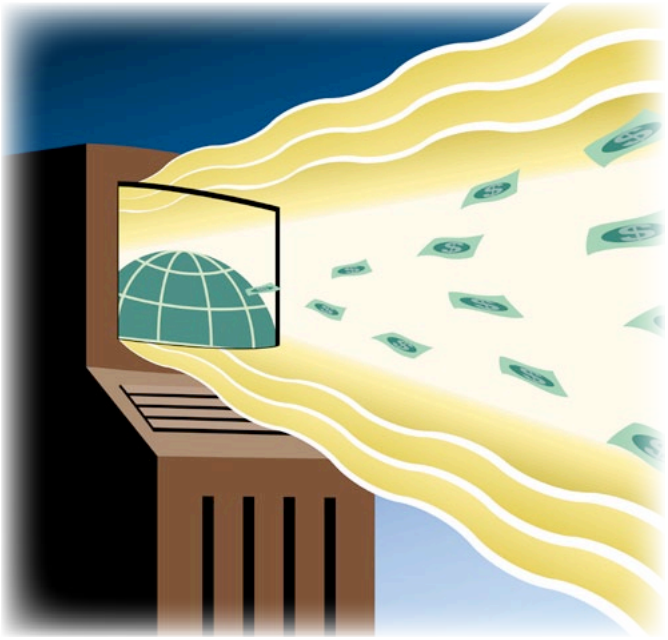
Eat the rich

Rich Clients – In oder out?

Tobias Bosch

Aktuelle Technologien im Webumfeld ermöglichen es, immer mehr Anwendungsarten als Webanwendungen zu realisieren. Dazu gehört seit Google Docs auch der Bereich der Office-Anwendungen. Das neue Betriebssystem Chrome OS von Google geht sogar so weit, dass darin ausschließlich Webanwendungen verwendet werden. Wie sind Rich Clients in diese Entwicklung einzuordnen? Sind diese damit „out"? Oder werden Webanwendungen so leistungsfähig, dass sie selbst zu Rich Clients werden können? In diesem Fall können die Oberflächen-Plattformen nicht mehr in Rich Clients und Webanwendungen getrennt werden. Für eine Architekturentscheidung werden daher feinere Entscheidungskriterien benötigt. Dieser Artikel beschreibt ausgesuchte Architekturkriterien für Oberflächen und wendet sie auf JavaFX, Adobe Flex und Eclipse RCP an.



Was ist eigentlich ein Rich Client?

Dies wird unterschiedlich definiert (siehe die Diskussion in [Mül09]). Eine der wichtigsten Eigenschaften von Rich Clients ist wohl eine reichhaltige, benutzerfreundliche und schnell reagierende Oberfläche. Rich Clients müssen nicht bei jeder Benutzerinteraktion einen Server-Request absenden und erlauben somit eine optimierte Client/Server-Kommunikation. Manche Rich-Client-Anwendungen sind offlinefähig, d. h. sie synchronisieren ihre Daten nur zu bestimmten Zeitpunkten mit einem Server. Je nachdem, wie der Begriff Rich Client definiert wird, können hier noch weitere Eigenschaften genannt werden.

Spätestens seit der Ankündigung von HTML5 (siehe [HTML5]) verschwimmen die Grenzen zwischen Rich Clients und Webanwendungen: HTML5-Anwendungen erlauben sehr reichhaltige Oberflächen, u. a. durch die Einführung des Canvas-Objekts. Offlinefähigkeit ist mithilfe von Local Storage ebenfalls möglich. Zudem kann der Netzwerkverkehr durch die Einführung von WebSockets sehr genau kontrolliert werden. Es gibt sogar Frameworks, die HTML5-Anwendungen wie lokal installierte Applikationen aussehen lassen (siehe z. B. Appcelerator [AppCel]).

Webanwendungen und Rich Clients können somit nicht anhand eindeutiger Kriterien voneinander abgegrenzt werden, sondern sie gehen ineinander über, sodass Webanwendungen gleichzeitig Rich Clients sein können. Bei Architekturentscheidungen muss somit nicht entschieden werden, ob ein Rich Client oder eine Webanwendung erstellt wird, sondern mögliche Plattformen müssen anhand von spezifischen Kriterien verglichen werden.

Im Folgenden werden für Architekturentscheidungen relevante Kriterien für Oberflächen aufgeführt. Anhand dieser Kriterien werden drei prominente Oberflächen-Plattformen, die sich selbst als „Rich Client Platform" bezeichnen, betrachtet: JavaFX [JavaFX], Adobe Flex [Flex] und Eclipse RCP [RCP].

