



Services mit einem **kontextbezogenen Gesicht**

Das Web 2.0 mit Ajax, REST oder RSS und neue RIA-Technologien wie JavaFX, AIR von Adobe oder Silverlight von Microsoft eröffnen neue Möglichkeiten, die Benutzeroberflächen und die Benutzerinteraktionen mit Services auf einer Webplattform zu gestalten.

von Marc Steger und Christian Kappert

Die Konstruktion von Benutzeroberflächen zur Interaktion mit einem oder mehreren Services erfolgt entweder durch die Nutzung von eher kontextfreien Widgets oder durch die Bereitstellung einer kontextspezifischen, bedienfertigen GUI. Der Benutzer fordert im zunehmenden Maße einen hohen Bedienkomfort bei diesen Service-Interaktionsvarianten. Insbesondere

die Ausprägung des Service Clients als Rich Internet Application (RIA) bietet hier ein großes Potenzial, indem sie die Vorteile von Webanwendungen mit denen von Desktopanwendungen verbindet: „reichhaltige“, clientseitige Präsentations- und Interaktionsfähigkeiten kombiniert mit dem Zugriff und der Aktualisierung über das Internet. Möglich wird dies dank der heutigen

weiten Verbreitung von breitbandigen Internetzugängen und clientseitigen Präsentations- und Kommunikations-

Anmerkung

Dieser Beitrag ergänzt den Artikel von Marc Steger und Christian Kappert: User-facing SOA, Java Magazin 03.2008, S. 66–75.

technologien wie Ajax, JavaFX, Flash/Flex/AIR oder Silverlight, deren Profile wir im Folgenden vorstellen.

Ajax Frameworks

Die Ajax Frameworks unterstützen Cross-Plattform- und Cross-Browserfunktionalitäten, indem sie dem Entwickler eine plattform- und browserneutrale Abstraktionsschicht zur Entwicklung der RIAs anbieten. Diese Abstraktionsschicht wird auf Basis von XHTML/DOM, CSS und JavaScript realisiert.

Die asynchrone Kommunikation erfolgt mittels XMLHttpRequest und in der Regel XML bzw. JSON. Viele Frameworks bieten darüber hinaus komplexe GUI-Komponenten mit Rich-Client-Funktionalitäten an. Leider gibt es (bisher) nicht *den* Standard für diese Abstraktionsschicht, sodass man es immer mit einem speziellen Framework und speziellen Schnittstellen zu tun hat.

Für Erleichterungen und eine verstärkte Standardisierung bei der Entwicklung von Ajax-Anwendungen will hier die OpenAjax Alliance sorgen, ein Konsortium aus namhaften Herstellern [1].

Die Frameworks lassen sich grundsätzlich in clientseitige und serverseitige Frameworks unterteilen. Clientseitige Frameworks sind in der Regel JavaScript-Bibliotheken wie Dojo oder Prototype/Scriptaculous. Serverseitige Frameworks, wie Rich Faces (AJAX4JSF) oder jMaki im Java-Umfeld, unterstützen die Konstruktion von MVC2-basierten Webanwendungen.

Runtime

Ajax-Anwendungen setzen als Laufzeitumgebung lediglich einen Webbrowser voraus. Plug-ins sind nicht notwendig. Jedoch muss die eingesetzte Browserversion XML und XMLHttpRequests unterstützen. Dies ist bei allen aktuellen Webbrowsern der Fall.

Installation und Versionsmanagement

Da die Bibliotheken mit der Webseite über das Netz geladen werden, gibt es keine speziellen Mechanismen für die

Installation und das Versionsmanagement.

Integration von Services

Die Kommunikation mit Services erfolgt allein per XMLHttpRequest. Insofern sind RESTful-Services bevorzugte und WS-Services mögliche Wahl.

Content Bridges

Die RIAs basieren auf XHTML, CSS und JavaScript und können in der Regel andere RIA-Komponenten wie Flash einbetten. Dies erfordert dann die Installation von Plug-ins. In welchem Umfang man auf eingebettete Inhalte zugreifen kann, hängt von der Unterstützung der entsprechenden Plug-ins ab.

JavaFX

JavaFX (JFX) ist eine neue Produktfamilie von Sun zur Adressierung des RIA-Marktes. Aktuelle Mitglieder der Produktfamilie sind [2]:

- *JavaFX Script* ist eine Skripting-Sprache (für GUI-Designer) zur Entwicklung von auf Java-Technologie basierten RIAs.
- *JavaFX Mobile* ist eine komplett neue Ausführungsumgebung (Application Stack) für mobile Endgeräte. Die Umgebung basiert auf Java SE mit Support von Java-ME-Applikationen.

