Single Sourcing in Java: Desktop-Anwendung & Web-Applikation aus einer Quelle

Björn Christoph Fischer & Oliver Zandner
Triestram & Partner GmbH (t&p)
Bochum

Schlüsselworte:

Single Sourcing, Java, Web 2.0, Rich Internet Applications (RIA), Rich Ajax Platform (RAP), Rich Client Platform (RCP), Eclipse, SWT, RWT, Migration von Oracle Forms/Reports nach Java

Einleitung

Rich Internet Applications wie etwa Google Maps sind aus dem Internet nicht mehr wegzudenken. Sie zeichnen sich aus durch ihre hohe Interaktivität. Und sie sind überall dort verfügbar, wo ein Internet-Zugang und ein Browser vorhanden sind. Nutzer erwarten diese Merkmale deshalb auch zunehmend von ihren Business-Anwendungen. Diese sind zwar komfortabel zu bedienen, aber nicht mobil, denn sie sind an das Endgerät gebunden, auf dem sie installiert sind.

Wie lassen sich die beiden genannten Anforderungen nach hohem Bedienungskomfort und nach Mobilität zusammenbringen? Es ließen sich zwei separate Programme entwickeln – eine Desktop- und eine Web-Applikation. Das wäre jedoch wirtschaftlich nicht sinnvoll, da in etwa doppelter Aufwand und damit doppelte Kosten anfallen. Eine Lösung bietet das Single Sourcing mit Eclipse RCP und RAP.

Ausgangslage im Projekt

Die oben genannten Anforderungen stammen aus der Produkt-Entwicklung: Die Triestram & Partner GmbH (**t&p**) entwickelt seit 1991 lisa.lims – ein Labor-Informations- und Managementsystem. lisa.lims basiert auf der Oracle Datenbank sowie Oracle Forms und Reports. Oracle Forms läuft seit der Version 6i im Browser, und zwar als Java Applet. Das ist jedoch an Voraussetzungen gebunden: Es ist das Java Plug-In für den Browser erforderlich und es müssen die Oracle-spezifischen Java-Bibliotheken vorhanden sein. Damit ist die Mobilität des Benutzers eingeschränkt.

Die Erwartungen neuer Nutzer an lisa.lims sind geprägt durch die eingangs beschriebene Situation: Bestimmte Gruppen von Anwendern sind mobil außerhalb ihres Büros tätig; sie erfassen Daten auf mobilen Endgeräten und werten sie auch dort aus. Andere Anwender haben einen festen Arbeitsplatz und erwarten vor allem hohen Bedienungskomfort – zum Beispiel derart, dass die Anwendung

komplett über Funktionstasten bzw. Tastatur-Kürzel bedienbar ist. Um beide Anforderungen wirtschaftlich und effizient erfüllen zu können, sollten sie aus einer Code-Basis bedient werden. Daher basiert lisa.lims mit der Version 10 auf dem Single-Sourcing-Ansatz.

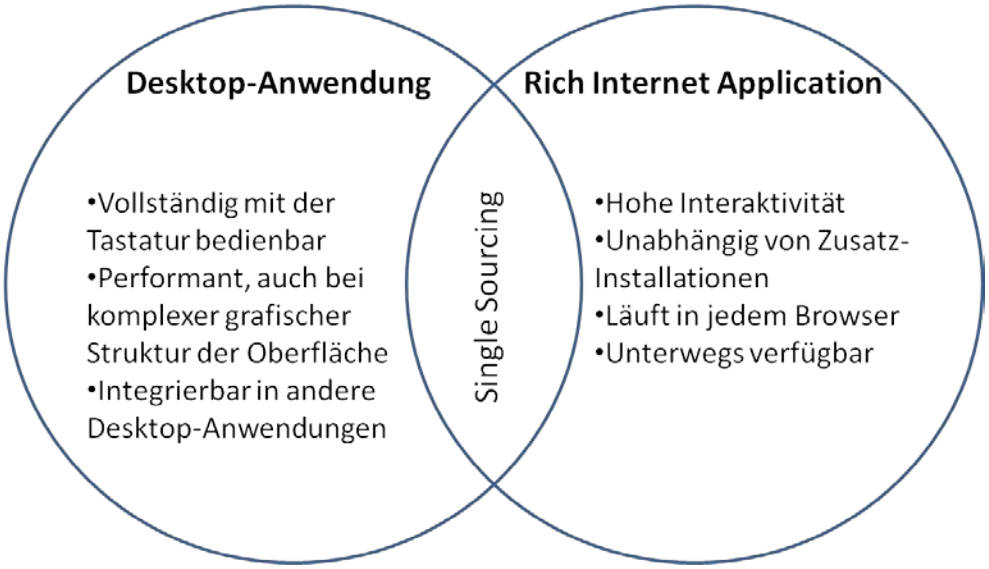

Abb. 1: Single Sourcing – Funktionale Sicht

Welches Ziel verfolgt Single Sourcing?

