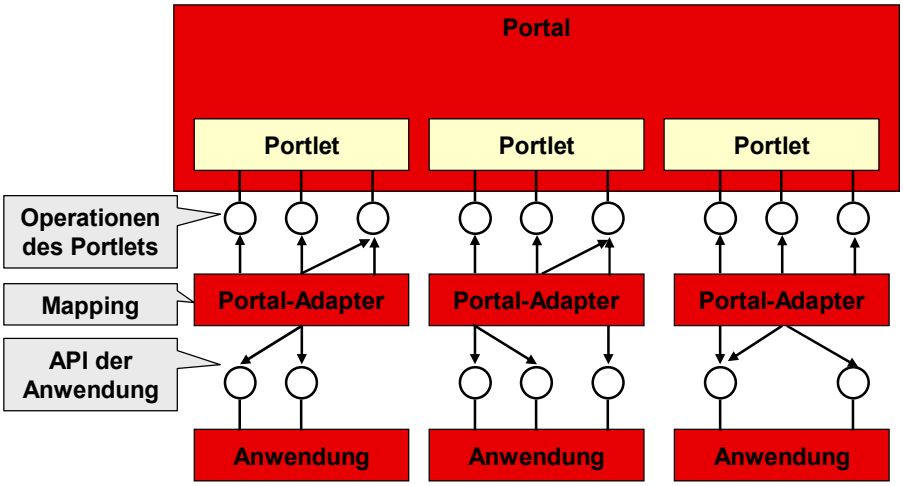

Abbildung 3: Portal-Adapter

Der Portal-Adapter wird so implementiert, dass er für verschiedene Schnittstellen sowohl der Anwendung als auch der verwendeten Portlets allein durch Konfiguration angepasst werden kann (z.B. Name der Methoden, Parameter, Typen, etc.). Zudem wird ein Mapping der unterschiedlichen Methoden durchgeführt.

So können unterschiedliche Anwendungen mit wenig Aufwand in das Portal integriert werden, sofern sie eine ausreichend konforme Schnittstelle aufweisen. Die Granularität der Schnittstellen von Anwendung und Portal dürfen nicht zu sehr abweichen. Hat man mehrere Anwendungen zu integrieren, die auf derselben Architektur basieren, ist der Portal-Adapter sicherlich ein erfolgversprechender Ansatz – solange eine Oberfläche mit wenigen, einfachen Operationen genügt. Diese Integrationsmethode empfiehlt sich vor allem für Anwendungen, die den Großteil ihrer Geschäftslogik auf dem Backend realisieren, so dass eine ausgefeilte Oberfläche nicht immer notwendig ist. Benötigt man allerdings eine komplexe GUI, muss man eventuell zum Inlineframe oder zur Portalisierung (s. Methode 5: Portalisierung) greifen.

Methode 5: Portalisierung

Nahezu alle Portalframeworks unterstützen die im Jahr 2003 veröffentlichte Portlet-Spezifikation JSR 168 [JSR168]. Eine Webanwendung, die auf Basis dieser Spezifikation gebaut wurde, kann mit geringem Aufwand in ein Portal integriert werden und wird dort direkt als

Portlet dargestellt. Es wird eine GUI-Integration ähnlich dem Inlineframe erreicht, ohne jedoch potenziellen Kontextverlust hinnehmen zu müssen. Außerdem erhält ein JSR 168 Portlet direkt das Look&Feel des Portals, so dass an dieser Stelle keine Änderungen mehr nötig sind. Des Weiteren spezifiziert der Standard ein Statusmodell für Portlets, so dass grundlegende Operationen, wie Maximieren, Minimieren und Verschieben bereits unterstützt werden. Daher kann es sinnvoll sein, eine Anwendung auf JSR 168 umzustellen – hier „Portalisierung“ genannt. Gerade bei J2EE-Anwendungen kann dies einfach sein, da die Portlet-Spezifikation sich stark an die bewährte Servlet-Spezifikation anlehnt. Bei der Implementierung von neuen Anwendungen sollte daher bereits auf dieses Architekturmerkmal geachtet werden.

