04-2014 | Winter | www.ijug.eu

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

Aus der Community – für die Community

Java aktuell

Mit Lambda-Ausdrücken einfacher programmieren

Go for the Money
Währungen und Geldbeträge in Java

Oracle-Produkte
Die Vorteile von Forms und Java vereint

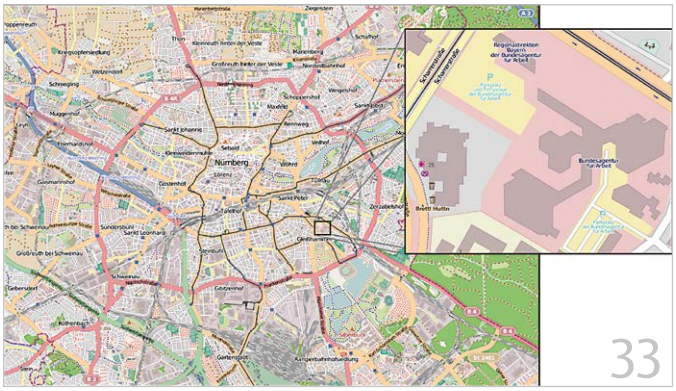
Pimp my Jenkins
Noch mehr praktische Plug-ins

Sonderdruck

D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

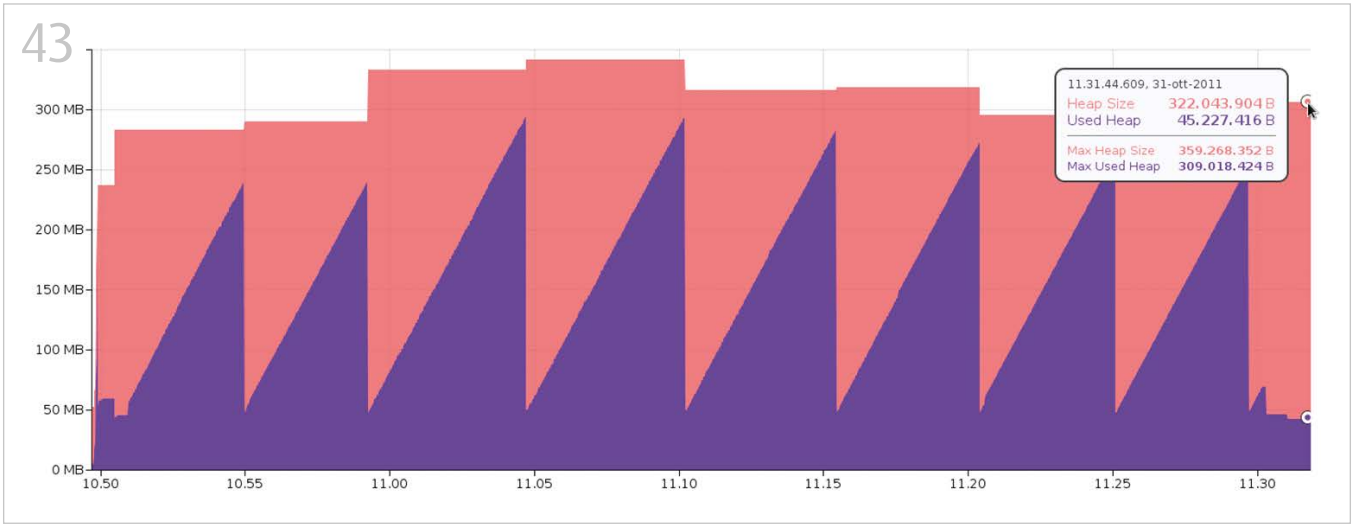
4 191978 304903 04

iJUG Verbund



OpenStreetMap ist vielen kommerziellen Diensten überlegen, Seite 33

Alles über Währungen und Geldbeträge in Java, Seite 20



WURFL ist verglichen mit selbst dem nicht reduzierten OpenDDR-Vokabular deutlich speicherhungriger, Seite 43

Schnell entwickeln – die Vorteile von Forms und Java vereint

René Jahn, SIB Visions GmbH

Mit Forms hat Oracle vor vielen Jahren eines der ersten Produkte für das Rapid Application Development (RAD) auf den Markt gebracht. Damit konnten innerhalb kürzester Zeit mittels PL/SQL-Syntax klassische ERP-/Datenbank-Anwendungen entwickelt werden. Das Produkt wird weltweit von unzähligen Unternehmen eingesetzt und ist nach wie vor nicht wegzudenken.

