Fallstudie

„B3“ – Eine konfigurierbare Plattform für das Kerngeschäft eines Medienunternehmens

Der Fernsehsender RTL2 benötigt zur Unterstützung seiner internen Abläufe komplexe Softwaresysteme. Da der freie Markt keine Software anbietet, die den besonderen Bedürfnissen des Senders entspricht, war eine eigene Plattform zur Unterstützung des Kerngeschäfts geschaffen worden. Auf dieser Plattform basierten etwa 20 Teilapplikationen zur Verwaltung und Planung von Fernseh-Assets. Um auch zukünftigen Anforderungen gewachsen zu sein, wurde im Sommer 2003 das Projekt „B3“ mit dem Ziel initiiert, die nächste Generation dieser Plattform zu entwickeln.

Ausgangslage und Zielsetzung

Die vorausgehende Plattform war als Zwei-Schichten-Architektur konzipiert: Die Client-Applikationen, entwickelt mit Visual Basic, greifen direkt auf eine Microsoft SQL-Server Datenbank zu. In der Datenbank befinden sich sowohl Geschäftsdaten und Geschäftslogik als auch Konfigurationsdaten. Ferner verwaltet die Datenbank die Benutzerdaten und steuert ein feingranulares Berechtigungssystem.

Ziel des Projekts „B3“ war es, auf Grundlage der vorhandenen Datenbank eine neue Plattform als Drei-Schichten-Architektur zu erstellen. Daraus ergaben sich drei wesentliche Anforderungen:

1. Die bestehende Datenbank mit der in ihr enthaltenen Logik und allen Daten muss weiterhin verwendet werden.
2. Die bestehenden Teilapplikationen müssen schnell und zuverlässig portiert werden können.
3. Alte und neue Teilapplikationen müssen im Parallelbetrieb laufen können.

Zusätzlich sollte als Einstiegspunkt in die neue Plattform ein Portal entwickelt werden, über das die Mitarbeiter Zugang zu allen Teilapplikationen erhalten.

Das „B3“-Portal

Technologie

Zu Beginn des Projekts „B3“ fiel die Entscheidung für Java als technologische Basis. Die Datenbank blieb als zentrale Systemkomponente bestehen. Für die Mittelschicht der Plattform wurde der J2EE-kompatible Applikationsserver JBoss ausgewählt. Als Persistenzschicht wurde das Tool Hibernate integriert, das Geschäftsdaten objektorientiert als Geschäftsklassen zur Verfügung stellt.

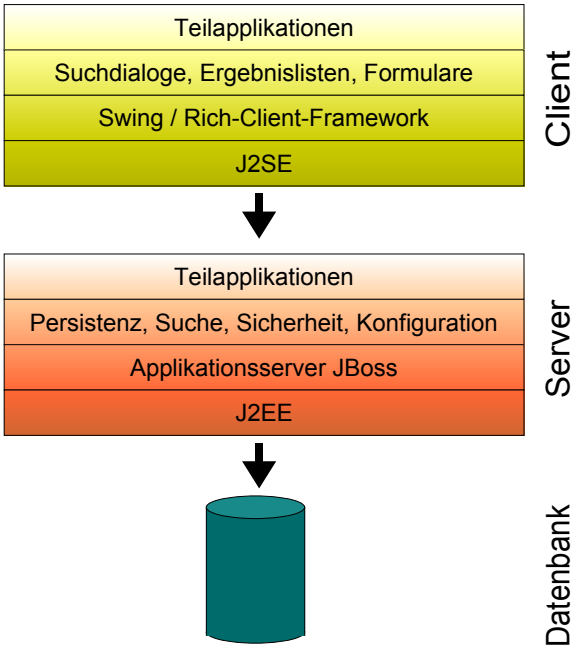
Auf der Client-Seite fiel die Wahl auf die Swing-Bibliothek von Java. Um sicherzustellen, dass alle Client-Applikationen entsprechend einer Gui-Richtlinie einheitlich aussehen und sich gleichartig bedienen lassen, wurde ein Framework entworfen, das die Grundlage aller Client-Applikationen bildet.

Architektur

Die Basis-Architektur der Plattform „B3“ besteht aus einem Kern, auf den alle Teilapplikationen aufsetzen. Dieser Kern umfasst Funktionen für Client-Applikationen und Dienste, die im Applikationsserver ablaufen. Der Kern ist in hohem Maße konfigurierbar. Dies bedeutet, die meisten wiederkehrenden Funktionen der Teilapplikationen müssen nicht immer wieder neu programmiert werden, sondern können durch Konfigurationsdaten modelliert werden.