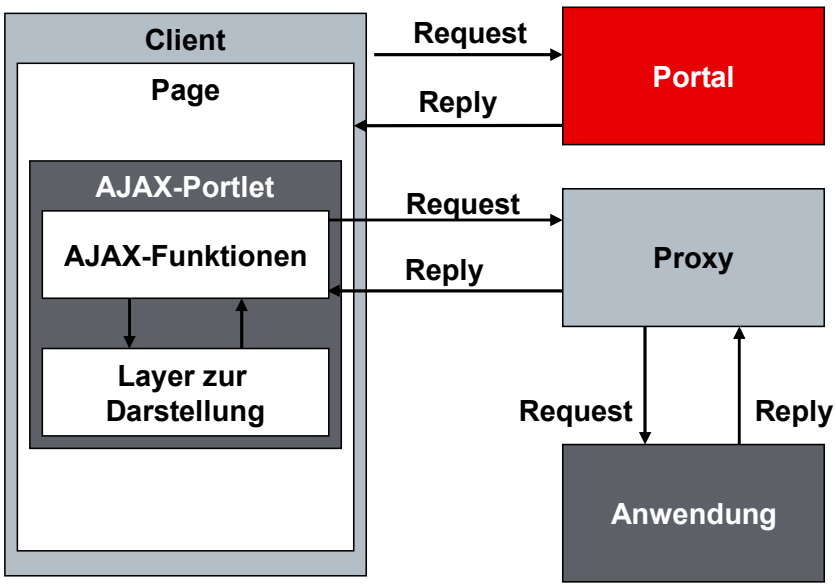

Abbildung 4: Integration mit AJAX

Allerdings wird auch durch JSR 168 keine weitergehende Integration, wie z.B. Workflow Management, unterstützt. Die Version 2.0 der Portlet-Spezifikation wurde im Juni 2008 verabschiedet [JSR286]. Dort stehen vor allem die Themen der Portlet-Portlet-Kommunikation wie Event-Mechanismen, Austausch von Session-Data und Render-Parametern zwischen verschiedenen Portlets im Vordergrund [IBM]. Der Portlet-Container von Liferay unterstützt in der aktuellen Version bereits die 2.0 Version [LR].

Methode 6: AJAX

Alternativ ist es auch denkbar, die Applikationen nicht im Server des Portals sondern auf dem Client zu integrieren. Durch die rapide Entwicklung der AJAX-Technologie [AJAX] in den letzten Jahren wird ein solches Vorgehen nun möglich.

Das kann man sich in etwa so vorstellen: Der Server generiert eine HTML-Seite, die entsprechenden JavaScript-Code und AJAX-Funktionen enthält. Beim Rendern dieser Seite im Browser wird der AJAX-Code aktiv und setzt für alle darzustellenden Anwendungen (möglicherweise parallel) Requests ab. Auch AJAX-Requests dürfen nur zur gleichen Domain gehen, von der die aktuelle Seite stammt. Wenn der Server der Anwendung nicht in der gleichen Domain steht, kann ein Proxa eingesetzt werden (s Abbildung 6). Nach Eintreffen der Antworten, fügen die AJAX-Funktionen das jeweils notwendige HTML passend in die aktuelle Seite ein. Alle Links im HTML Code werden durch die AJAX-Funktionen vorher so geändert,

dass bei ihrer Ausführung zuerst eine AJAX-Methode aufgerufen wird, die wiederum den jeweiligen Link vom Server der Anwendung asynchron anfordert. Damit das funktioniert muss die Anwendung „friendly code“ liefern, d.h. das erzeugte HTML muss entsprechend gut transformierbar sein (z.B. darf es keine relativen URLs enthalten).

Diese Methode hat verschiedene Vorteile:

- Da beim Zugriff auf die Anwendungen der Server des Portals nicht durchlaufen wird, bleibt dessen Belastung gering
- Da ein direkter programmatischer Zugriff auf das von an Applikationen erzeugte HTML besteht, ist analog zum Web Clipping eine Transformation denkbar (um z.B. das Layout der GUI dem Portal anzupassen).