In den letzten Jahren wurde es jedoch still um Forms, es gab zwar neue Versionen, aber keine neuen Features beziehungsweise Modernisierung. Die Community half sich selbst über die Runden und versuchte mithilfe von Java-Komponenten, neuen Schwung in die Applikationen zu bringen. Doch so richtig glücklich wurde die Forms-Community damit nicht, weil Java sehr komplex wirkt und die Entwicklung dadurch eher verlangsamt als beschleunigt wird. Doch da muss der Autor widersprechen: Mit Java kann genauso, wenn nicht sogar effizienter entwickelt werden als mit Forms. Wenn man die Vorteile beider Technologien vereint, entstehen innerhalb kürzester Zeit modernste Lösungen.

Über Forms

Das Produkt „Forms" wurde erstmals Ende des Jahres 1980 veröffentlicht. Die ersten Versionen waren noch zeichenorientiert (bis Forms 3) und hatten kein GUI, wie wir es gewohnt sind. Im Laufe der Jahre erfolgten zahlreiche Versionssprünge (6/6*i*, 9*i*, 10*g*, 11*g*), Verbesserungen und Technologie-Wechsel. Die letzte wirklich große Veränderung war jedoch der Wechsel von einer Client/Server- zu einer Drei-Tier-Architektur, auch „Web Forms" genannt. Seit dieser Umstellung laufen Forms-Anwendungen als Java-Applet im Webbrowser und die Unterstützung für den Client/Server-Betrieb wurde eingestellt.

Die Beziehung zwischen Oracle Forms und Java besteht also schon seit einigen Jahren und selbst das Forms-GUI läuft in Java. Seitdem ist es auch möglich, eigene Java-Komponenten in Forms zu integrieren, sogenannte „Pluggable Java Components" (PJC) oder „Java Beans".

Die enormen Vorteile von Forms sind die Entwicklung mit PL/SQL-Syntax, die nahtlose Integration der Oracle-Datenbank und die direkte Verwendung von SQL. Somit kann mit einer einzigen Programmiersprache eine komplexe Datenbank-Lösung mit angebundenem GUI entwickelt werden. Doch das in die Jahre gekommene Forms hat auch wesentliche Nachteile: Der Forms Builder, die Entwicklungsumgebung für Forms-Applikationen, ist alles andere als zeitgemäß und wurde auch schon etliche Jahre nicht mehr weiterentwickelt. Sie bietet zwar einen WYSIWYG-Editor, doch es fehlen grundlegende Dinge wie ein Layout-Manager. Die Anbindung von REST, Web-Services oder die Interaktion mit Schnittstellen ist zwar technisch kein Problem und auch lösbar, jedoch weitaus komplexer und zeitaufwändiger, als dies mit Java der Fall wäre.

Am Ende ist Forms zwar ein effizientes und mächtiges Werkzeug, aber angestaubt, proprietär und kaum ansprechend für junge Entwickler. Der fehlende Nachwuchs an Entwicklern wird auch zunehmend zu einem Problem für Forms-Kunden.

Ergänzung durch Java

An dieser Stelle kommt Java ins Spiel, denn es bringt genau das mit, was Forms fehlt. Es ist offen, flexibel, modern und spricht die jungen Entwickler an. Doch Java ist grundsätzlich nur eine Programmiersprache, mit der man beliebige Aufgabenstellungen umsetzen kann. Um die Vorteile von Java in die Forms-Welt zu übernehmen, benötigt man Frameworks, die ähnlich wie Forms funktionieren. Andernfalls können zwar beide Technologien vereint werden, aber die Forms-Entwickler werden sich mit Java nicht anfreunden. Es ist auch eher unrealistisch, dass ein Forms-Entwickler ohne die von Forms bekannten Möglichkeiten mit Java arbeitet. Wenn man beides erfolgreich kombinieren möchte, muss auch der Funktionsumfang vereinbar sein.

