integriert diese in die Mapping-Dateien. Des Weiteren werden die neuen Mappings der bestehenden Konfiguration hinzugefügt. Trotzdem sollte man noch einmal einen prüfenden Blick auf die erstellten Dateien werfen, um mögliche Fehler gleich am Anfang zu beheben. Dabei hilft der Hibernate Editor, welcher auch Bestandteil des Plug-ins ist.

### Resumee

Der Hibernate Synchronizer bietet all jenen, die „mal schnell" eine kleine Anwendung mit Hibernate schreiben wollen, eine sehr gute Basis, um zügig einen Persistenz-Layer zu schreiben.

*Michael Plöd*

### Links & Literatur

[1] hibernatesynch.sourceforge.net

[2] Michael Plöd: Aus dem Schlaf erwacht. Hibernate 3: Was ist neu in der dritten Generation?, in *Java Magazin* 4.2005

[3] James Elliott: Working with Hibernate in Eclipse: www.onjava.com/pub/a/onjava/2005/01/05/hibernate.html

# Quantum

## Plug-in zur übersichtlichen Verwaltung der Datenbanken

Quantum bietet eine eigene Perspektive, um eine übersichtliche Verwaltung der Datenbanken zu ermöglichen. Die Perspektive teilt sich auf in eine Baumansicht, einen SQL Editor und eine Listenansicht. In der Baumansicht werden alle konfigurierten Datenbankverbindungen aufgelistet. Datenbankverbindungen – so genannte Bookmarks – können mittels eines New Bookmark Wizard hinzugefügt werden. Hier muss zunächst der zur Datenbank passende JDBC-Treiber ausgewählt werden. Sollte der entsprechende Treiber noch nicht bei Quantum angemeldet sein, kann man dies praktischerweise gleich von diesem Dialog aus nachholen. Das Zusammenstellen der JDBC Connection URL wird durch den Assistenten sehr erleichtert, da Quantum den Connection String automatisch anhand des ausgewählten Treibers und der vom Anwender eingegebenen Parameter zusammenfügt. Einen einmal eingerichteten Treiber kann man leider nachträglich nicht mehr konfigurieren, sondern muss ihn löschen und neu anlegen. Hier besteht noch Verbesserungsbedarf. Klappt man ein Bookmark auf, so wird automatisch eine Verbindung zu der entsprechenden Datenbank hergestellt. Mithilfe der Baumansicht kann man nun das Schema der Datenbank navigieren. Jede Tabelle bzw.

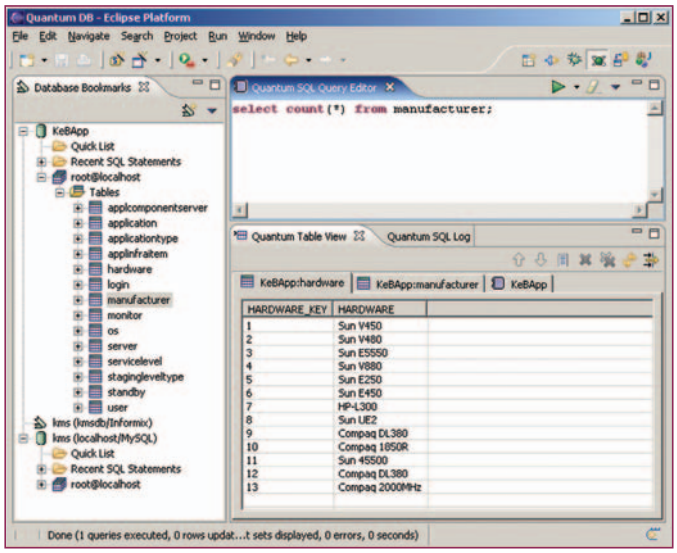


Abb. 1: Die Quantum-Perspektive

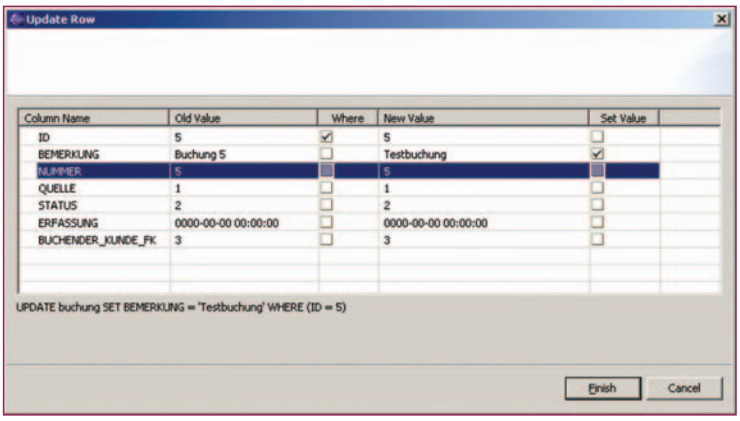


Abb. 2: Definition eines Update Statement

View wird durch einen eigenen Baumknoten dargestellt. Klappt man diesen Knoten auf, so zeigt Quantum die Attribute samt deren Typen und Feldgrößen an. Schlüsselspalten werden durch ein vorangestelltes Schlüsselsymbol gekennzeichnet. Damit man auch in umfangreichen Datenbankschema den Überblick nicht verliert, kann man die besonders interessanten Tabellen bzw. Views in eine so genannte Quick List aufnehmen.

