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

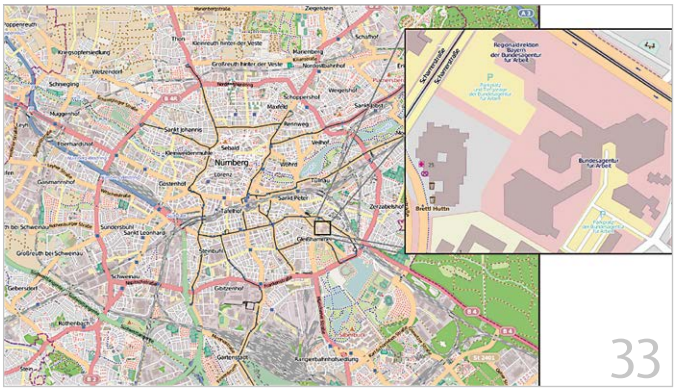
Pimp my Jenkins
Noch mehr praktische Plug-ins

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

5 Das Java-Tagebuch
Andreas Badelt, Leiter der DOAG SIG Java

8 JDK 8 im Fokus der Entwickler
Wolfgang Weigend

15 Einmal Lambda und zurück – die Vereinfachung des TestRule-API
Günter Jantzen

20 Go for the Money – eine Einführung in JSR 354
Anatole Tresch

24 Scripting in Java 8
Lars Gregori

28 JGiven: Pragmatisches Behavioral-Driven-Development für Java
Dr. Jan Schäfer

33 Geodatensuche und Datenanreicherung mit Quelldaten von OpenStreetMap
Dr. Christian Winkler

38 Pimp my Jenkins – mit noch mehr Plug-ins
Sebastian Laag

41 Apache DeviceMap
Werner Kei

46 Schnell entwickeln – die Vorteile von Forms und Java vereint
René Jahn

50 Oracle-ADF-Business-Service-Abstraktion: Data Controls unter der Lupe
Hendrik Gossens

54 Neue Features in JDeveloper und ADF 12*c*
Jürgen Menge

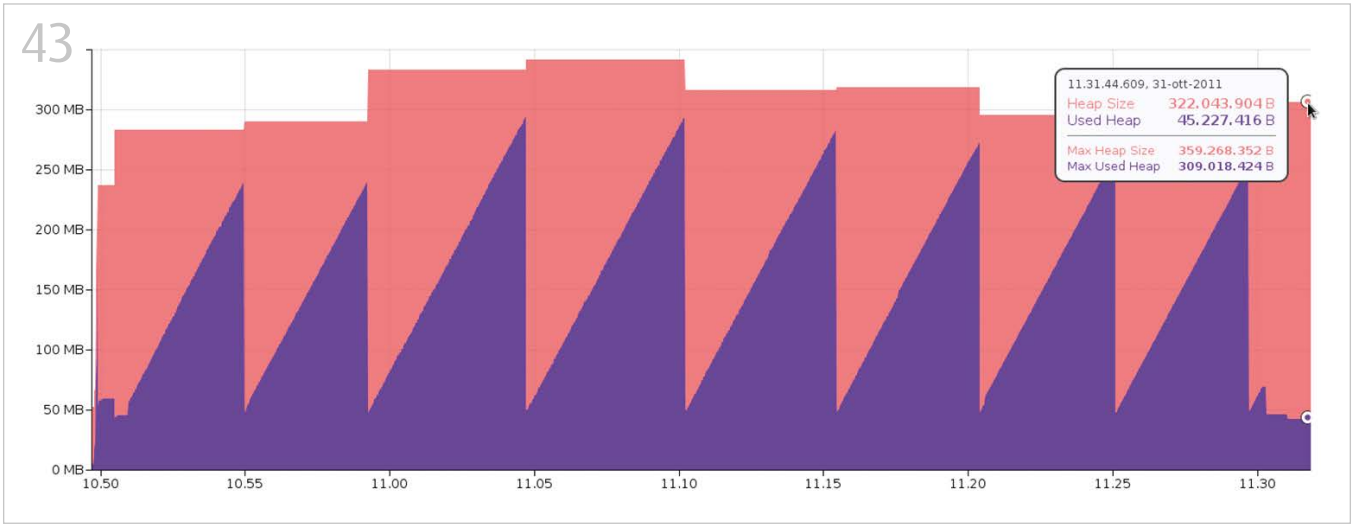
57 Der (fast) perfekte Comparator
Heiner Kücker