ADF oder Apex

Die Vorteile von Java sind natürlich auch Oracle bekannt und so wurde schon früh begonnen, an einem möglichen Nachfolger für Forms zu arbeiten. Das Application Development Framework (ADF) wurde unter anderem auch dafür entwickelt. Es handelt sich dabei um ein auf Java-EE basierendes Framework für die Erstellung von Web-Anwendungen. So vollständig und mächtig ADF auch ist, so groß ist auch die Hürde der Einarbeitung für Forms-Entwickler. Denn mit ADF verlässt man PL/SQL und begibt man sich in die komplexe Welt von Java EE. Auch die gewohnten Entwicklungstools müssen gewechselt werden, denn ohne JDeveloper würde man auf Effizienz verzichten. Es ist auch nicht daran zu denken, die Forms-Entwicklung durch eine Java-Entwicklung zu ersetzen, denn das Applikations-Know-how steckt in den Forms-Entwicklern. Das Problem einer Integration oder gar einer Migration ist daher nicht nur technischer Natur.

Ein weiteres Produkt, das als Forms-Nachfolger infrage kommt, wäre Apex. Der große Vorteil dabei ist die kleinere

Technologie-Hürde, denn mit Apex bleibt man im gewohnten Umfeld von PL/SQL. Es können ebenfalls ansprechende Web-Anwendungen umgesetzt werden – auch komplett ohne Java. Bei komplexeren Aufgabenstellungen führt aber wohl kein Weg an JavaScript vorbei, denn interaktive Web-Anwendungen ohne JavaScript sind nicht denkbar. Die Vorteile von Apex liegen auf der Hand und dennoch ist dieses Produkt kein vollständiger Ersatz für Oracle Forms. Forms ist weitaus umfangreicher und optimierter für Geschäftsanwendungen hinsichtlich Datenmengen und Eingabeverhalten der Benutzer.

Es ist zwar immer stark abhängig von den Anforderungen an eine Geschäftsanwendung, aber in vielen Fällen führen weder ADF noch Apex zum gleichen Ergebnis wie Forms. Wenn Forms also gesetzt ist, sollte auf keinen Fall auf Java verzichtet werden, denn dadurch erhält man Ersatz für fehlende Features und gewinnt vor allem durch die Offenheit.

Forms und Java vereint

Wie bereits erwähnt, lässt sich mit Java nahezu jedes Problem lösen. Ohne geeignete Frameworks ist man jedoch nicht so effizient wie mit Forms. Zu viele Frameworks dürfen es aber auch nicht sein, denn sonst wirkt Java wieder zu kompliziert. Außerdem schafft man Abhängigkeiten, auf die man gerne verzichten würde. Daher sollte die Wahl auf ein Full-Stack-Framework fallen, das allerdings für Desktop-Applikationen geeignet ist. Ein reines Framework für die Entwicklung von Web-Anwendungen ist keine Option, um beide Welten zu vereinen.

Das Open-Source-Framework JVx [1] ist ein interessanter Kandidat, weil es ähnlich wie Forms funktioniert (Events, Trigger, LOVs), weil es ohne Abhängigkeiten auskommt und weil die Nähe zur Datenbank ein großer Vorteil ist. Es ist außerdem auch GUI-Technologie-unabhängig, da es den Single-Sourcing-Ansatz verfolgt. Damit lässt sich eine einmalig entwickelte Applikation am Desktop, im Webbrowser oder als native App am Smartphone betreiben, ohne den Sourcecode zu verändern. Das ist sicherlich auch für Oracle-Forms-Anwendungen eine neue Perspektive und ermöglicht den einfachen Einstieg in die Web- und Mobile-Entwicklung.