Methode 7: Lokale Integration (Applet)

Die Integration von lokalen Anwendungen auf dem aktuellen PC des Benutzers (z.B. Word und Excel) stellt einen Sonderfall dar. Dazu kann z.B. ein Applet verwendet werden, das mit den Aufrufpfaden und Parametern der lokalen Anwendungen konfiguriert wird. Sobald der Benutzer auf einen Menüeintrag des Portals klickt, wird im Hintergrund das Applet (z.B. per JavaScript) angesprochen. Das Applet wiederum ruft dann die lokale Anwendung über einen Betriebssystemsaufruf auf (s. Abbildung 5). Dazu muss das Applet allerdings entsprechende Rechte auf dem lokalen PC besitzen. Schon aufgrund der damit verbundenen Sicherheitsthematik ist dieses Integrationsmodell daher für Internet-Portale keine Option. In abgesicherten Intranets dagegen kann dieses Verfahren durchaus angewendet werden. Der Grad der Integration kann stark unterschiedlich sein. Vom einfachen Aufruf bis zur vollständigen Integration über die entsprechenden Schnittstellen der Anwendung ist alles möglich. So kann z.B. ein Excel in einen Layer der Portalseite eingebunden werden, während die Werte über das Applet wieder in das Portal zurückgeführt werden.

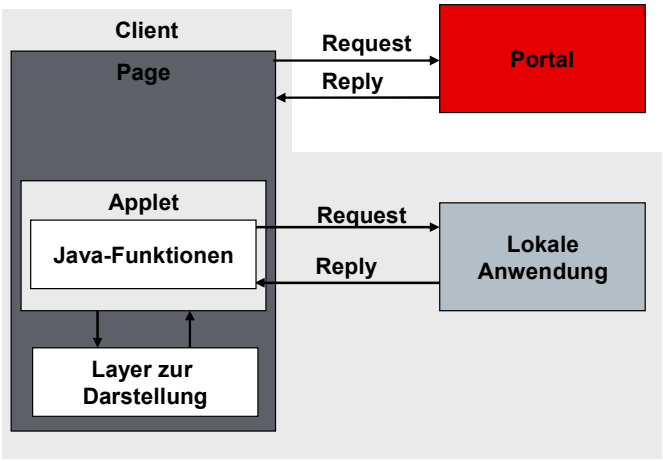

Abbildung 5: Integration mit Applet

Entscheidungsmatrix

Die gerade vorgestellten Integrationsmethoden können nun gegen die o.a. Ziele und Grundprinzipien der Anwendungsintegration geprüft werden. Kriterien die nicht trennscharf sind, die also nicht sehr von Eigenschaften der Integrationsmethode als eher von Eigenschaften der

Anwendung oder anderer Infrastruktur abhängen, werden nicht bewertet. Das Ergebnis ist folgende Matrix, die als Entscheidungshilfe bei der Bewertung und bei der Auswahl von Integrationsmethoden fungieren kann. Ein „-„ bedeutet, dass hier keine Entscheidung getroffen werden konnte.

Kriterium / Methode	Schwache Kopplung	Einfachheit	Geringe Last	Wenig Aufwand	Angleichung Look&Feel	Prozess-Mgmt.	Beibehaltung App-Kontext
Direkter Aufruf	ja	ja	ja	Ja	nein	nein	ja
Inlineframe	ja	ja	ja	Ja	nein	Nein	nein
Webclipping	nein	nein	nein	Nein	ja	ja	ja
Portal-Adapter	nein	ja	nein	-	ja	ja	ja
Portalisierung	nein	-	nein	Nein	ja	ja	ja
AJAX	ja	ja	ja	Ja	-	-	Ja
Applet	ja	variabel	ja	variabel	nein	variabel	nein

Eigenschaften der Anwendungen

Die Qualität der Anwendungsintegration hängt nicht nur vom Portalprodukt und der verwendeten Integrationsmethode ab. Da Portal und Anwendung technisch interagieren müssen, sind auch die technischen Eigenschaften der Anwendung wichtige Faktoren. Es lassen sich die folgenden Punkte nennen, die großen Einfluss auf die Integrierbarkeit der Anwendung haben:

- Die Anwendung muss über eine (oder mehrere) URLs aufrufbar sein. Diese Eigenschaft bedingt, dass es sich um eine Webanwendung handelt, bzw. dass die Anwendung über eine Web-GUI verfügt. Mehrere URLs erleichtern die Umsetzung von Workflow-Themen, bei denen Teile der Anwendung zu verschiedenen Zeiten im Workflow eingesetzt werden.