Das Ziel von Single Sourcing ist es, die „klassische“ Desktop-Anwendung und die Rich Internet Application aus derselben Code-Basis zu bedienen, um Kosten zu senken und die Wartbarkeit der Applikation zu erhöhen:

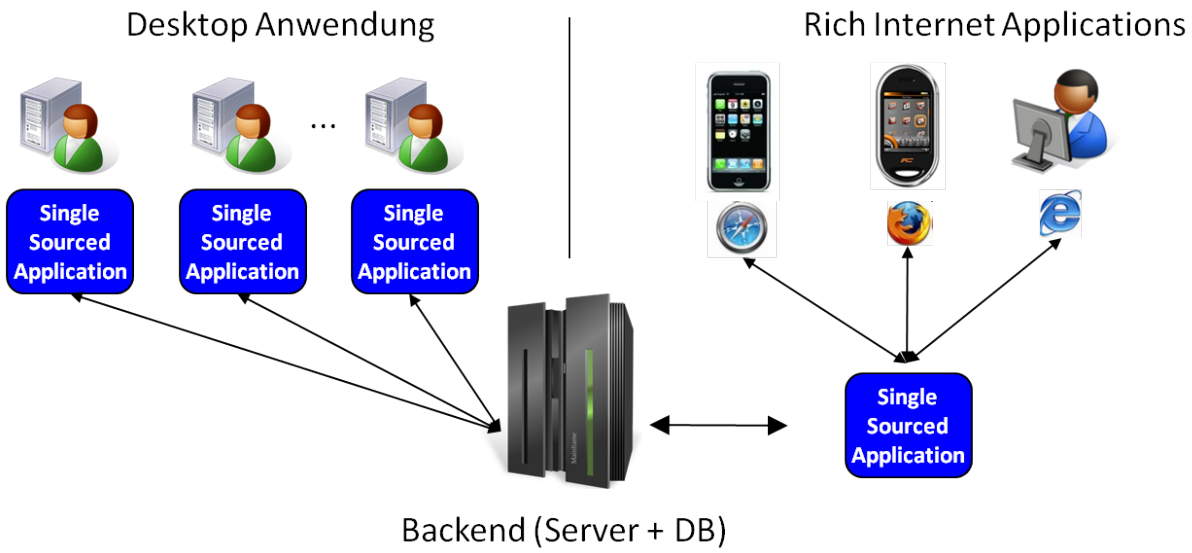

Abb. 2: Single Sourcing – Ziel

Wie funktioniert Single Sourcing mit Eclipse?

Als Entwicklungs- und Laufzeitumgebung für das oben genannte Projekt wurde Eclipse gewählt. Eclipse stellt zur Entwicklung von grafischen Benutzeroberflächen für Desktop-Anwendungen die sog. Rich Client Platform zu Verfügung (RCP). Die grafische Benutzeroberfläche der Anwendung wird aus mächtigen Komponenten zusammengesetzt. Es handelt sich dabei um Komponenten des sog. Standard Widget Toolkit (SWT) – einer Bibliothek von Oberflächen-Elementen wie Button, TextField usw. Bemerkenswert hierbei ist: Die einzelnen Komponenten werden dargestellt mit den grafischen (Bord-)Mitteln des jeweiligen Betriebssystems, auf dem die Anwendung läuft. Dies bildet die technische Grundlage dafür, die Betriebssystem-Schicht gegen eine Web-Server-Schicht austauschen zu können und somit Single Sourcing betreiben zu können (siehe unten).