60 Clientseitige Anwendungsintegration: Eine Frage der Herkunft
Sascha Zak

64 Unbekannte Kostbarkeiten des SDK Heute: Die Klasse „Objects“
Bernd Müller

66 Inserenten

66 Impressum



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

Neue Features in JDeveloper und ADF 12*c*

Jürgen Menge, Oracle Deutschland B.V. & Co. KG

Um eine integrierte Entwicklungsumgebung für die Fusion Middleware 12c zur Verfügung zu stellen, bietet die neue Version von JDeveloper und ADF zahlreiche neue Features.

JDeveloper ist eine von drei Java-Entwicklungsumgebungen, die mit dem Hause Oracle verbunden sind. Im Unterschied zum Oracle Enterprise Pack for Eclipse (OEPE) und NetBeans verantwortet Oracle allein die Weiterentwicklung des JDeveloper. Grund dafür ist, dass Oracle JDeveloper für die Entwicklung der Fusion Applications und der Fusion Middleware nutzt. Er steht darüber hinaus auch Partnern und Kunden zur Verfügung, die die Fusion Middleware einsetzen beziehungsweise eigene Applikationen entwickeln wollen.

Bei der Entwicklung von Java-Anwendungen kommt das Oracle Application Development Framework (ADF) zum Einsatz. Es integriert verschiedene Standards (wie EJB/JPA, JAX-WS, JAX-RS) und ergänzt diese um eigene Implementierungen und Frameworks (wie ADF Business Components, ADF Faces). Typische Web-Anwendungen auf Basis von Oracle ADF setzen im Business Service Layer wahlweise das Framework ADF Business Components oder EJB/JPA und im View Layer ADF Faces mit dem zugehörigen ADF Controller ein (siehe Abbildung 1).