Laufzeitumgebung

Alle hier diskutierten Plattformen verwenden eine Laufzeitumgebung, um von dem unterliegenden Client-System zu abstrahieren (Adobe AIR [AIR], Adobe Flash Player [Flash] bzw. Java-VM). Dadurch wird die Verbreitung von Anwendungen auf viele unterschiedliche Systeme erleichtert.

Eine Laufzeitumgebung beschränkt allerdings auch die Möglichkeiten einer Anwendung auf eine definierte Menge von Funktionen. Manche Laufzeitumgebungen erlauben es daher, Erweiterungen in Form von nativem Code zu verwenden. Dies ist notwendig, wenn auf spezielle Schnittstellen des Client-Systems wie z. B. Scanner zugegriffen werden soll.

Eine solche Erweiterung der Plattform muss allerdings vor dem Einsatz auf jedem System installiert werden, auf dem die Anwendung zum Einsatz kommen soll. Java erlaubt eine solche Erweiterung der Laufzeitumgebung mithilfe des Java Native Interface [JNI]. Für Adobe AIR bzw. Adobe Flash Player existiert dagegen kein entsprechendes Konzept.

Verteilung

Jede Anwendung, die lokal auf dem Client-System ausgeführt wird, muss vor ihrer Ausführung auf das Client-System übertragen werden. Sowohl Flex als auch Java bieten die Möglichkeit, die Anwendung innerhalb eines Browsers auszuführen (via Adobe Flash Player bzw. Java-Applets). Dadurch wird die Anwendung geladen, sobald eine entsprechende HTML-Seite im Browser geöffnet wird. Beim Beenden des Browsers wird die Anwendung beendet und wieder gelöscht. Eclipse RCP ist nicht als Applet verwendbar, da die SWT-Oberfläche die Laufzeitumgebung via JNI erweitert, was gegen die Sicherheitsbestimmungen von Applets verstößt.

Neben der Ausführung im Browser bieten Flex und Java die Möglichkeit, eine Anwendung permanent auf dem Client-System zu installieren. Dabei ist es jeweils möglich, die Anwendung über das Internet zu verteilen und sie durch einen einzelnen Klick zu installieren (sogenannte 1-Click-Installation, siehe Adobe AIR bzw. Java WebStart [JWS]).

Bevor eine der Oberflächen-Plattformen verwendet werden kann, muss die entsprechende Plattform einmalig auf dem Client-System installiert werden. Im Gegensatz zu einem Browser gehören diese Plattformen nicht zur Standardinstallation vieler Systeme.

Oberflächen-Technologie

Die betrachteten Plattformen verwenden unterschiedliche Oberflächen-Technologien: Eclipse RCP setzt auf das Standard Widget Toolkit [SWT]. SWT bildet Oberflächen-Komponenten direkt auf die UI-Komponenten des Betriebssystems ab (sogenannte Heavyweight UI). Es wurde entwickelt, um die Oberfläche möglichst reaktionsfähig und schnell zu machen.

Java FX und Adobe AIR verwenden dagegen einen anderen Ansatz: Nur dort, wo es notwendig ist (z. B. für den Rahmen der Anwendung), werden direkt Oberflächen-Komponenten des Betriebssystems verwendet. Die feingranularen Komponenten wie Buttons usw. werden dagegen innerhalb der Laufzeitumgebung gerendert (sogenannte Lightweight UI). Dieses Vorgehen erlaubt größere Flexibilität bei der Gestaltung der Oberflächen-Komponenten.

Binding

Alle untersuchten Oberflächen-Plattformen unterstützen das sogenannte Binding. Dieses ermöglicht die Trennung von Darstellung und Verhalten der Oberfläche. Dazu werden die UI-Komponenten mit einem separaten Oberflächenmodell synchronisiert, das den Zustand und das Verhalten der Oberfläche repräsentiert. Ein Textfeld kann z. B. an ein Objekt mit einem String-Feld gebunden werden. Ändert sich das Textfeld, so wird das String-Feld in dem Objekt aktualisiert und umgekehrt.

Diese Trennung von Darstellung und Verhalten wird auch als „Presentation Model Pattern“ [PMP] bezeichnet (s. Abb. 1). Es erleichtert die Einführung einer spezialisierten UI-Designer-Rolle in Projekten. Außerdem werden dadurch die Änderung der Darstellung und das Testen des Verhaltens der Oberfläche erleichtert.