- Die Anwendung muss in einem Frame sauber darstellbar sein. Das bedeutet, dass sie z.B. nicht das aktuelle BrowserTop-Frames des Browsers manipulieren darf, was dazu führen würde, dass der Benutzer beim Aufruf der Anwendung das Portal verlässt. Insbesondere dynamischer HTML-Code, wie er z.B. mit JavaScript erzeugt werden kann, ist hier besonders fehleranfällig.

- Die Anwendung sollte Elemente der eigenen Logik unterdrücken können. Insbesondere ist es wichtig, das eigene Login und die damit verbundene Authentisierung des Benutzers an das Portal oder eine dritte Software delegieren zu können. Das gleiche gilt für das Logout, das nur noch aus dem Portal aufgerufen werden sollte.

- Die Anwendung sollte über den Aufruf einer URL ein Logout ausführen können. Diese URL würde aufgerufen, wenn der Benutzer sich aus dem Portal – und damit aus dem

Single-Sign-On abmeldet (s. untenstehender Abschnitt zu diesem Thema).

- Die Anwendung sollte in der Lage sein, Informationen aus der HTTP-Session entgegenzunehmen. Dies ist insbesondere wichtig, um den Namen des Nutzers von einer Single-Sign-On-Software zu übernehmen.

- Die Anwendung muss in der Lage sein, ein Mapping von Rollen zu den Einstiegspunkten (URLs) zu definieren. Auf diese Weise lassen sich die Einstiegspunkte der Anwendung dann im Portal über Rollen personalisieren.

- Der Session-Timeout der Anwendung muss einstellbar sein. Es dürfen keine Benutzer-Sessions von integrierten Anwendungen beendet werden, bevor sich der Benutzer aktiv aus dem Portal abmeldet. Alternativ dürfen die Benutzer-Sessions erst dann enden, wenn auch die Session des Portals (bzw. der Single-Sign-On-Software) abläuft.

Single-Sign-On und Single-Sign-Off

Eine der grundlegenden Anforderungen an ein Portal ist das Single-Sign-On (SSO): Die Logins aller Anwendungen sollen so synchronisieret werden dass der Benutzer sich nur noch beim ersten Zugriff auf das Portal oder eine integrierte Ressource *authentisieren* muss [SSO]. Anschließend wird kein weiterer Login mehr verlangt, so dass für den Benutzer der zugehörige Aufwand für Logins und Verwaltung der Passworte entfällt. SSO geht aber über eine reine Passwort-Synchronisation hinaus, da tatsächlich nur noch eine Abfrage des Passworts stattfindet.

Eine oft geforderte Eigenschaft des SSO ist, dass es *symmetrisch* sein soll. D.h. unabhängig davon, ob der Benutzer zuerst direkt das Portal oder eine integrierte Anwendung aufruft, soll in jedem Fall derselbe Login-Mechanismus greifen. Nach dem Login soll der Authentisierungszustand stets der gleiche sein, unabhängig davon, auf welchem Weg der Login erreich wurde. Diese Anforderung wird meist durch einen SSO-Verbund („circle of trust“) realisiert [CIRCLE]. Die Anwendungen und das Portal sind gleichberechtigte Partner – auch das Portal ist nur eine Anwendung unter vielen im SSO-Verbund. Ein Benutzer wird immer für den gesamten Verbund authentisiert. Welche Berechtigungen der Benutzer dann im Verbund besitzt, ist eine Frage der *Autorisierung*, welche von den Anwendungen bzw. vom Portal getrennt implementiert werden muss. Wird eine Anwendung in mehr als ein Portal integriert (also in mehrere SSO-Verbünde), muss definiert werden, welcher Verbund der jeweils *führende* ist. Es ist denkbar, dass der Benutzer durch das Verwenden der Anwendung gleich in beiden Verbünden angemeldet wird, was allerdings Passwort-Synchronisation zwischen den beiden Verbünden voraussetzt. Man muss sich aber immer für eine konkrete Login-Maske entscheiden, denn die kann je nach Verbund eine andere Funktionalität zur Verfügung stellen (z.B. zusätzlich Auswahl der Sprache etc.).