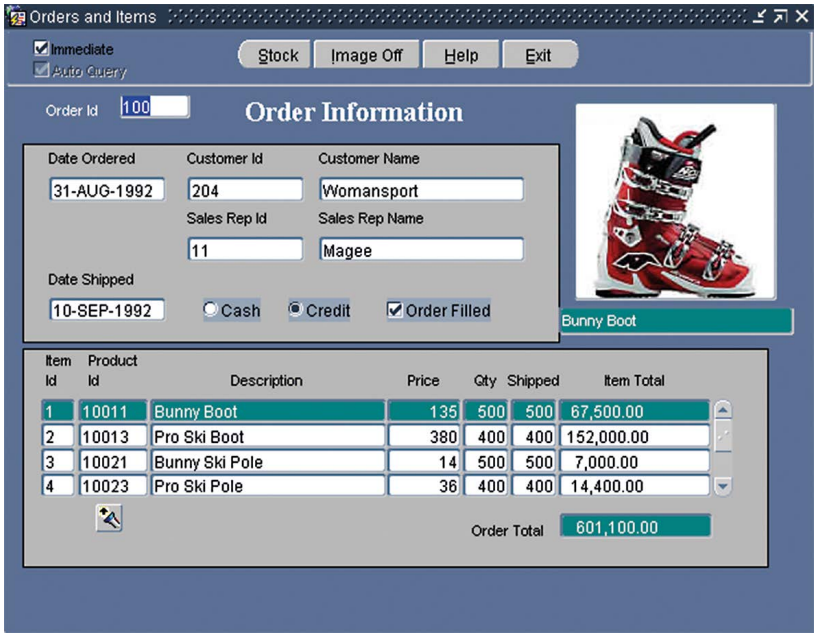


Abbildung 1: Order-Information mit Oracle Forms

Ein konkretes Beispiel: Die offizielle Summit-Demo-Applikation [2] von Oracle, die für Forms und ADF zum Download angeboten wird, verwaltet Kunden, Sportartikel und Angebote. Es existieren mehrere Bearbeitungsmasken, doch wir konzentrieren uns ausschließlich auf die Maske „Order Information" (siehe Abbildung 1). Diese zeigt Bestellungen an. Eine Bestellung enthält den jeweiligen Kunden und alle Artikel inklusive Vorschaubild. Der Zahlungsbetrag wird ausgewiesen und es können weitere Artikel zur Bestellung hinzugefügt werden.

Java-Umsetzung

Im ersten Schritt verwendet der Autor das gleiche Datenmodell und erstellt mit Java eine nahezu identische Maske. Diese kommt anschließend direkt in Oracle Forms anstatt der ursprünglichen Forms-Maske zum Einsatz, um die nahtlose Integration beider Welten zu zeigen. Abschließend wird die Maske auch im Webbrowser ohne Applet laufen.

Die Umsetzung der Java-Maske ist nicht sonderlich aufwändig, da sie keinerlei Logik beinhaltet. Die Codierung beschränkt sich auf die Datenbindung und das Layout. Auch die in Forms übliche Auswahl von Datensätzen mittels Sub-Masken kann entfallen, da Java-übliche Auswahllisten/ComboBoxen zum Einsatz kommen.

```
DBStorage dbsSOrd = new DB-
Storage();
dbsSOrd.setWritebackTable("S_
ORD");
dbsSOrd.
setDBAccess(getDBAccess());
dbsSOrd.open();

DBStorage dbsSItem = new DB-
Storage();
dbsSItem.setWritebackTable("S_
ITEM");
dbsSItem.
setDBAccess(getDBAccess());
dbsSItem.open();
```

Listing 1

Die Datenbindung für die Liste der Produkte ist mit wenigen Code-Zeilen erledigt. Zuerst definiert man den Zugriff auf die Tabelle in der Datenbank. Dafür stellt JVx die Klasse „DBStorage" zur Verfügung (siehe Listing 1). Diese übernimmt sämtliche CRUD-Vorgänge und es ist kein zusätzlicher Code erforderlich. Man liest alle Bestellungen („dbsSOrd") aus der Tabelle „S_ORD" und findet die zugehörigen Artikel („dbsSItem") in der Tabelle „S_ITEM".