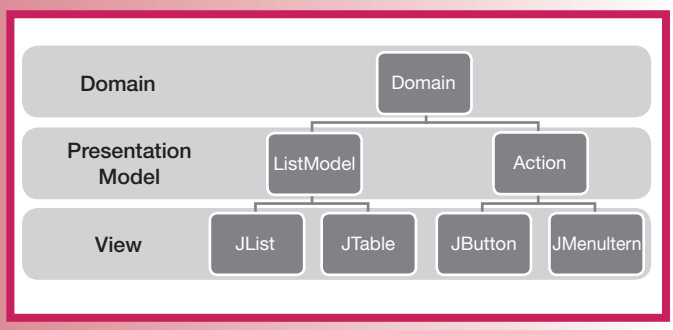


Abb. 1: Presentation Model

Die Verwendung einer Binding-Schicht hat sich in vielen Bereichen bewährt: Es wird von JavaServer Faces verwendet, Microsoft setzt in seiner Bibliothek Windows Presentation Foundation auf eine Binding-Schicht und in vielen Java-Programmen wird das Framework JGoodies-Binding [Bind] eingesetzt, das eine Binding-Schicht für Swing darstellt.

Flex und JavaFX bieten ein Binding zwischen beliebigen Variablen, auch unabhängig von Oberflächen-Elementen. Bei beiden Plattformen lässt sich das Binding auch über die deklarative Definition des Komponentenbaums angeben (siehe unten). Das Binding in Eclipse RCP muss zurzeit (Version 3) noch programmatisch erstellt werden. Ab Version 4 existiert hier ebenfalls eine deklarative Oberflächendefinition (siehe unten) mit integriertem Binding.

Deklarative Oberflächendefinition

Sowohl Flex als auch JavaFX erlauben es, den Komponentenbaum für die Oberfläche *deklarativ* zu erstellen. Damit ist gemeint, dass die Struktur des Codes die Struktur der Oberfläche widerspiegelt, vor allem bezüglich der Verschachtelung (s. Abb. 2). Zu einer deklarativen Oberflächendefinition gehört ebenfalls eine sehr kompakte und prägnante DSL (Domain Specific Language). JavaFX verwendet dazu eine an Java angelehnte DSL, Flex setzt auf XML-Dateien in einem speziellen Format (MXML).

Der Einsatz einer deklarativen Oberflächendefinition beschleunigt die Entwicklung einer Oberfläche deutlich. Außerdem wird dadurch die Erstellung von Design-Werkzeugen erleichtert. Eclipse RCP (Version 3) unterstützt dieses Vorgehen noch nicht. In der Version 4 wird dies dann mithilfe von XWT [XWT] ebenfalls möglich sein.

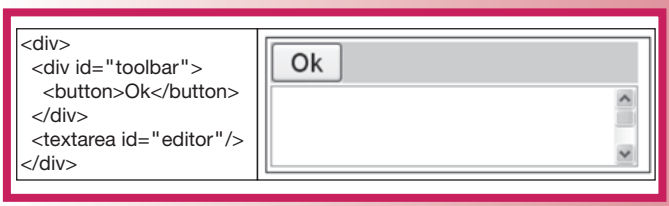


Abb. 2: Deklarative Oberflächendefinition

Animationen

Animationen machen eine Anwendung optisch ansprechend und steigern die Akzeptanz beim Endbenutzer. Deshalb bietet sowohl Flex als auch JavaFX eine besondere Unterstützung für Szenengraphen in Form von Zuständen und animierten Übergängen zwischen diesen Zuständen. Eclipse RCP erlaubt Animationen lediglich durch die Verwendung von Java2D, was zu umfangreichem und komplexem Code führt.

Anwendungsrahmen

Eclipse RCP bietet als einzige der untersuchten Plattformen einen Anwendungsrahmen für die Oberfläche, die sogenannte Workbench. Diese definiert ein Grundgerüst, das durch die jeweilige Anwendung erweitert werden muss, und bereits viele vordefinierte Funktionen enthält (s. Abb. 3).

Die Anwendung stellt dazu ihre Basisfunktionalität in Form von sogenannten Views, Editoren und Commands zur Verfügung. Editoren repräsentieren hierbei Dokumente, die der Benutzer öffnen, bearbeiten und wieder speichern kann. Views stellen ebenfalls fachliche Sachverhalte dar, diese können jedoch nicht geöffnet, bearbeitet und gespeichert werden. Commands sind Aktionen, die der Benutzer ausführen kann (z. B. Buttons).