Im Applikationsserver umfasst der Kern Dienste für *Persistenz*, also das Laden und Speichern von Daten, für das Ausführen von *Suchabfragen*, für *Sicherheit*, also die Überprüfung von Berechtigungen, und für die Zusammenstellung von *Konfigurationsdaten*. Alle Dienste laufen zustandslos und lassen sich daher leicht skalieren.

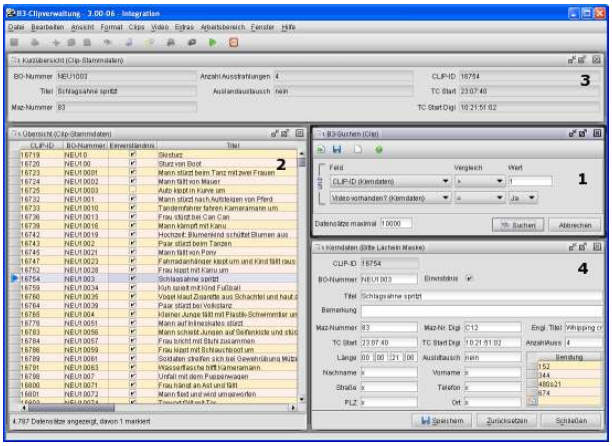
Das *Framework*, das den Kern der Client-Applikationen bildet, ermöglicht es, diese Applikationen modular zu entwickeln. Jede Client-Applikation kann somit individuell aus einzelnen Bausteinen zusammensetzt werden. Eine zentrale Konfiguration stellt auch langfristig sicher, dass sich die Client-Applikationen mit geringem Aufwand an neue Anforderungen anpassen lassen.



Die Drei-Schichten-Architektur von „B3“

Realisierung

Die wichtigsten Konzepte der Realisierung werden anhand der Clipverwaltung erläutert, die als erste Teilapplikation portiert worden ist. Sie weist vier typische Fenster auf, wie in der folgenden Abbildung zu sehen ist: einen Suchdialog (1), eine Tabelle für die Ergebnisdarstellung (2) und zwei Formularfenster, nämlich eine schreibgeschützte Kurzübersicht (3) und ein Kerndatenformular zum Ändern der Datensätze (4).



Die „B3-Clipverwaltung“

Das generische Client-Framework erhält die Konfigurationsdaten einer Client-Applikation zur Laufzeit vom Applikationsserver. Dort liegen diese Daten teilweise statisch vor, teilweise werden sie dynamisch durch Informationen aus der Datenbank ergänzt. Das Framework interpretiert diese Daten und initialisiert die Teilapplikation. Dieser Prozess wird im Folgenden für die Suche, die Ergebnisdarstellung und ein Formular skizziert.

Um eine *Suchabfrage* auszuführen, steht den Anwendern ein Suchdialog zur Verfügung. Der Dialog stellt eine Liste von Suchfeldern bereit, aus denen die Anwender beliebig komplexe Suchabfragen erstellen können. Die Liste der zur Verfügung stehenden Suchfelder wird in Form von Konfigurationsdaten übermittelt, wie folgender Ausschnitt zeigt:

```
<searchdialog identifer="CLVW">
  <search type="de.rtl2.b3.clvw.bo.Clip">
    <field name="Id"/>
    <field name="MazNr"/>
    <field name="VideoVorhanden"/>
  </search>
</searchdialog>
```

Die Anwender können frei wählen, welche Suchfelder sie kombinieren und verschachteln wollen. Dazu stehen Verknüpfungen mit UND und ODER zur Verfügung:

Der Suchdialog der „B3-Clipverwaltung“

Sobald ein Anwender eine Suche startet, erstellt der Suchdialog eine Beschreibung der Suche und schickt sie zur Bearbeitung an den Applikationsserver. Dort übernimmt eine Komponente mit dem Namen *Query Engine*[1] zunächst die Aufgabe, aus der Beschreibung der Suchabfrage eine SQL-Abfrage zu formulieren. Anschließend führt sie die SQL-Abfrage aus und sendet die aufbereiteten Ergebnisse an die Client-Applikation zurück.