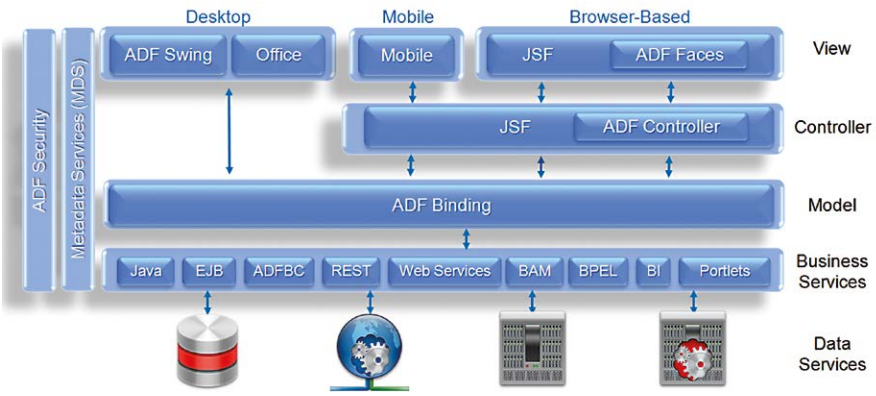


Abbildung 1: Oracle-ADF-Architektur

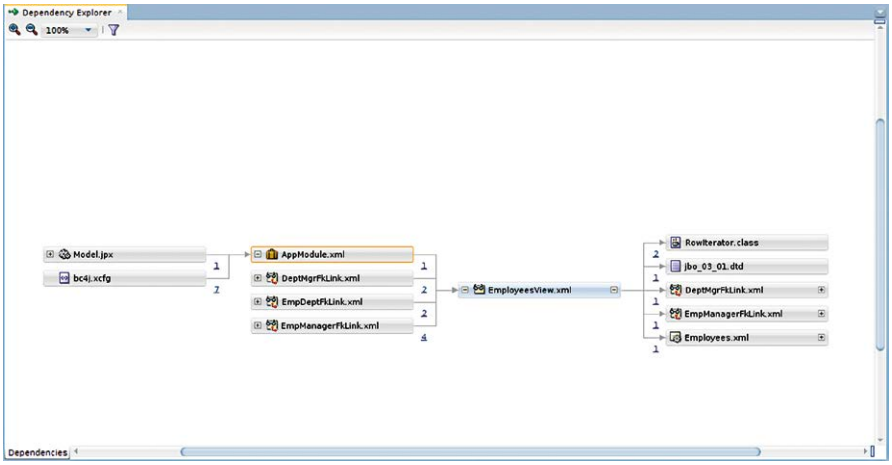


Abbildung 2: Dependency Viewer im Oracle JDeveloper

Oracle ADF muss für die Laufzeitumgebung lizenziert sein. Mit ADF Essentials gibt es allerdings eine kostenfreie Edition mit einem eingeschränkten Funktionsumfang.

JDeveloper unterstützt als Entwicklungsumgebung den vollen Funktionsumfang von ADF, wohingegen OEPE nur jene Bestandteile von ADF unterstützt, die für Java EE-Entwickler interessant sind.

Ende Juni 2014 ist die Version 12.1.3 des JDeveloper zeitgleich mit dem OEPE 12.1.3 und einer großen Zahl von Fusion Middleware-Produkten erschienen. Mit dieser Version steht auch das Mobile Application Framework (MAF) 2.0 als Extension sowohl für JDeveloper als auch für Eclipse (OEPE) zur Entwicklung von hybriden Mobil-Applikationen zur Verfügung. Der Artikel gibt einen Überblick über die im aktuellen Release 12*c* verfügbaren neuen Funktionalitäten, der aufgrund von deren Vielzahl nicht den Anspruch auf Vollständigkeit erfüllt. Für Entwickler besteht die Möglichkeit, den JDeveloper sowie OEPE kostenfrei aus dem Oracle Technology Network (siehe „http://www.oracle.com/technetwork/developer-tools“) herunterzuladen und zu installieren, um mit den neuen Möglichkeiten praktisch zu arbeiten.

Neue JDeveloper-Features

Oracle JDeveloper ist komplett in Java geschrieben und verwendet in der Version 12*c* das JDK7. Er nutzt das Modulkonzept von OSGi, sodass benötigte Funktionen erst bei Bedarf geladen werden. Für Testzwecke ist ein WebLogic Server 12*c* in die

IDE integriert. Ab JDeveloper 12.1.3 kann die Java SE 8 zum Compile und Test von Applikationen genutzt werden.

Die Teams von JDeveloper und NetBeans tauschen Entwicklungsergebnisse aus. So wird im JDeveloper 12*c* das Windowing-System von NetBeans verwendet, das viele Verbesserungen für den Benutzer bringt. Ebenfalls neu ist der Dependency Viewer, in OEPE unter dem Namen „AppXray" bekannt. Er zeigt die Abhängigkeiten zwischen Artefakten einer Applikation in grafischer Form an (siehe Abbildung 2).

Web Development

Mit dem JDeveloper 12*c* wird die Entwicklung von HTML5 und CSS 3 durch entsprechende Editoren unterstützt. Ebenfalls neu ist die Unterstützung von Java EE 6, also Standards wie EJB 3.1, Servlet 3.0, CDI, JPA 2.0, EL 2.2 (siehe Abbildung 3).