JavaFX ist ein völlig neues Produkt und JavaFX Script eine völlig neue Sprache, sodass noch keine Entwickler-Community existiert. Zudem ist die Tool-Unterstützung insbesondere für Content Designer noch sehr spärlich. Die Tatsache jedoch, dass JavaFX sich nahtlos mit Java integrieren lässt und die Java-Plattform als Runtime besitzt, kann insbesondere im Mobile- und Embedded-Bereich ein nicht zu unterschätzender Vorteil sein. Sollte Sun die Hersteller von mobilen Endgeräten dazu bringen, die JavaFX-Mobile-Plattform so zu unterstützen, wie die bisherige Java-ME-Plattform, so wäre Sun mit einem Verbreitungsgrad von um die 90 % konkurrenzlos. Sun muss es aber auch schaffen, die große Java-Entwickler-Community von JavaFX zu

überzeugen. Laut Quickvote vom Java Magazin (9.2007) glauben immerhin 34 %, dass sich JavaFX auf dem RIA-Markt durchsetzen könnte. Erste Schritte im Desktopsegment im Vergleich zu Adobe (Flash/AIR) aufzuholen, werden mit der kommenden schlanken Consumer JRE (siehe Runtime) unternommen. JavaFX legt eine Entwicklung von Anwendungen nach dem MVP-Muster nahe.

Runtime

JavaFX-Anwendungen können grundsätzlich auf allen Plattformen mit Java Runtime ausgeführt werden. Konkret sind diese Desktopsysteme mit Java SE 6 (plus JavaFX-Runtime-Libs) und Mobile-/Embedded-Systeme mit JavaFX Mobile. Sun stellt die so genannte „Java Consumer JRE“ in Form von mehreren Java SE 6 Updates (auch Update N genannt) zur Verfügung. Die Consumer JRE soll modular aufgebaut sein und über ein neues Deploy Toolkit schneller und einfacher über das Web installierbar sein. Die Consumer JRE soll lediglich 3–4 MB groß sein und erst bei Bedarf weitere Module über das Web nachladen. Weitere Features der Consumer JRE sind neue Video- und Soundfähigkeiten sowie ein neues Cross-Plattform-Look-and-Feel (Nimbus). Eine direkte Ausführung im Webbrowser über ein Plug-in ist aktuell nicht möglich. Als Workaround kann ein FX-Skript innerhalb eines Java Applets ausgeführt werden.

JavaFX Script wird aktuell von der Runtime (FXShell) interpretiert. Entsprechend langsam ist die Ausführung von Skriptcode im Vergleich zu compiliertem Code. Da JavaFX eine optimale Integration zu Java-Code (Java-Klassen) besitzt und dieser vom Java Hot Spot Compiler in nativen Code übersetzt wird, sollten bei der Entwicklung von JavaFX-Anwendungen die folgenden einfachen Regeln eingehalten werden:

- Deklarative Beschreibung der Oberflächenelemente, Binding zum Datenmodell und einfache Eventverarbeitung in JavaFX Script
- Abbildung des Präsentationsmodells in Java (Pojos)

- Komplexe Eventverarbeitung wie Berechnungen für Animationen oder serverseitige Serviceaufrufe (WS, EJB, RMI) in Java (Pojos)

In Zukunft soll jedoch ein spezieller JavaFX Compiler den Skriptcode in einen normalen Java Bytecode umwandeln und somit eine Performanz von normalem Java-Code erreichen. Ein kleiner Benchmark (<http://bubblemark.com/>) zeigt übrigens einen Vergleich zwischen purem JavaFX-Script-Code und einem mit Java optimiertem Code.

Installation und Versionsmanagement

Im Zusammenspiel mit Java WebStart kann Installation und Update von Applikationen erfolgen. Hierzu werden bei dem ersten Zugriff auf die Applikation alle benötigten Ressourcen (Jars) aus dem Internet geladen und lokal gecached (Java Application Cache). Sobald der Benutzer eine Applikation über den Java Application Cache Viewer oder über einen Shortcut startet, prüft Java WebStart automatisch, ob auf der Ursprungsquelle irgendeine *jar*-Datei der Applikation in einer neueren Version vorliegt. Dieses wird dann vor dem Start der Applikation nachgeladen. Ist keine Onlineverbindung möglich, wird die gecachte Version gestartet, womit JavaFX im Zusammenspiel mit Java WebStart offlinefähig ist.

Integration von Services

Da aus JavaFX heraus Java-Klassen verwendet und sämtliche Java-APIs genutzt werden können, bestehen die gleichen Möglichkeiten zur Serviceintegration wie in Java.

