



Bruno Schäffer

# Rich Internet Applications auf Java-Basis

Wer heutzutage an Rich Internet Applications (RIAs) denkt, der verbindet damit auf der Technologie-seite häufig Ajax, in manchen Fällen auch noch Flex oder Silverlight. Java als Basis für RIAs wird eher selten genannt. Dabei gibt es Szenarien, in denen Java seine Stärken sehr wohl ausspielen kann.

Interessanterweise hat Java ursprünglich seinen Durchbruch der Aussicht zu verdanken, eine Technologie zu sein, die es erlaubt, leichtgewichtige und leicht zu verteilende Applikationen für das Internet zu entwickeln. Die Applet-Euphorie ebte jedoch relativ schnell ab, den wahren Durchbruch erlebte Java schliesslich im Enterprise-Umfeld.

Wieso war den Applets kein Erfolg beschieden? Im Wesentlichen gab es dafür drei Gründe: Erstens war Java zu diesem Zeitpunkt noch nicht reif genug für die Entwicklung von anspruchsvolleren Applikationen, vor allem was die Realisierung von Benutzeroberflächen betraf. Zweitens führte die rasch fortschreitende Entwicklung von Java zu einer Vielzahl von unterschiedlichen Versionen der JRE (Java Runtime Environment). Die Entwickler waren sehr schnell überfordert, Applets zu entwickeln, die auf unterschiedlichen JRE-Versionen lauffähig waren. Und drittens waren Applets im Normalfall Fat-Clients, und Java bot keine Hilfestellung für die architektonische Aufteilung zwischen Client und Server.

## RIAs – Fat oder Thin?

Die generelle Architekturfrage von RIAs lautet: Fat oder Thin? Die am Markt verfügbaren RIA-Technologien lassen jeweils keine Wahl, sie schreiben klar vor, welches Architektur-szenario zur Anwendung kommt. Zusätzlich erlebt man häufig, dass die Gestaltungsmöglichkeiten der Benutzeroberfläche ein wesentlich höheres Gewicht bei der Technologiewahl bekommen als die Architekturfrage. Dennoch lohnt es sich, über die Vor- und Nachteile von Thin oder Fat Clients nachzudenken.

Thin Clients verlagern Präsentations- und Geschäftslogik auf den Server und vermeiden damit die keineswegs triviale Fragestellung, wie die Aufteilung einer Applikation zwischen Client und Server vorzunehmen ist. Die Anbindung der Präsentationslogik an die Geschäftslogik ist einfacher, da im Idealfall beide sogar im gleichen Adressraum zur Ausführung gelangen. Damit haben Fragen wie Zugriff auf grosse Datenmengen, Caching, konsolidiertes Logging oder redundanzfreie Validierung we-

sentlich weniger Brisanz. Der applikationsunabhängige Client ist durch Änderungen in der Applikation meistens nicht betroffen, was den Ausbreitungsdruck stark mindert. Darüber hinaus bietet ein Thin Client wesentlich weniger Angriffsfläche für Attacken, da der Client-Teil applikationsunabhängig ist. Schliesslich ist die Integration in klassische Webapplikationen, die architektonisch ebenfalls Thin Clients sind, klar einfacher. Fat Clients haben Vorteile im Ausnutzen der Hardwareressourcen des Clients (CPU, Memory). Bei komplexen Applikationen kann damit die Serverbelastung vermindert werden. Falls Offline-fähigkeit gefordert ist, so ist diese mit einem Fat Client leichter zu realisieren. Die Vorteile des Fat Clients sind typischerweise die Nachteile des Thin Clients und vice versa.