Aus diesen Grundelementen entsteht dann das Hauptfenster einer Eclipse-RCP-Anwendung: Die Commands werden je nach ihrer Konfiguration in der Toolbar und/oder im Menü angezeigt. Die Views werden zu sogenannten Perspektiven zusammengefasst. Perspektiven eigenen sich sehr gut für die Definition von fachlichen Sichten auf die Anwendung, wie z. B. „Adresserfassung“ oder „Mahnungen schreiben“. Editoren werden schließlich in einem Teil des Hauptfensters angezeigt, den alle Perspektiven gemeinsam besitzen.

Eine Eclipse-RCP-Anwendung lässt sich darüber hinaus sehr leicht vom jeweiligen Benutzer auf seine Bedürfnisse anpassen, wie z. B. die Anordnung und die Auswahl der Views in einer Perspektive, die Platzierung der Commands in Menüs und Toolbars, die Definition von Tastaturkürzeln für Commands usw.

Zusammenfassung

Oberflächen-Plattformen können nicht mehr in Rich Clients und Webanwendungen getrennt werden. Daher werden für Architekturentscheidungen feinere Entscheidungskriterien benötigt. In diesem Artikel wurden die Architekturkriterien Laufzeitumgebung, Verteilung, Oberflächen-Technologie, Binding, deklarative Oberflächendefinition, Animationen und Anwendungsrahmen vorgestellt und für drei Oberflächen-Plattformen untersucht.

Die Plattformen JavaFX, Adobe Flex und Eclipse RCP bieten weitreichende Möglichkeiten für leistungsfähige Oberflächen. Alle drei setzen die Installation einer Laufzeitumgebung auf dem Client-System voraus. Sie unterstützen eine 1-Click-Installation und bis auf Eclipse RCP die Ausführung im Browser. JavaFX und Adobe Flex eignen sich beispielsweise durch den Einsatz einer Lightweight UI, der Verwendung von Binding in Kombination mit einer deklarativen Oberflächendefinition und der integrierten Unterstützung von Animationen sehr gut für Anwendungen mit hohen Anforderungen an das Design. Eclipse RCP bietet dagegen durch seinen Anwendungsrahmen einen sehr hohen Mehrwert an Funktionalität, sofern die vorgegebene Struktur des Anwendungsrahmens den Anforderungen nicht widerspricht.

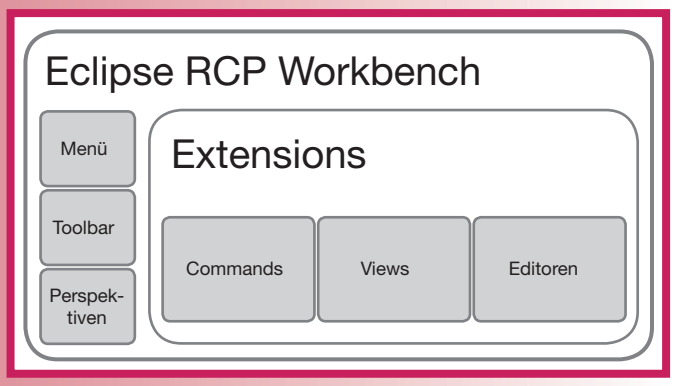


Abb. 3: Eclipse RCP Workbench

Links

[AIR] Adobe AIR, http://www.adobe.com/products/air/
[AppCel] Appcelerator, http://www.appcelerator.com/
[Bind] JGoodies Data Binding, https://binding.dev.java.net
[Chrome] Google Chrome OS, http://www.chromium.org/chromium-projects
[Flash] Adobe Flash, http://get.adobe.com/de/flashplayer/
[Flex] Adobe Flex, http://www.adobe.com/products/flex/
[HTML5] 5th major revision of the HyperText Markup Language, W3C, http://dev.w3.org/html5/spec/Overview.html
[JavaFX] JavaFX, http://javafx.com/
[JNI] Java Native Interface, http://java.sun.com/docs/books/jni/
[JWS] Java WebStart, http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp
[Mül09] Kolumne: Tour de GUI, Florian Müller, in: JavaMagazin 07/2009
[PMP] Presentation Model Pattern, http://martinfowler.com/eaaDev/PresentationModel.html
[RCP] Eclipse Rich Client Platform, http://www.eclipse.org/rcp/
[SWT] Eclipse Standard Widget Toolkit, http://www.eclipse.org/swt/
[XWT] Eclipse XML Window Toolkit, http://wiki.eclipse.org/E4/XWT

Dipl.-Inform. **Tobias Bosch** ist Senior Consultant bei der OPITZ CONSULTING GmbH. Sein Schwerpunkt liegt auf dem Entwurf und der Implementierung von Enterprise-Architekturen sowie auf der Durchführung von Systemintegrationen.
E-Mail: tobias.bosch@opitz-consulting.com