Viele Portalprodukte bringen eigene, proprietäre Single-Sign-On-Lösungen mit. Da in einem SSO-Verbund das Portal keine besondere Stellung mehr einnimmt, kann aber genauso gut eine dritte Software, die für diese Aufgabe spezialisiert ist, verwendet werden. Auch im Open-source-Umfeld gibt es bereits SSO-Produkte, z.B. CAS [CAS], so dass die Anbindung an existierende Anwendungen und das Portal einfach umsetzbar ist. Da es aktuell keinen übergreifenden Standard für SSO-Software gibt, legt man sich mit der Entscheidung auf jeden Fall für ein Produkt fest und muss im Migrationsfall mit entsprechendem Aufwand rechnen.

Typischerweise wird der SSO-Verbund dabei so realisiert, dass der Benutzer beim ersten Login ein Ticket bekommt (eine eindeutige elektronische Zeichenfolge), das beim Zugriff auf weitere Ressourcen im Verbund erneut validiert werden muss. Sowohl die Validierung des Tickets als auch die Durchführung des Logins sollten an die SSO-Software delegiert werden, um eine Aufweichung der Sicherheit durch Mehrfachimplementierung zu verhindern. Zu beachten ist, dass die Parameter für diese Delegation (also z.B. die URL für die SSO-Software im Releaseprozess der Anwendungen berücksichtigt werden muss, da auf anderen Umgebungen voraussichtlich andere Server zum Einsatz kommen. Die Architektur und die Betriebsaspekte (Ausfallsicherheit, Backup, etc.) müssen genau durchdacht werden, da SSO-Systeme eine hohe Kritikalität besitzen. Fällt nämlich das SSO-System aus, kann keine der integrierten Anwendungen mehr erreicht werden, was gravierende Folgen für das betroffene Unternehmen haben kann.

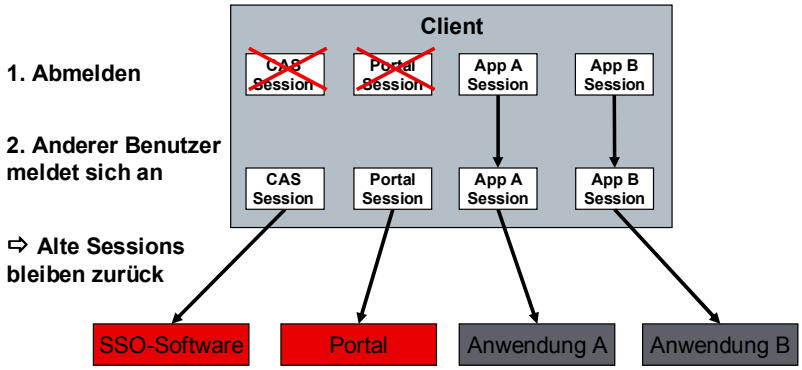

Abbildung 6: Portal ohne Single-Sign-Off

Häufig wird übersehen, dass neben dem Single-Sign-On auch ein Single-Sign-Off notwendig ist [OFF]. Denn jede Anwendung legt für einen Benutzer eine Session an (sofern sie nicht zustandslos ist, was selten der Fall ist). Diese Session muss beim Abmelden des Benutzers gelöscht werden, damit es nicht möglich ist, dass ein anderer Benutzer sich seine Rechte erschleichen kann (s. Abbildung 6). Jede Anwendung, die der Benutzer nach dem Login in den SSO-Verbund aufruft, legt eine Session für ihn an, so dass im Moment des Logout diverse Sessions im Hintergrund für ihn existieren. Will man nicht nach jedem Portal-Logout den Browser schließen, müssen alle diese Sessions nun gelöscht werden.