Neben der bestehenden Funktionalität für JAX-WS Services kommen in JDeveloper 12*c* folgende Funktionen für RESTful Web Services hinzu:

- Generierung von RESTful Services (1.1, 2.0) aus annotierten POJOs oder aus einer WADL-Datei (Web Application Description Language)
- Generierung eines JAX-RS Client Proxy für RESTful Web Services
- Generierung eines Web Service Data Controls für RESTful Web Services. Data Controls können vom UI-Entwickler per „Drag & Drop" verwendet werden, um auf den Service zuzugreifen.

Die Unterstützung für JAX-RS ist besonders für Applikationen wichtig, die für mobile Endgeräte entwickelt werden.

ADF Business Components

Mithilfe des Frameworks ADF Business Components (ADFbc) können Datenzugriffe auf eine relationale Datenbank und die Geschäftslogik des Business Service implementiert werden. Mit JDeveloper 12*c* sind einige interessante neue Funktionen dazugekommen:

- Für die Arbeit mit den Business Components muss keine Datenbank online im Zugriff sein; im Offline-Modus kommt ein lokales Abbild des Datenbank-Schemas in der Applikation zum Einsatz.
- Das Standard-Mapping von Oracle- auf Java-Datentypen wurde in „Java Extended for Oracle" geändert, um das Exponieren von ADFbc-Komponenten als Web Service zu erleichtern.
- Groovy kann in ADFbc benutzt werden, um Validierungen oder abgeleitete Attribute zu definieren. Mit dem aktuellen Release stehen dafür sowohl ein Editor als auch die Möglichkeit des Code-Debuggings zur Verfügung.
- ADF Business Components können als RESTful Web Services exponiert werden.

Standard EJB/JPA

Anstelle von ADFbc können Entwickler den Standard EJB/JPA für den Datenzugriff nutzen. Allerdings gab es im JDeveloper bisher keine weitgehende Unterstützung einer deklarativen Arbeitsweise. Mit JDeveloper 12*c* hat es hier eine Angleichung des Funktionsumfangs gegeben:

- Mit Sparse Bean Data Control lassen sich Modell-getriebene Wertelisten (LOV) definieren.
- Der Modus für den Datenzugriff über das Data Control ist durch Annotatio-

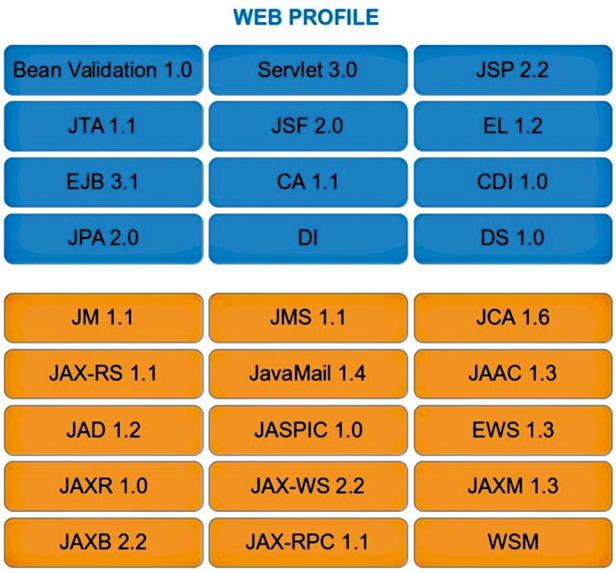


Abbildung 3: Java EE 6 mit Web Profile

```
import oracle.adf.model.adapter.bean.annotation.AttributeHint;
import oracle.adf.model.adapter.bean.annotation.ControlHintType;
import oracle.adf.model.adapter.bean.annotation.DateFormatter;
import oracle.adf.model.adapter.bean.annotation.FormatterType ;
import oracle.adf.model.adapter.bean.annotation.TimeZoneID

@AttributeHint  ( label = "Hire Date", tooltip = "Type date in the
form MM/dd/YYYY",
                  display = true, controlType = ControlHintType.
DEFAULT, width = 40, height = 20,
                  autoSubmit = true)

@DateFormatter ( type = FormatterType.SIMPLE_DATE, format = "MM/
dd/YYYY", formatter = "",
                timezoneId = TimeZoneID.DEFAULT)

public Date getHireDate() { return hireDate; }
```