Um die Daten einer Tabelle zu betrachten, öffnet man sie mit einem Doppelklick auf den entsprechenden Knoten in der Baumansicht. Nun werden die Daten der Tabelle in die Table View geladen. Eine direkte Bearbeitung der angezeigten Daten ist nicht möglich, jedoch enthält das Kontextmenü der Table View Einträge für das Einfügen, Löschen und Ändern von Datensätzen. Das Zusammenstellen der erforderlichen SQL Statements erfolgt recht komfortabel dialoggestützt (Abb. 2).

Für komplexere Abfragen steht ein SQL Query Editor zur Verfügung, der direkt über der Table View angezeigt wird. Hier können beliebige SQL-Kommandos erfasst und abgesetzt werden. Erfreulicherweise kann man die so entwickelten SQL-Skripte gleich in eine Textdatei exportieren. Der Import von bereits existierenden SQL-Skripten ist natürlich auch möglich. Der SQL-Editor muss auch für den Fall herhalten, dass man eine neue Tabelle oder eine View anlegen möchte, denn leider bietet Quantum hierfür keinen Wizard an.

### Fazit

Quantum ist ein recht praktisches Plug-in, das sich gut für den schnellen Blick in die Datenbank eignet. Insbesondere die einfache Konfiguration der Datenbankverbindungen trägt sehr zur Usability bei. Einziger Wermutstropfen ist das Fehlen von Funktionen zum Anlegen von Tabellen, Views und Stored Procedures. Hier muss man sich mit handgeschriebenen DDL-Anweisungen behelfen.

*Peter Friese*

### Links & Literatur

[1] quantum.sourceforge.net

# QuiGen

## Erstellen und Testen von Velocity Templates

Die Template Engine Velocity [1] der Apache Software Foundation entwickelt sich immer mehr zum Standard für Templategestützte Generierung. Das Open-Source-Plug-in QuiGen [2] bietet umfangreiche Unterstützung für das Entwickeln von Velocity Templates mit Eclipse. QuiGen kommt mit einer ganz eigenen Perspektive (Abb. 1). Die Perspektive bietet einen komfortablen Editor für Velocity Templates mit Syntax Highlighting und Code Completion für die VTL (Velocity Template Language). Die Outline View stellt die Struktur von Velocity-Direktiven in Baumdarstellung dar. Abbildung 2 zeigt das Syntax Highlighting, es stammt aus dem in QuiGen enthaltenen Plug-in VeloEclipse, welches zusätzlich Completion und Highlighting für HTML bereitstellt, was für die Generierung von Webseiten praktisch ist. So ist die QuiGen Website mit QuiGen selbst generiert worden.

### Kontext

In jedem Template gibt es Referenzen auf Daten, die zur Generierung von Ausgaben herangezogen werden. Um mit Velocity Ausgaben zu erzeugen, stellt man diese Daten in den so genannten Velocity Context. QuiGen ermöglicht mittels einiger mitgelieferter Document Models, geeignete Dateien aus Eclipse als Datenlieferanten für Velocity Templates zu verwenden. Dazu erweitert es Eclipse um eine QuiGen Context Details View. Diese zeigt für eine Template-Datei mit der Erweiterung *.vm* alle Variablen an, die im Velocity Context für diese Datei bereitgestellt werden (Abb. 3).

Der Clou: Für diese Variablen wird im Editor Content Assist unterstützt; gibt man also den Variablennamen in der Form *$variablenName* an, so wird mit Ctrl + Space automatisch eine Auswahl von Properties und Methoden der zur Variable korrespondierenden Java-Klasse angezeigt. Dies ist extrem hilfreich bei komplexen Templates, die beispielsweise aus dem Quelltext einer Java-Klasse mit Javadoc Annotations zusätzlichen Java-Code erzeugen. Ebenso hilfreich ist, dass aus dem Editor heraus sofort die Generierung angestoßen werden kann; die Ausgabe erfolgt in der Console View. Man kann also sofort testen, was aus dem Template erzeugt wird.

### Document Models

QuiGen bietet derzeit (Version 0.2.2) Unterstützung für folgende Eingabedateien: Java-Quelldateien, XML-Dateien, Java-