Die Eignung von Java als Basis muss aus zwei Gesichtspunkten beurteilt werden, da Java sowohl Laufzeit- wie Entwicklungsumgebung ist. Aus einer rein funktionalen Sicht ist die JRE eine ideale Ausführungsumgebung für RIAs. Sie bietet nicht nur Platz für Java-Applikationen, sondern auch Skriptsprachen wie zum Beispiel Groovy können diese Ausführungsumgebung nutzen. Selbst anspruchsvolle Applikationen vermögen die Grenzen der JRE auf zeitgemässer Hardware kaum auszureizen. Auch die Stabilität gilt als ausgezeichnet. Die Verfügbarkeit auf praktisch allen Plattformen ist ein weiterer Pluspunkt. Darüber hinaus ist die Sicherheit (vor allem im Vergleich zu Javascript) um Längen besser. Schliesslich ist die Marktdurchdringung von Java weit besser als allgemein angenommen. Mehr als 90 Prozent aller mit dem Internet verbundenen PCs haben eine Version einer JRE installiert, und auf mobilen Geräten ist Java mit Abstand die am meisten verbreitete Laufzeitplattform.

Es gibt jedoch Faktoren, die die Verwendung einer JRE für RIAs beeinträchtigen. Zum einen existiert eine Vielzahl von unterschiedlichen JRE-Versionen, und ein Entwickler ist gezwungen, diese Versionsunterschiede im Client-Code abzufedern. Den (gelegentlichen) Anwender anzuhalten, vor dem Starten der Anwendung noch schnell eine aktuelle

Version der JRE zu installieren, ist nicht praktikabel. Die schiere Grösse der JRE stellt die meisten Anwender einfach auf eine zu grosse Geduldprobe. Sun arbeitet zwar daran, im Rahmen der sogenannten Consumer-JRE den Installationsprozess substanziell zu verschlanken. Bislang haben sich die Versprechen allerdings noch nicht im gewünschten Umfang materialisiert. Deshalb ist Java als Laufzeitumgebung für RIAs im B2C-Umfeld eher mit Vorsicht zu geniessen. Dagegen ist die JRE für RIAs in einem kontrollierten Umfeld (Intranet oder B2B) eine hervorragende Basis.

## Was braucht es für die Entwicklung von RIAs?

Für die Beurteilung von Java als Entwicklungsplattform für RIAs müssen wir uns zuerst fragen, was ein Entwickler für die Entwicklung von RIAs denn benötigt. Dabei sollen nicht nur Funktionalität und Entwicklungseffizienz im Vordergrund stehen, sondern die Lösung sollte auch nachhaltig im Sinne moderater Wartungskosten sein. Die Anforderungen an die Funktionalität hängen natürlich sehr stark von der Art der RIA ab. Im B2C-Umfeld müssen javabasierte RIAs vor allem durch visuelle Gestaltung, Animationen und Multimediaintegration überzeugen, um Interesse beim Endanwender zu wecken. Dagegen ist für B2B die Anwenderproduktivität wesentlich wichtiger. Hier sind Animationen oder Multimedia weniger gefragt, oft sogar hinderlich.

Für die Entwicklereffizienz und Nachhaltigkeit ist Homogenität von entscheidender Bedeutung. Je weniger sich Entwickler mit unterschiedlichen Technologien herumschlagen müssen, desto produktiver sind sie und auch die Qualität der Lösung steigt. Als Negativbeispiel seien hier diverse Ajax-Umgebungen erwähnt, bei denen man gleichzeitig Experte in Java, Javascript, HTML, CSS oder XML sein muss. Auch der Einsatz von Domain Specific