Content Bridges

Eine direkte Integration in den Browser ist derzeit nur über Java Applets möglich. Insofern gibt es derzeit auch keine bidirektionalen Brücken zwischen JavaFX und anderen Content-Welten.

Adobe Flash, Flex, AIR

Adobe stellt mit der Flex-Software, Adobe Integrated Runtime (AIR) und der Flash-Player-Software eine Platt-

form zur Verfügung, die eine Entwicklung von RIAs auf Basis von HTML und CSS, JavaScript und Ajax, Flash und Flex ermöglicht [3]. Diese RIAs sollen unabhängig davon sein, ob sie im Web, auf dem Desktop oder auf mobilen Endgeräten eingesetzt werden. AIR-Anwendungen können sowohl Ressourcen im Web als auch auf lokale Daten in Form von Dateien oder einer einfachen, integrierten SQLite-Datenbank zugreifen.

Runtime

Die betriebssystemübergreifende Laufzeitumgebung AIR ermöglicht es, dass die RIAs direkt aus dem Netz geladen und außerhalb eines Browsers ausgeführt werden können. Dabei wird die Applikation ähnlich wie bei Java Applets in einer Sandbox ausgeführt, in der man unterschiedliche Security Levels konfigurieren kann. AIR-Applikationen bestehen entweder aus kompilierten Bytecode (Flash/Flex Inhalte) oder aus zu interpretierenden Skript (JavaScript, HTML). Der Bytecode wird im Flash Player ab Version 9 ausgeführt, das HTML über die HTML-Engine der AIR-Laufzeitumgebung. Die HTML-Engine ist die in KHTML und Safari verwendete Open Source Engine Webkit.

Unter Verwendung der Flex-3-Entwicklungsumgebung kann die Präsentation in XML (*.mxml*) unter Verwendung eines speziellen MXML-Schemas sowie der Skriptsprache ActionScript 3, die konform zum ECMA-4-Standard ist, implementiert und in der Regel wahlweise für den Flash Player ab Version 9 (*.swf*) oder die AIR-Laufzeitumgebung (*.air*) kompiliert und paketiert werden. Bei komplexeren Flash-basierten RIAs besteht die Möglichkeit, die Downloadzeit zu reduzieren, indem gemeinsam genutzter Code unter Nutzung so genannter Runtime Shared Libraries (RSL) ausgelagert wird. Der Flash Player 9 besitzt einen eigenen Cache, über den Adobe-Plattformkomponenten vorbei an den Browser-Cache vorgehalten werden können. Von Adobe signierte Basisbibliotheken werden nur beim ersten Anwendungsstart geladen und stehen als permanente RSL-Kopie lokal automatisch auch anderen Anwendungen zur Verfügung.

Für mobile Endgeräte gibt es Flash Lite 3, die mithilfe einer Flash Runtime auf dem Mobile Device ausgeführt werden können.

Installation und Versionsmanagement

Die Distribution von AIR-Applikationen erfolgt über AIR-Installationsdateien (mit Extension *.air*). Den Installationsprozess selbst führt die AIR Runtime auf Basis der AIR-Installationsdatei aus. Hierbei können zwar bestimmte Parameter konfiguriert werden, der eigentliche Installationsprozessablauf ist aber nicht beeinflussbar. Jede AIR-Installationsdatei muss signiert sein. Falls ein Update für die Applikation erzeugt werden soll, muss das Zertifikat der ersten Installation verwendet werden, um auch das Update zu signieren.

Integration von Services

Für den Aufruf von Services werden in ActionScript folgende Klassen zur Verfügung gestellt:

- `HTTPService`, um auf Daten im REST-Stil zuzugreifen
- `WebService`, um auf Web Services zuzugreifen, die via SOAP bereitgestellt werden
- `RemoteObject`, um auf Web Services zuzugreifen, die über das kompakte binäre aber proprietäre AMF-(ActionScript-Messaging-Format-)Protokoll bereitgestellt werden. Serverseitig wird hierzu das AMF Gateway für die AMF-basierte Kommunikation über http benötigt.

Diese Funktionalitäten werden auch über MXML-Tags angeboten.

Content Bridges

Über die Flex Ajax Bridge kann aus JavaScript heraus über JavaScript-Wrapper-Klassen auf Flex-Inhalte zugegriffen werden und umgekehrt. Außerdem enthält Flex 3 einen Browsermanager, um Deep-Linking umzusetzen: Beliebige Flash-Komponenten können mit einer URL belegt werden, die Flash-Applikation selbst kann die URL modifizieren. Dies ist insbesondere unter der anhaltenden Diskussion interessant,

ob RIA-Technologien wie AIR nur das Kommunikationsmedium des Webs benutzen aber sonst ein Paralleluniversum aufbauen, da ihre Inhalte aufgrund fehlender Verlinkung über das Web schwer auffindbar sind.