Im nächsten Schritt geht es um die Anbindung der GUI-Tabelle an das Datenobjekt. Dazu bietet JVx mit der Klasse „RemoteDataBook" die Möglichkeit, auf das Datenobjekt standardisiert zuzugreifen

(siehe Listing 2). Für den Zugriff auf die Bestellungen („dbsSOrd“) kommt das DataBook „rdbSOrd“ und für den Zugriff auf die zugeordneten Artikel („dbsSItem“) das DataBook „rdbSItem“ zum Einsatz. Abschließend benötigt man noch ein GUI-Element für die Anzeige der Daten (siehe Listing 3). Dazu wird die Tabelle „navItems“ mit dem DataBook „rdbSItem“ verknüpft. Natürlich fehlt jetzt noch das Layout, aber darauf wird an dieser Stelle verzichtet, da es sich um üblichen GUI-Code mit Positionierung von Komponenten handelt. Die in Java entwickelte Maske sieht nun sehr ähnlich zum Original in Forms aus (siehe Abbildung 2) und deckt denselben Funktionsumfang ab.

Im Unterschied zu Forms bietet die Java-Variante in jeder Tabelle Sortier- und verschiebbare Spalten, Auswahllisten in- und außerhalb von Tabellen sowie Datensatz-Operationen (Einfügen, Löschen, Suche, Export) direkt bei den Daten.

Sieht aus wie Forms, ist aber Java

Bisher wurde in zwei unterschiedlichen Technologien die gleiche Anforderung umgesetzt. Das Besondere daran ist jedoch, dass die Java-Variante äußerst wenig Code benötigt und jeder Forms-Entwickler ohne große Schulung damit umgehen kann. Es sind auch keine Model-Definitionen sowie keine XML-Konfigurationen erforderlich und es ist mit zwei Java-Klassen eine saubere Drei-Tier-Lösung entstanden. Das Beste kommt natürlich zum Schluss, denn nun wird die Java-Maske in die bestehende Forms-Anwendung eingebettet, um die Vorteile von Java in Forms bereitzustellen. Es besteht dadurch die Möglichkeit, neue Anforderungen mit Java umzusetzen und dann in Forms zu integrieren, oder aber auch, die Forms-Applikation schrittweise durch Java zu modernisieren.

Die lauffähige Forms-Applikation ist voll funktionsfähig und bietet die gewohnten Möglichkeiten. Abbildung 3 zeigt die Java-Maske innerhalb einer Forms-Anwendung. Es wurde absichtlich der Forms-Applikationsrahmen belassen und das Fenster enthält einen zusätzlichen Close-Button, der direkt mit Forms hinzugefügt wurde.

Die Integration einer Java-Maske in Forms ist relativ einfach, denn Forms erlaubt die Einbindung von PJC oder Java Beans. Abbildung 4 zeigt, wie zu einem be-

```
RemoteDataBook rdbSOrd = new RemoteDataBook();
rdbSOrd.setName("order");
rdbSOrd.setDataSource(getDataSource());
rdbSOrd.open();

RemoteDataBook rdbSItem = new RemoteDataBook();
rdbSItem.setName("item");
rdbSItem.setDataSource(getDataSource());
rdbSItem.setMasterReference(
   new ReferenceDefinition(new String[] { "ORD_ID" },
                           rdbSOrd, new String[] { "ID" }));
rdbSItem.open();
```

Listing 2

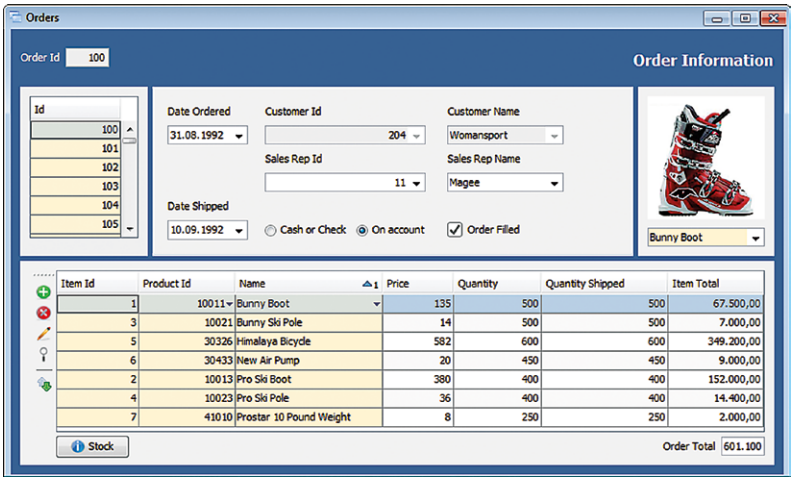


Abbildung 2: Order-Information mit Java

```
NavigationTable navSItem = new NavigationTable();
navSItem.setDataBook(rdbSItem);
```