In der Client-Applikation wird das *Ergebnis der Suchabfrage* in einer Tabelle angezeigt. Dafür ist eine weitere konfigurierbare Komponente verantwortlich. Sie wird mit Informationen darüber initialisiert, in welcher Reihenfolge die Ergebnisdaten angezeigt werden sollen und welche Operationen auf diesen Daten möglich sind:

```
<overviewdialog identifer="CLVW">
  <table type="de.rt2.b3.clvw.bo.Clip"
         editable="true" deletable="true"
         creatable="true">
    <column field="ID" width="80"/>
    <column field="BoNummer" width="80"/>
    <column field="Titel" width="250"/>
  </table>
</overviewdialog>
```

Die folgende Abbildung zeigt die Tabelle mit den Suchergebnissen:

1 Die Funktionsweise einer *Query Engine* ist in einem Design Pattern beschrieben, erhältlich unter http://www.tim-wellhausen.de/papers/QueryEngine.pdf

Die Ergebnisdarstellung

In dieser Tabelle können Daten wie in einer Tabellenkalkulation geändert werden. Die geänderten Geschäftsobjekte werden über den Persistenzdienst des Applikationsservers in die Datenbank zurückgeschrieben. Der Persistenzdienst ist generisch aufgebaut und kann die Klassen aller Teilapplikationen verarbeiten.

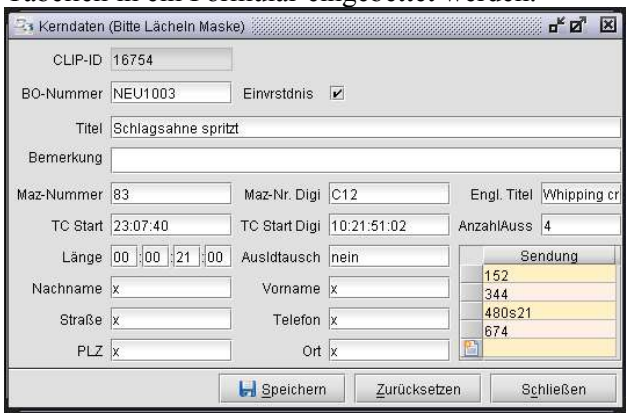
Eine weitere konfigurierbare Komponente übernimmt den Aufbau und die Steuerung der *Formulare*. Diese Komponente wird mit einer Beschreibung der Bestandteile eines Formulars initialisiert. Sie baut das Formular dynamisch auf, indem sie alle Eingabefelder erzeugt und entsprechend der Konfigurationsdaten anordnet. Einen Ausschnitt der statischen Konfigurationsdaten für das Kerndatenformular zeigt der folgende Abschnitt:

```
<businessobjectdialog identifer="CLVW">
  <form type="de.rtl2.b3.clvw.bo.Clip">
    <panel type="columns">
      <column width="medium">
        <numberfield id="Id"/>
        <textfield id="Titel" width="3"/>
        <textarea id="Bemerkung"width="3"/>
        <timecode id="Laenge"/>
      </column>
    </panel>
  </form>
</businessobjectdialog>
```

Der Applikationsserver erweitert diese XML-Daten zur Laufzeit mit Informationen aus der Datenbank. Dazu gehören die Namen und Tooltipps der Felder sowie Pflichtfeld- und Berechtigungsinformationen.

Damit ein Geschäftsobjekt bearbeitet werden kann, kopiert die Komponente die Daten des Geschäftsobjekts in die Eingabefelder des Formulars. Nachdem ein Anwender seine Änderungen abgeschlossen hat, schreibt die Komponente die geänderten Daten in das Geschäftsobjekt zurück und schickt es zum Speichern an den Persistenzdienst des Applikationsservers.

Wie in der folgenden Abbildung des Kerndatenformulars zu sehen ist, können neben einfachen Eingabefeldern für Texte und Zahlen auch komplexere Elemente wie Timecode-Eingaben und Tabellen in ein Formular eingebettet werden.



Das Kerndatenformular

Resümee

Das Projekt „B3“ begann mit einem Kernteam von drei Personen, das innerhalb eines Jahres auf acht Personen vergrößert wurde. In dieser Zeit wurde die Architektur der neuen Plattform einschließlich aller konfigurierbaren Komponenten und Dienste fertig gestellt. Weiterhin sind das Portal und insgesamt sechs Teilapplikationen in den produktiven Betrieb übergegangen. Zwei weitere Teilapplikationen sind in Entwicklung: ein Informationssystem mit mehreren Dutzend Formularen und ein komplexes Planungssystem.

Der intensive Einsatz von Konfigurationsdaten hat redundanten Programmcode weitestgehend vermieden, wodurch die Teilapplikationen bedeutend robuster sind als zuvor. Da alle Standardfunktionen im Kern der Plattform umgesetzt sind, konnte die Entwicklungszeit der Teilapplikationen reduziert werden. Durch die zentrale Konfiguration der Applikationen ist schließlich auch der produktive Betrieb vom Deployment bis zur Fehlerbehebung deutlich effizienter geworden.

Tim Wellhausen arbeitet freiberuflich als Softwarearchitekt und Berater in München. Sein Schwerpunkt ist die Entwicklung von Client- und Server-Applikationen in Java.
E-Mail: kontakt@tim-wellhausen.de.