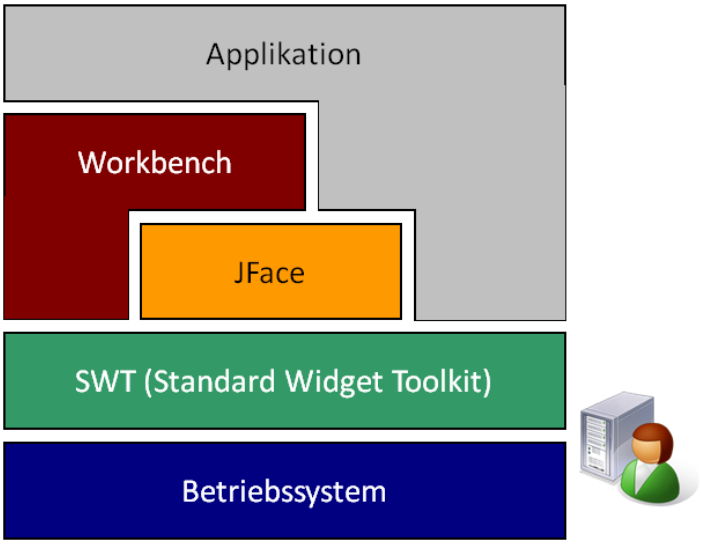

Abb. 3: Eclipse Rich Client Platform (RCP) & Standard Widget Toolkit (SWT)

Von der Eclipse Rich Client Platform zur Rich Ajax Platform
Wie wird aus der RCP-Desktop-Anwendung die Rich Internet Application? Mittels der sog. Rich Ajax Platform (RAP): Im obigen Modell werden die unteren beiden Schichten ausgetauscht – und zwar das SWT gegen das Rap Widget Toolkit (RWT) und das Betriebssystem gegen einen Servlet-Container, da es sich ja um eine Web-Anwendung handelt:

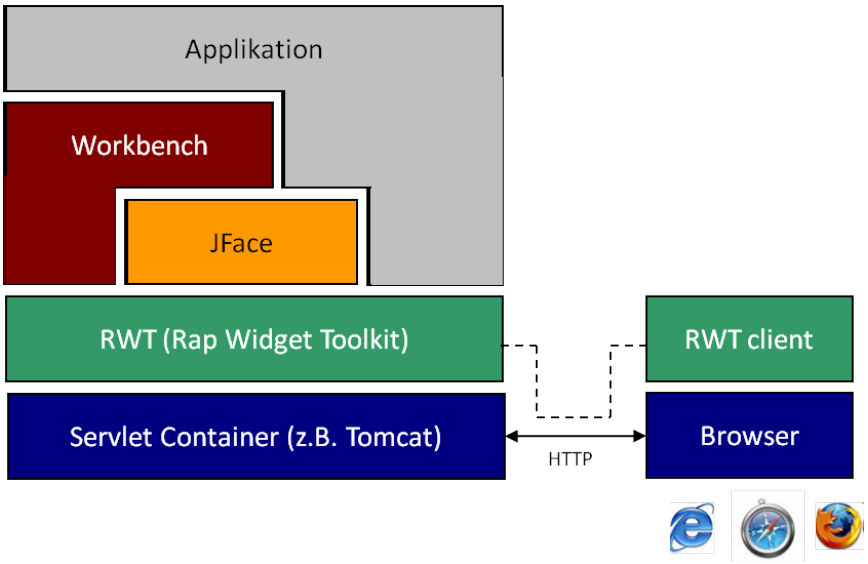

Abb. 4: Rap Widget Toolkit (RWT)

Bei RWT handelt es sich ebenfalls um eine Komponenten-Bibliothek zum Erstellen grafischer Benutzeroberflächen – allerdings auf Basis von Servlets, JavaScript und einer Ajax-Kommunikation zwischen Browser und Anwendung. RWT hat zwei wichtige Besonderheiten:

1. Seine Schnittstellen (APIs) sind kompatibel mit denen von SWT. Das bedeutet: Der Quellcode einer SWT-basierten Applikation muss nicht geändert werden, wenn SWT gegen RWT ausgetauscht wird. Diese Eigenschaft ermöglicht das Single Sourcing Verfahren. Desktop-Anwendung und Rich Internet Application basieren auf demselben Quellcode, nutzen jedoch unterschiedliche grafische Bibliotheken (RWT vs. SWT), deren Schnittstellen identisch sind.

2. RWT basiert auf der Ajax-Technologie. Das Besondere daran ist: Wenn der Benutzer z.B. auf einen Button klickt, können als Reaktion auf dieses Ereignis nur ein oder mehrere Teile der Web-Seite im Browser aktualisiert werden, und nicht die komplette Seite. Dies macht einen großen Teil der Interaktivität und der Bedienungskomforts von Rich Internet Applications aus. Eine solche Internet Seite erreicht annähernd den Bedienungskomfort einer Desktop-Anwendung.