Listing 3

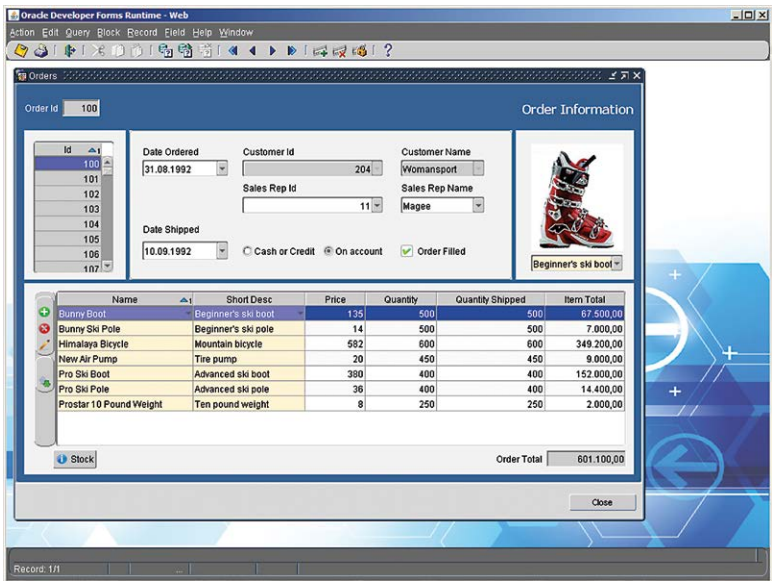


Abbildung 3: Order-Information mit Java, eingebettet in Forms

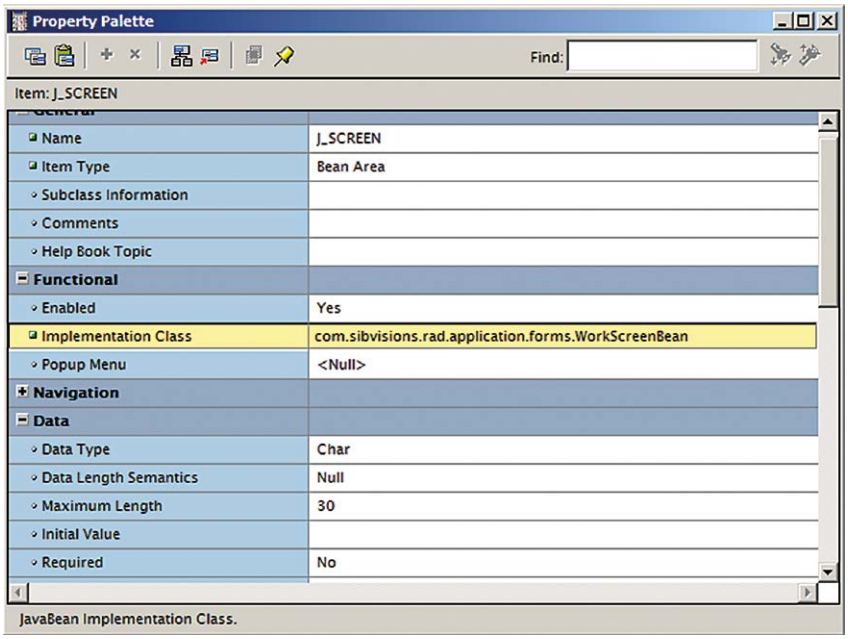


Abbildung 4: Java-Bean-Konfiguration

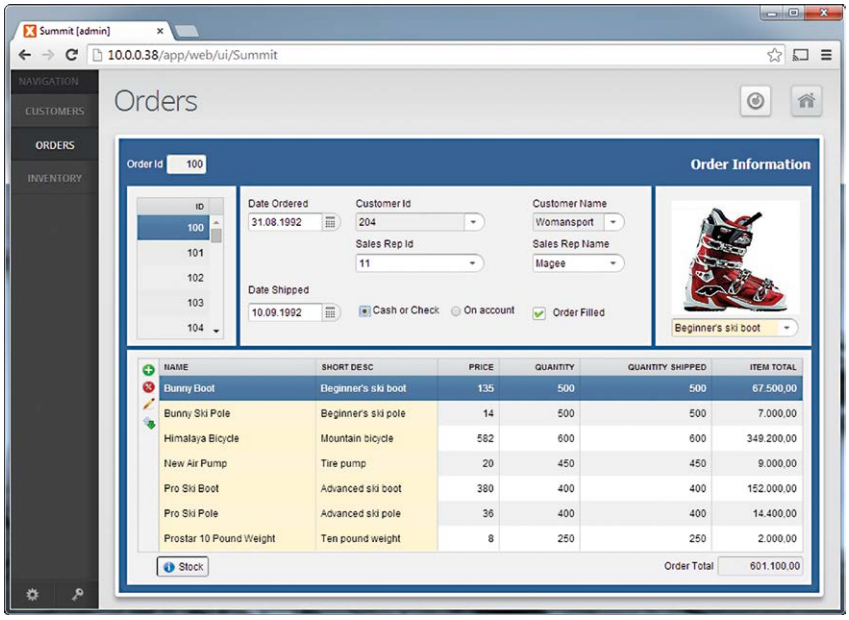


Abbildung 5: Das Ergebnis

reits vorhandenen Canvas eine Java Bean hinzugefügt und die Java-Klasse konfiguriert wurde.

Fazit

Die Integration von Java in Forms bietet vollkommen neue Möglichkeiten für die Forms-Entwicklung. Es stehen beispielsweise ansprechende „Look & Feels" zur Verfügung, um die Applikation aufzupeppen. Auch die Anzahl der GUI-Controls (Tree, TreeTable, Charts, Kalender) ist um ein Vielfaches höher.

Es gibt aber auch einen Produktivitätsgewinn durch die automatische Unterstützung von Mehrsprachigkeit, Layout-Manager für die automatische Größenanpassung von Masken, Drag & Drop, Upload- und Download von Dateien ohne Zusatzaufwand, einfache Integration von Open-Source-Reporting-Tools wie BIRT [3] oder JasperReports [4].

Durch den Einsatz von JVx sind noch weitere Anwendungsfälle möglich, die ansonsten nur mit erheblichem Mehraufwand realisierbar gewesen wären. Eine Applikation ist auf unterschiedlichen Plattformen und mit unterschiedlichen Technologien zukunftssicher einsetzbar. Als kleine Demonstration wird