Dieses Problem ist nur mit hohem Aufwand serverseitig lösbar, da die Sessions oft durch Cookies clientseitig gespeichert werden. Und Cookies dürfen nur zu dem Server, von dem sie

angelegt wurden, zurückgeschickt werden. Also kann kein anderer „zentraler“ Server eine Übersicht über die aktiven Sessions des Benutzers erhalten. Alternativ kann man Single-Sign-Off auch so realisieren, dass alle integrierten Anwendungen jeweils eine URL zur Verfügung stellen, mit der die aktuelle Session gelöscht wird. Diese URLs werden dann beim Logout aufgerufen. Beim Klick auf das „Logout“ im Portal würde der Benutzer dann sowohl aus der SSO-Anwendung (Ticket) als auch aus allen integrierten Anwendungen inklusive dem Portal abgemeldet. Eine etwas ungewöhnliche aber nichtsdestotrotz wirksame Implementierung dieses Algorithmus ruft nach dem Logout eine HTML-Seite auf, die den Aufruf der SSO-Urls der Anwendungen durch Inlineframes realisiert. Der HTML-Code könnte z.B. so aussehen:

```
<DIV STYLE=“visibility:hidden“>
        <IFRAME SRC=”http://app1/logout”/>
</DIV>
<DIV STYLE=“visibility:hidden“>
        <IFRAME SRC=”http://app2/logout”/>
</DIV>
```

Ausblick

Die Anwendungsintegration ist das wichtigste und zugleich das am meisten unterschätzte Thema bei Portaleinführungen. Sie treibt vor allem langfristig die Aufwände im Portalprojekt. Nur mit richtigen Konzepten und dem Verständnis für die grundlegenden Zusammenhänge kann die Integration beherrschbar bleiben. Die in diesem Artikel vorgestellten Methoden sind dafür nur ein Ausgangspunkt.

Es ist an der Zeit einen Standard zu etablieren, wie integrationsrelevante Daten abgelegt werden sollen (URLs, Rollen, Portlet-Parameter, etc.). Damit wäre es wesentlich einfacher, zwischen verschiedenen Portalprodukten zu wechseln. Es wird jedoch in absehbarer Zukunft weiterhin nötig sein, für das jeweils vorhandene Umfeld eine angepasste und geeignete Lösung zu suchen.

Literatur

[AI] BEA: Introduction to Application Integration http://edocs.bea.com/wli/docs70/aiover/index.htm

[AJAX] *AJAX: Getting started*, http://developer.mozilla.org/de/docs/AJAX:Getting_Started

[CAS] *JA-SIG Central Authentication Service* http://www.ja-sig.org/products/cas/

[CIRCLE] SSO im Web
http://www.uni-muenster.de/ZIV/inforum/2007-1/a18.html

[DEF] *Was ist ein Portal? Definition und Einsatz von Unternehmensportalen, Thorsten Gurzki, Fraunho-*

fer Institut für Arbeitswirtschaft und Organisation; Anja Kirchhofi, Fraunhofer Institut für Arbeitswirtschaft und Organisation; Henning Hinderer, Fraunhofer Institut für Arbeitswirtschaft und Organisation; Joannis Vlachakis, Fraunhofer Institut für Arbeitswirtschaft und Organisation, http://www.gurzki.de/publications/padem/Was%20ist%20ein%20Portal/

[IBM] *What comes next in the Portlet Specification V 2.0 with JSR 286* , Stefan Hepper, IBM, http://www.agilejava.com/downloads/TS-3627.pdf

IF] IFRAME http://de.wikipedia.org/wiki/Inlineframe

[JS] *Jetspeed 2 Enterprise Portal*, http://portals.apache.org/jetspeed-2/

[JSR168] *JSR-000168 Portlet Specification* http://jcp.org/aboutJava/communityprocess/review/jsr168/