Wo liegen die Grenzen des Single Sourcing?
Bei einer „single gesourcten" Java-Desktop-Applikation arbeitet jeder Nutzer mit seiner eigenen Instanz des Programms, die im Speicher seines **lokalen Rechners** läuft. Im Gegensatz dazu läuft bei einer „single gesourcten" Web-Anwendung lediglich **genau eine** Instanz der Anwendung im Speicher des **zentralen Web-Servers**. Das bedeutet, dass sich alle Anwender der Web-Anwendung diese eine Programm-Instanz teilen (Multiuser-Umfeld). Daraus resultieren die folgenden Grenzen des Single Sourcing:

1. Singletons (Einzelstücke):
   Ein Singleton ist eine Klasse, von der in einer Anwendung genau eine Instanz existiert. Wird dieses Entwurfsmuster in der Desktop-Anwendung genutzt, muss geprüft werden, ob das auch für die Web-Anwendung adäquat ist. Nicht adäquat wäre es zum Beispiel, wenn sich in der Web-Anwendung alle Benutzer dasselbe Singleton-Warenkorb-Objekt teilen.

2. Maus-Tracking:
   Maus-Tracking bedeutet, dass die Anwendung auf jede Bewegung reagiert, die der Benutzer mit der Maus vollzieht. So entsteht hohe Interaktivität in einer Desktop-Anwendung. Bei einer browser-basierten Anwendung würde jedoch die Übermittlung jeder Mausbewegung an den Server zu einer immensen Auslastung der Verbindung führen. Deshalb besitzt die Rich Ajax Platform keinerlei Möglichkeit auf Mausbewegungen zu reagieren. Jedoch könnte eine Anwendung das Maus-Tracking clientseitig (also im Browser per JavaScript) behandeln, was jedoch das Single Sourcing erschwert.

3. Desktop-Integration (z.B. Dialog „Datei öffnen / speichern")
   In den meisten Betriebssystemen existiert ein Dialog zur Auswahl von Dateien beim Öffnen bzw. Speichern. Im Web ist jedoch nicht allgemein festlegbar, ob die Datei auf der Client- oder auf der Server-Seite geöffnet bzw. gespeichert werden soll bzw. ob dem Benutzer diese Möglichkeit aus Sicherheitsgründen überhaupt zur Verfügung stehen darf. Deshalb verzichtet die Rich Ajax Platform auf einen solchen Dialog, der im Desktop-Bereich zur „Standardausstattung" gehört. Es

gibt jedoch Lösungen (z.B. das sog. Upload-Widget), um konkrete Anforderungen umzusetzen. Diese erfordern jedoch beim Single Sourcing besondere Aufmerksamkeit.

4. Key Bindings
   Als Key Binding bezeichnet man die Verknüpfung einer Programmaktion mit einer bestimmten Tastenkombination. Drückt der Benutzer die definierte Tastenkombination, wird die entsprechend gebundene Aktion ausgelöst. Im Gegensatz zur Desktop-Welt ist es mit JavaScript nicht möglich, sämtliche Tastenkombinationen für die Anwendung zu verwenden, da diese möglicherweise vom Browser belegt sind (z.B. F5 zur Aktualisierung der Webseite). Diese Einschränkungen sind beim Single Sourcing zu berücksichtigen.

Wie hoch ist die Wiederverwendungsrate im Projekt?
In dem oben beschriebenen Projekt ging es u.a. darum, auf der Basis des Single Sourcing Verfahrens ein Framework zu schaffen, um unsere Standard-Software lisa.lims in Richtung Java und Open Source zu modernisieren. Dazu wurde eine Code-Basis von z.Zt. ca. 24.000 Java-Code-Zeilen erstellt. Diese verteilen sich wie folgt:

- ca. 22.750 im Bereich „Common“
- ca. 460 im Bereich „RCP“
- ca. 820 im Bereich „RAP“

Damit ergibt sich eine Wiederverwendungsrate von ansehnlichen 95%:

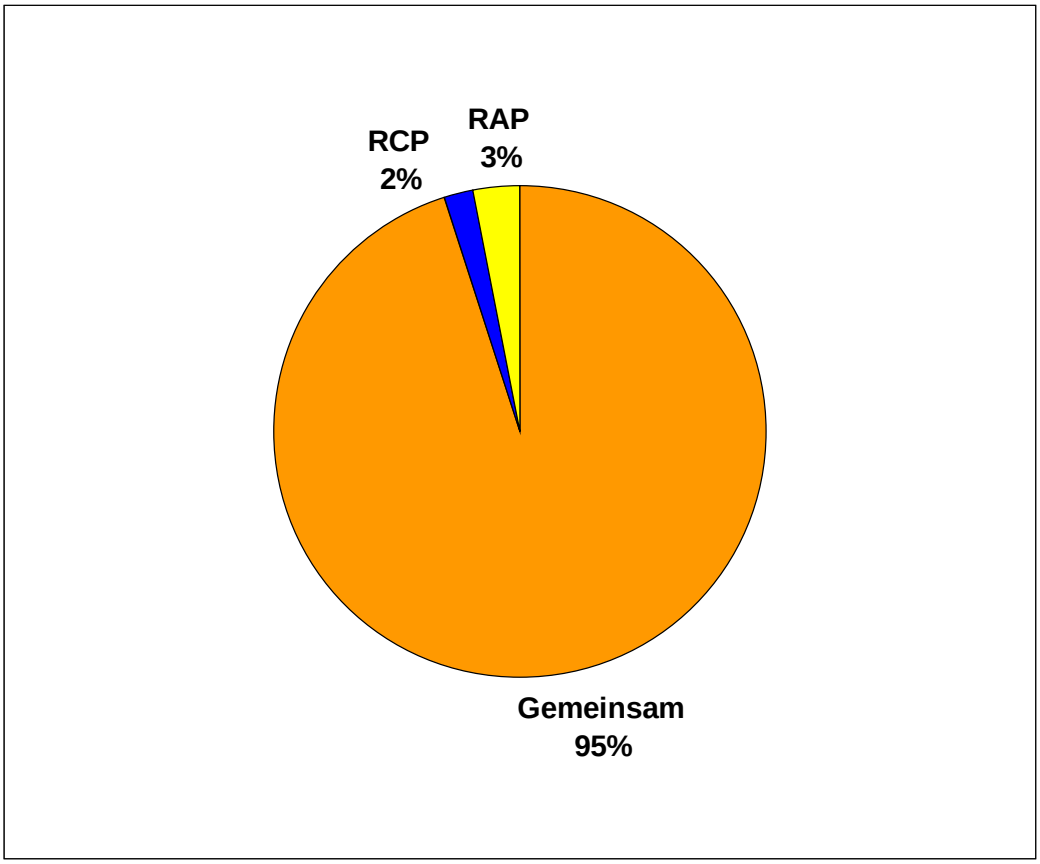

Abb. 5: Wiederverwendungsrate

Und das Fazit?
Unser Fazit lautet wie folgt:

1. Single Sourcing ist praktikabel. Auch bei einer umfangreicheren und komplexeren Anwendung wie lisa.lims.

2. Single Sourcing ist wirtschaftlich. Denn bei einer zweigleisigen Entwicklung von Desktop- und Rich Internet Application reduziert es den Entwicklungsaufwand erheblich – dank einer gemeinsamen Code-Basis von ca. 95%.

3. Single Sourcing ist schlank bzw. leichtgewichtig. Denn die gewählten Basis-Technologien wie RWT und SWT basieren allesamt auf der Java Standard Edition (JSE). Damit reicht für die Laufzeitumgebung ein schlichter Servlet-Container. Das macht die Anwendung unabhängig von den Hersteller-Spezifika der am Markt z.Zt. verfügbaren Application Server, die für eine Java Enterprise Architektur erforderlich wären.

4. Single Sourcing ist offen. Weiterer Java-Bibliotheken können einfach integriert werden. Wie etwa JFreeChart zur interaktiven Visualisierung von Messkurven. Ebenso einfach ist die Anbindung von Datenbanken.

5. Die Wahl von Eclipse RCP / RAP fiel im Rahmen der Migration der Standard-Software lisa.lims in Richtung Java. Die Wahl von Eclipse RCP / RAP für die Client-Seite hat die Wahl für das Spring Framework für die Server-Seite bedingt, da beide Plattformen auf OSGi basieren und so eine Durchgängigkeit in den Basistechnologien erreicht werden konnte. Durch das Spring Framework konnten weitere datenbank-relevante Anforderungen umgesetzt werden (wie zum Beispiel die Wiederverwendung von vorhandenem PL/SQL in der Datenbank, pessimistisches Sperren von Datensätzen und die dedizierte Benutzer-Anmeldung an der Datenbank[1]).

Kontaktadresse:

Björn Christoph Fischer & Oliver Zandner
Triestram & Partner GmbH (**t&p**)
Kohlenstraße 55
D-44795 Bochum

Telefon:	+49 (0) 234-9 43 75 - 0
Fax:	+49 (0) 234-45 22 06
E-Mail	b.fischer@t-p.com bzw. o.zandner@t-p.com
Internet:	www.t-p.com

[1] Siehe dazu das Paper „Smart Migration: Die schrittweise Umstellung eines komplexen Informationssystems von Oracle- auf Java-Open-Source-Technologien“, Thomas Haskes, Oliver Zandner, **t&p** GmbH.