die zuvor entwickelte Java-Maske – ohne Sourcecode-Änderung – im Webbrowser als HTML5-Anwendung betrieben (siehe Abbildung 5). Durch die UI-Abstraktion von JVx könnte das Backend als Forms-Applikation laufen, das Frontend als HTML5-Applikation und für Benutzer, die ständig unterwegs sind, ist auch eine Smartphone-Lösung möglich. Aber unabhängig von der eingesetzten Technologie muss am Ende des Tages eine Lösung entstehen, die übersichtlich, wartbar und auch in zehn Jahren noch lauffähig ist. Genau das ist möglich, wenn Forms und Java vereint im Einsatz sind.

Links

[1] JVx: http://sourceforge.net/projects/jvx
[2] Summit Applikation: https://blogs.oracle.com/jdevotnharvest/entry/adf_summit_sample_application_a
[3] BIRT: http://www.eclipse.org/birt
[4] JasperReports: http://www.jaspersoft.com

René Jahn
rene.jahn@sibvisions.com

René Jahn ist Mitbegründer der SIB Visions GmbH und Head of Research & Development. Er verfügt über langjährige Erfahrung im Bereich der Framework- und API-Entwicklung. Sein Interessenschwerpunkt liegt auf der Integration von State-of-the-Art-Technologien in klassische Business-Applikationen. Darüber hinaus betreut er die Open-Source-Sparte bei SIB Visions und veröffentlicht regelmäßig Artikel im Unternehmensblog.

http://ja.ijug.eu/14/4/11