Listing 1

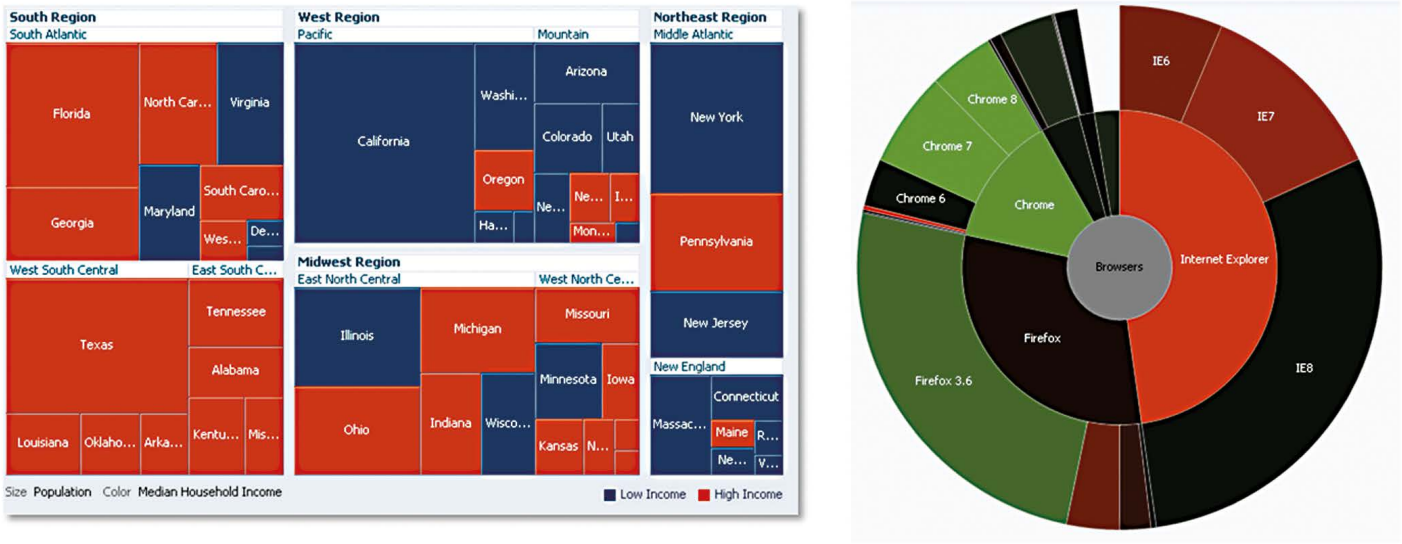


Abbildung 4: Treemap und Sunburst

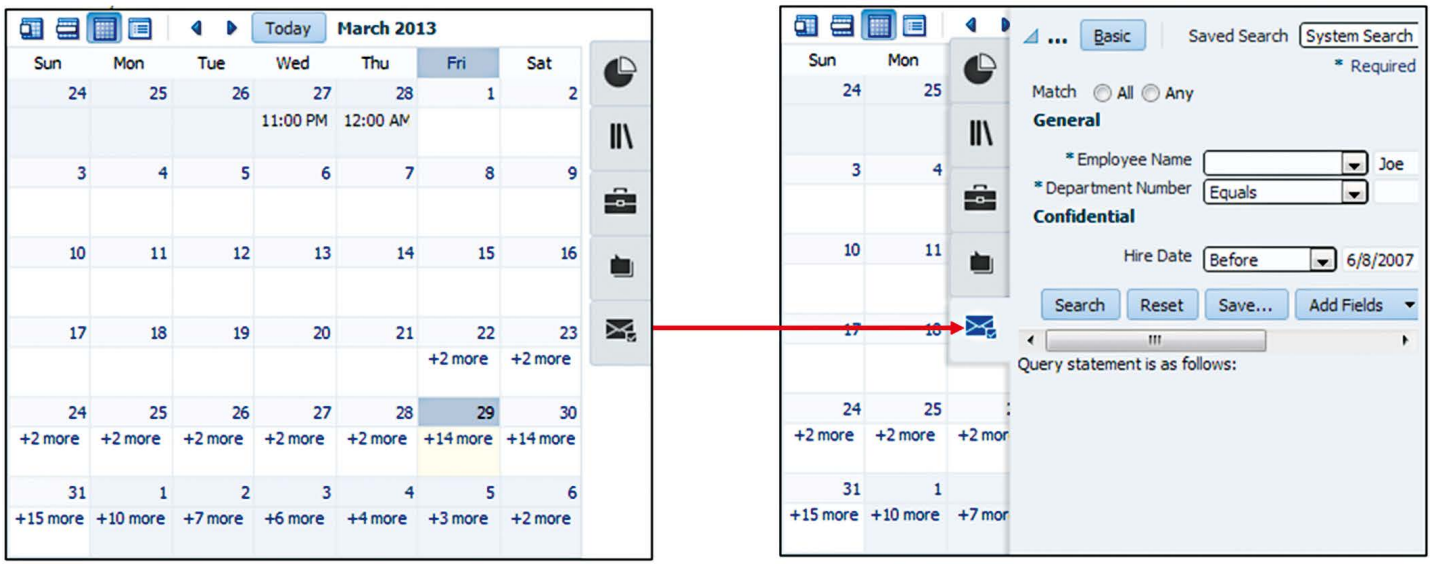


Abbildung 5: PanelDrawer

nen oder deklarativ einstellbar (Scrollable, Range Paging, NoPagination).
- Metadaten für die Darstellung im UI (Labels, Hilfe, Tooltips etc.) sind über Annotationen bereit (siehe Listing 1).

ADF Faces

Mit JDeveloper 12*c* wurde auf die Version 2 des JSF-Standards umgestellt, der als Neuerungen unter anderem Facelets und AJAX-Support mitbringt. Bei bestehenden ADF-Applikationen der Version 11g werden bestehende Pages von JSPX auf die neue JSF-Syntax automatisch migriert.

Mit ADF Faces steht dem UI-Entwickler eine große Zahl von Oberflächen-Komponenten zur Verfügung, die ein in unterschiedlichen Browsern getestetes „Look & Feel" einschließlich des Bedienkonzepts (Usability) mitbringen.

In jedem Release des JDeveloper kommen neue Oberflächen-Komponenten dazu. Dies ist auch in den Releases 12.1.2 beziehungsweise 12.1.3 der Fall. Bestehende Komponenten, die bisher in Flash implementiert waren, wurden in HTML5 umgeschrieben. Einige Beispiele für die neuen Komponenten im Release 12.1.3 sind:

- PanelGridLayout erlaubt die Definition einer Gitterstruktur im Layout und ähnelt damit den traditionellen Tabellen beim HTML Design.
- Treemap und Sunburst sind neue Chart-Typen, die die Bibliothek der DVT-Komponenten (Data Visualization Tools) erweitern (siehe Abbildung 4).
- PanelDrawer und PanelSpringboard erlauben eine komfortable Navigation und sind besonders für Applikationen auf Mobilgeräten geeignet (siehe Abbildung 5).
- Die Diagram-Komponente kann Daten in Form von Knoten und Beziehungen