**Dr. Bruno Schäffer** ist Mitgründer und Berater des Basler Softwareunternehmens Canoo AG.

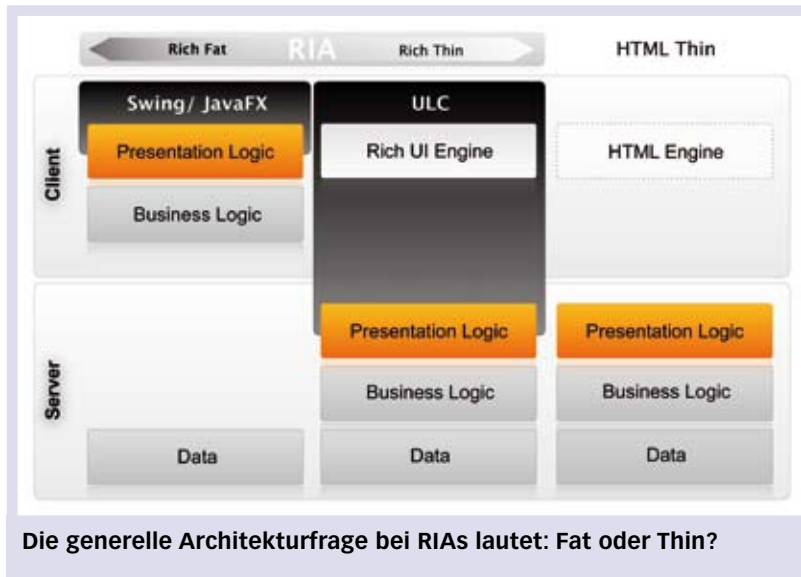
Languages (DSL) für die Beschreibung der Benutzeroberfläche (meist in Form einer deskriptiven Sprache auf der Basis von XML) ist mit Vorsicht zu geniessen. Was für einfache Beispiele verlockend aussieht, wird für RIAs mit komplexer Präsentationslogik schnell unhandlich, und spätestens in der Wartung sind die Vorteile der schnellen Erstellung verspielt. Die Homogenität einer javabasierten Lösung hat noch einen weiteren Vorteil. Im Enterprise-Umfeld ist die Wahrscheinlichkeit hoch, dass die serverseitigen Anteile einer RIA in Java implementiert sind. Damit kann serverseitiger Code sehr einfach auf der Client-Seite wiederverwendet werden, etwa die Validierungslogik.

Neben einer umfangreichen Bibliothek von funktional hochstehenden UI-Komponenten sind weitere Zutaten notwendig: Binding (d. h. Anbindung von UI-Komponenten an Datenobjekte sowie Synchronisation zwischen diesen), Layoutunterstützung und Bereitstellung von Anwendungsmustern für die Benutzeroberfläche. Die Einschränkung des Entscheidungsspielraums von Entwicklern durch die Anwendungsmuster klingt zwar bevormundend, fokussiert jedoch die kreativen Energien stärker auf die Lösungsumsetzung. Die Erfahrung zeigt, dass Homogenität gepaart mit diesen Zutaten und verpackt unter einem einfachen API das beste Rezept für Produktivität, Qualität und Nachhaltigkeit sind. Letztlich darf auch die Unterstützung für automatisiertes funktionales Testen der Benutzeroberfläche nicht fehlen, um Qualität und Nachhaltigkeit sicherstellen zu können.

### Drei typische Java-Vertreter

Wie sieht es nun in Java bezüglich dieser Anforderungen konkret aus? Drei typische Vertreter sind Swing, ULC und JavaFX:

- **Swing:** Swing ist die Standard-UI-Komponentenbibliothek von Java. Sie stellt eine reichhaltige Sammlung von Komponenten über ein Fat-Client-Programmiermodell zur Verfügung. Die Komponenten sind gut an die jeweilige Plattform angepasst und lassen sich über Skinning flexibel auch an andere visuelle Vorgaben anpassen. Die Unterstützung von automatisiertem Testen durch Bibliotheken wie Jemmy oder JFCUnit kann als sehr gut bezeichnet werden. Als RIA-Bibliothek konnte sich Swing dennoch nicht etablieren: Das API bietet zu wenig Abstraktionen, Binding



wird standardmässig nicht unterstützt, und das Layout ist für die meisten Entwickler zu komplex. Die daraus resultierende unzureichende Entwicklerproduktivität in Kombination mit dem Fat-Client-Programmiermodell hat Swing im RIA-Umfeld verständlicherweise keinen Erfolg beschert. Es gibt zwar Erweiterungen wie etwa die JGoodies Swing Suite, Beans Binding oder SwiXml, aber für Swing scheint der RIA-Zug abgefahren zu sein.