[JSR286] JSR 286 http://jcp.org/en/jsr/detail?id=286

[LR] *Liferay.com Web Site*, http://www.liferay.com/web/guest/home

[OFF] *Single Sign on and Single Sign Off in a non homogeneous Portal front ended environment.* Alan Berg, Bas Toeter, Central Computing Services, Universiteit van Amsterdam, The Netherlands. http://www.mc.manchester.ac.uk/eunis2005/medialibrary/papers/paper_104.pdf

[OPT] *Options for rapid integration of Web applications into WebSphere Portal,* http://www.ibm.com/developerworks/websphere/library/techarticles/0607_boezeman/0607_boezeman.html, Richard Gornitzky, John Boezeman, IBM, Juli 2006

[PORTLET] Zhe Wang *No Portlet is an Island* http://websphere.sys-con.com/read/43383.htm

[SSO] SSO http://www.securitymanager.de/magazin/artikel_996_single_sign_on_komfort_fuer_den_benutzer_oder.html

[WC] *The web clipping portlet* http://www.portletbridge.org/index.html

Der Autor

Seit ihm sein Vater Lochkarten zum Spielen mitbrachte, gilt die Leidenschaft von Juri Urbainczyk Computern und ihrer Software. Erste „Gehversuche“ gab es auf einem Commodore C64 und schon in der Studienzeit arbeitete er freiberuflich in der IT-Branche. Nach dem Abschluss seines Studiums 1993 arbeitete Herr Urbainczyk zunächst als Softwareentwickler und lernte so die Sprachen C++ und Java kennen. Seit 1997 ist Herr Urbainczyk als Berater u.a. für Kunden wie BMW, Deutsche Bahn und Deutsche Lufthansa als Anforderungsmanager und Projektleiter tätig. Seine weiteren Schwerpunkte sind Portal-Architekturen, Rich Internet Applications und änderbare Software-Architekturen. Seit 2009 ist Herr Urbainczyk Projektmanager und Bereichsleiter bei der A:gon Solutions GmbH in Eschborn. Die Freizeit verbringt Herr Urbainczyk mit seiner Familie in der Wetterau oder produziert und genießt elektronische Musik.

Email: juri.urbainczyk@agon-solutions.de
Web: https://www.xing.com/profile/Juri_Urbainczyk

A:gon Solutions GmbH

Die A:gon Solutions GmbH, 2004 gegründet, ist ein unabhängiges IT-Dienstleistungsunternehmen mit Firmensitz in Eschborn bei Frankfurt und weiteren Standorten in Hamburg und Berlin. Das branchenübergreifende Dienstleistungsportfolio von A:gon umfasst das „A:gon-proven IT-Consulting“, eine bewährte, herstellerneutrale IT-Beratung; sowie die „A:gon-tailored IT-Solutions“, zu denen maßgeschneiderte, individuelle Softwareentwicklung, passgenaue Softwareintegration und effiziente Business Intelligence Lösungen gehören. A:gon stellt sich mit professionellem Projektmanagement, proaktivem Anforderungsmanagement und änderbaren Softwarearchitekturen auf die individuellen Bedürfnisse seiner Kunden ein. In ausgewählten Branchen wie Banken, Versicherungen, Aviation und Health Care bietet A:gon gemeinsam mit seinen Partnern Lösungen, die auf fundiertem Geschäftsprozess-Know-how beruhen und speziell auf die jeweilige Branche zugeschnitten sind: die „A:gon-tailored Business Solutions“. Die plattformübergreifende technologische Kompetenz bei A:gon reicht von klassischen Mainframe-Architekturen bis hin zu modernen Java/JEE Web- und Portal-Architekturen. Zu den Referenzkunden von A:gon gehören unter anderen die AOK Berlin-Brandenburg, die Commerzbank, die Deutsche Bank, die Deutsche Bank Bauspar, die Deutsche Börse, die Finanz Informatik und die Deutsche Lufthansa.

Frankfurter Strasse 71-75
D-65760 Eschborn
Telefon : +49 6196 80269 0
Telefax : +49 6196 80269 11
http://www.agon-solutions.de

Handelsregister Frankfurt HRB 58185
St.-Nr. 4022826171
Geschäftsführer: Udo Peters