Microsoft Silverlight

Das Windows Presentation Foundation Framework (WPF) hat bereits auf Basis der eXtensible Application Markup Language (XAML) die Definition von reichhaltigen Benutzerschnittstellen adressiert. In Microsoft Silverlight [4] wird ein eigenständiges, in der ersten Version aber zu der WPF-XAML kompatibles XAML-Markup zur Definition von Benutzerschnittstellen benutzt, die als Teil einer Webanwendung verwendet werden können. Die Benutzerschnittstelle wird dabei auf Basis von XAML und JavaScript Events definiert, die dann wie bei HTML eine Eventbasierte Interaktion mit über JavaScript Event Handler implementierten Logikcode ermöglichen, wodurch ein MVP-Muster unterstützt wird. Silverlight fokussiert die Entwicklung von reichhaltigen Webanwendungen und nicht die Entwicklung von Desktopanwendungen. Hierzu steht weiterhin WPF zur Verfügung. Dies hat zur Konsequenz, dass das vorrangige Designziel lautet, ein Maximum an Funktionalität bei minimalen Downloadvolumen anzubieten. Deshalb wurde Silverlight XAML auch als eigenständige Sprache neben WPF-XAML entworfen. Anders als WPF benötigt Silverlight z.B. derzeit nicht das .NET-Framework und bietet auch nur eine eingeschränkte Funktionalität über XAML-Tag-Libraries zur Gestaltung von Oberflächen an. Ab Silverlight-Version-1.1 (bzw. Anfang nächsten Jahres 2.0: 1.1 soll jetzt direkt als 2.0 vermarktet werden) wird neben JavaScript aber auch Managed Code (z.B. C#) unterstützt werden. Was dies für das ursprüngliche Designziel bedeutet, ist abzuwarten – eine Preview ist jetzt schon verfügbar.

Runtime

Die Laufzeitumgebung von Silverlight besteht aus einem Browser Plug-in, das

die mit XAML beschriebenen Elemente rendert. Aktuell werden Plug-ins für den Internet Explorer und Firefox unter Windows sowie Firefox und Safari unter MacOS X angeboten.

Installation und Versionsmanagement

Aktuell muss der Benutzer beim Laden einer Webseite, die Silverlight-Bestandteile einbettet, die Laufzeitumgebung manuell herunterladen. Spätestens in Version 2.0 soll der Browser die Laufzeitumgebung so herunterladen, wie er es bei jeder anderen ActiveX-Lösung (z.B. Flash) machen würde. Das soll eine Unterstützung für eine eventuell erforderliche Aktualisierung der Laufzeitumgebung einschließen.

Integration von Services

Mit der Unterstützung von Managed Code kann man die dort zur Verfügung stehende Bandbreite für die Integration von Services nutzen. Insbesondere Web Services und das Windows Communication Foundation Framework (WCF) spielen hier eine Rolle.

Content Bridges

HTML- und Silverlight-Elemente können in einer Seite gemischt werden. Jedes Silverlight-Element, welches das XAML hostet, wird als ein Host-Objekt mit einem eindeutigen Namen über JavaScript erzeugt und unterhalb eines HTML-Elements eingebettet. Die IDs

des umschließenden HTML-Elements und des Host-Objekts müssen dabei natürlich eindeutig sein.

Der Silverlight-Host ist damit einfach nur ein weiteres Objekt, das im HTML-DOM enthalten ist. Dies ermöglicht, XAML- und HTML-Objekte aus dem gleichen JavaScript-Code heraus zu ändern. Darüber hinaus ist es ohne weiteres möglich, Silverlight in ASP.NET-Webseiten zu nutzen und umgekehrt.



Marc Steger ist Senior-IT-Architekt der WidasConcepts GmbH, einer Unternehmensberatung für IT-Architekturen und Softwareentwicklung. Sein Tätigkeitsschwerpunkt liegt in der Konzeption und Realisierung von komponentenbasierten, serviceorientierten Softwarearchitekturen auf Basis der Java-EE-Plattform.



Christian Kappert ist Geschäftsführer der WidasConcepts GmbH und beschäftigt sich hauptsächlich mit serviceorientierten Architekturen, Webarchitekturen und modellgetriebener Softwareentwicklung.

Links & Literatur

- [1] Homepage der OpenAjax Alliance: www.openajax.org
- [2] Sun Developer Network: JavaFX: java.sun.com/javafx/index.jsp
- [3] Produktseite von Adobe zu AIR: labs.adobe.com/technologies/air/
- [4] Produktseite von Microsoft zu Silverlight: www.microsoft.com/silverlight/