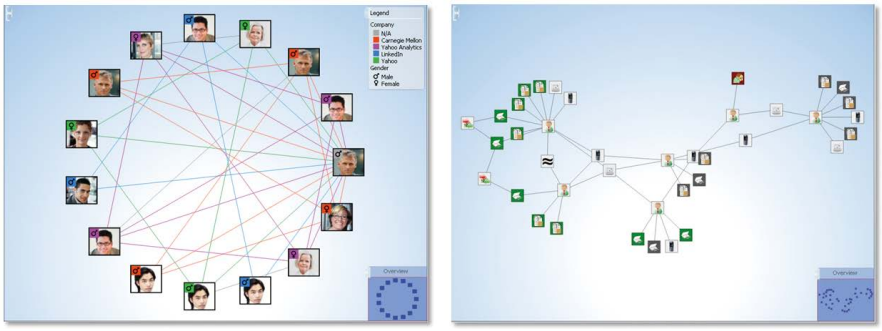


Abbildung 6: Diagram

darstellen und bietet großen Freiraum in Bezug auf das Layout (siehe Abbildung 6).

Für viele Komponenten ist die Bedienung durch Gesten hinzugekommen, um dem wachsenden Bedarf an mobilen Applikationen gerecht zu werden. Eine Live-Darstellung aller zur Verfügung stehenden Komponenten findet man unter „http://jdevadf.oracle.com".

Mit dem JDeveloper 12*c* sind neue Skins (Neuer Default Skin: „Skyros") hinzugekommen. Der grafische Skin.Editor hat einige Verbesserungen erfahren. Zudem wurde Maven 3 als Standard Build System integriert. Applikationen können bereits beim Anlegen in einem Maven Repository verwaltet oder zu einem späteren Zeitpunkt in ein solches überführt werden. Neben der Untersützung für SVN und anderen verbreiteten Systemen kommt mit dem JDeveloper 12*c* git/GitHub als weiteres populäres Versionskontrollsystem hinzu.

Weitergehende Informationen

- Oracle JDeveloper 12*c* (12.1.2.0.0.): www.oracle.com/technetwork/developer-tools/jdev/documentation/1212-nf-1964675.html
- Oracle JDeveloper 12*c* (12.1.3.0.0): www.oracle.com/technetwork/developer-tools/jdev/documentation/1213nf-2222743.html?msgid=3-10243408667

Dr. Jürgen Menge
juergen.menge@oracle.com

Dr. Jürgen Menge hat bei Oracle zunächst sechs Jahre in der Schulung als Trainer für die Entwicklungs-Werkzeuge Oracle Designer und Oracle Forms gearbeitet, um dann als technischer Berater in den Lizenzvertrieb zu wechseln. Seine fachlichen Schwerpunkte sind sowohl die klassischen Entwicklungs-Werkzeuge als auch die neuen, Standard-basierten Tools (JDeveloper/ADF, BI Publisher etc.).

http://ja.ijug.eu/14/4/13

Der (fast) perfekte Comparator

Heiner Kücker, Freiberufler

Eine Liste ganz einfach sortieren, entsprechende Methode aufrufen, Comparator übergeben. Dieser Artikel zeigt, dass auch ein einfacher Comparator Verbesserungsmöglichkeiten birgt.

Jeder Java-Programmierer hat sicher schon mal so einen Comparator geschrieben (siehe Listing 1).

Wenn nach einer anderen „Property (Member)" unserer Bean sortiert werden soll, müssen der gesamte Code kopiert und die abzufragenden Properties angepasst werden. Gibt es mehrere Beans, sind außerdem die beiden Parameter-Typen der „compare"-Methode zu ändern, also bereits vier Code-Stellen – eine langweilige und fehleranfällige Angelegenheit.

Getter ohne Reflektion

Basis dieses Alternativ-Vorschlags ist ein Property-Accessor, der ohne Reflektion auskommt, in diesem Fall nur der lesende Anteil, der „Getter" (siehe Listing 2).

Die beiden Typ-Parameter (Generics) „B-(Bean)" und „P(Property)" bestimmen die jeweils beteiligten Typen. Der Sinn der „get"-Methode sollte klar sein. Dieses Interface wird für alle benötigten Properties (Member) implementieren (siehe Listing 3).

Die Getter-Objekte sind statisch, existieren also nur einmal je JVM. Durch die Typ-Parameter sind Namensverwechslungen