- **ULC:** UltraLightClient, eine Thin-Client RIA-Library ist mit dem Anspruch angetreten, Swing für RIAs tauglich zu machen. Dazu wurde das Swing-API auf die Serverseite verlagert, auf der Clientseite kommt lediglich ein minimaler swingbasierter Renderer zum Einsatz, der zudem applikationsunabhängig ist. Das serverseitige Programmiermodell von ULC entlastet den Entwickler von der Aufgabe, eine ideale Aufteilung zwischen Client und Server vorzunehmen. Darüber hinaus stellt ULC höherwertige Abstraktionen für die effiziente Entwicklung von B2B-RIAs zur Verfügung: Form Layout, Binding, Input-Validierung sowie einfache Konfiguration und einfaches Deployment als Applet, via Java Web Start oder als Java-Applikation.

- **JavaFX:** Webapplikationen haben die visuelle Latte für Consumer-RIAs sehr hoch gelegt. Grundsätzlich lassen sich auch mit Swing cool aussehende Applikationen entwickeln, aber trotz diverser Erweiterungen war dies nie ganz einfach. Um zu vermeiden, dass Java in diesem Bereich aufs Abstellgleis fährt, hat Sun mit JavaFX eine neue Option ins Spiel gebracht. JavaFX ist eine DSL für die Definition von Benutzeroberflächen. Mit dieser Sprache lässt sich eine moderne webartige Benutzeroberfläche wesentlich prägnanter und effizienter beschreiben, als dies mit Swing möglich ist. JavaFX gibt es in zwei Versionen: als leichtgewichtige Erweiterung

von Java SE und als JavaFX Mobile für Mobilgeräte mit Java ME. JavaFX Mobile ist eine leicht abgespeckte Version von JavaFX, die den Hardwareerestriktionen von Mobilgeräten Rechnung trägt.

JavaFX macht es sehr einfach, die Benutzeroberfläche visuell attraktiv zu gestalten sowie Multimedia und Animationen zu integrieren. Die visuelle Anpassung kann praktischerweise über ein CSS-File erfolgen. Ein weiteres Merkmal von JavaFX ist Binding, was zu einer wesentlichen Reduktion des Codes führt.

Für Consumer-RIAs hat sich eine Aufteilung in Design (ausgeführt durch einen grafischen Designer bzw. UI-Experten) und Entwicklung (durch einen Java-Entwickler) etabliert. Diese Arbeitsteilung wird von JavaFX unterstützt, indem das Design in Adobe Photoshop oder Illustrator realisiert und dann über ein Plug-in nach JavaFX überführt werden kann.

JavaFX adressiert auch den Nachteil von üblichen DSLs und ihre mangelnde Integration in Java. Die JavaFX-Sprache erlaubt das Erzeugen von Java-Objekten und den Aufruf von Java-Code. Diese Eigenschaft kann nicht hoch genug eingeschätzt werden, ermöglicht sie doch die einfache Verwendung der bewährten Java-Klassenbibliothek oder von bestehendem applikatorischen Java-Code. Auch die Einbindung von Webservices ist sehr elegant gelöst.

Aus Architektursicht ist JavaFX ausschliesslich auf die Entwicklung von Fat-Clients ausgerichtet. Die Vor- und Nachteile gelten auch für auf JavaFX basierende Fat-Clients. Zudem ist JavaFX eine relativ junge Technologie, was sich an etlichen Stellen zeigt. Weiter fehlt die Unterstützung von Offlinefähigkeit, sprich lokale Datenhaltung inklusive Synchronisation mit dem Server. Schliesslich gibt es noch keine Möglichkeit JavaFX-Benutzeroberflächen automatisiert zu testen. Dennoch ist mit JavaFX eine ernstzunehmende Technologie für RIAs aufgetaucht und das derzeitige Engagement von Sun lässt das Beste hoffen.

### Jazoon'09

Die internationale Java-Konferenz Jazoon findet 2009 vom 22. – 25. Juni in Sihlcity in Zürich statt. Informationen und Anmeldung unter: [www.jazoon.com](http://www.jazoon.com)