SimpleVal

Timm Albers

2. Oktober 2014

Contents

Einleitung **2**

Vorüberlegungen **2**

Problemstellung 2

Datenbank 2

Grafische Darstellung 2

Softwaredesign 2

Tests 3

Zusammenfassung 3

Implementierung **3**

Umgebung 3

Änderungen 3

Einleitung

Dies ist die Dokumentation des Programmes *SimpleVal*. *SimpleVal* stellt Messergebnisse grafisch dar. Im folgenden ist der Prozess der Entwicklung dokumentiert.

Vorüberlegungen

Problemstellung

Zu erstellen ist ein Programm, dass Messdaten grafisch darstellt. Außerdem sollen die Daten auch tabelarisch dargestellt werden. Die beiden Arten der Darstellung sollen parallel erfolgen.

Die Daten liegen in Form einer SQLite- oder MySQL-Datenbank vor.

Datenbank

Es ist uns überlassen, welche Datenbank wir verwenden möchten. Aufgrund der Geschwindigkeit von MySQL, werde ich dieses in meiner Implementierung verwenden. Es ist allerdings nicht förderlich, die Anwendung auf MySQL zu beschränken. Ich werde daher eine Art der Implementierung wählen, die es zulässt Implementierungen anderer SQL-Datenbanken im Nachhinein hinzuzufügen (z.B. abstrakte Klasse `DB`, erbende Klasse `MySQL` usw.).

Als Entwurfsmuster für die Implementierung der Datenzugriffe werde ich sowohl *Data Access Object / Data Transfer Object*[1] (ein Objekt kapselt den Datenzugriff; die zu verarbeitenden Daten werden durch Objekte repräsentiert), als auch *Eager Loading*[2] (die zu ladenden Daten werden möglichst früh geladen) verwenden. *Data Transfer Object* (DTO) ermöglicht einen einheitlichen und einfachen Datenzugriff, wobei die Software sehr modular gehalten werden kann, da für andere Datensätze nur neue DTOs implementiert werden müssen. *Eager Loading* sorgt hauptsächlich dafür, dass alle Daten beim Programmstart geladen werden. Dies soll dazu führen, dass sich die Benutzung des Programms flüssig anfühlt und ist in diesem Fall möglich, da wir es nicht mit großen Datenmengen zu tun haben. Der einzige Zugriff des Programms auf die Datenbank erfolgt somit direkt nach dem Programmstart.

Für den Zugriff auf die Datenbanken wird *sql2o*[3] verwendet. Die Library kapselt den Zugriff auf Datenbanken mittels *JDBC*[4].

Grafische Darstellung

Für die Darstellung der Daten in Form eines Grafen verwende ich die Library *GRAL*[5]. Die Library übernimmt die grafische Darstellung von Grafen.

Softwaredesign

Die Software wird in etwa in die Packete `db`, `gui` und `controller` unterteilt sein. Eventuell ist es sinnvoll weitere Packete einzuführen.

[1] http://de.wikipedia.org/wiki/Transferobjekt
[2] vgl. http://de.wikipedia.org/wiki/Lazy_Loading
[3] http://www.sql2o.org
[4] http://de.wikipedia.org/wiki/Java_Database_Connectivity
[5] http://trac.erichseifert.de/gral

Tests

Es werden Unittests implementiert. Hierfür wird die Library *JUnit* verwendet.

Zusammenfassung

1. Datenbank
    1. MySQL
    2. Modulare Implementierung (MySQL durch z.B. SQLite austauschbar)
    3. *DAO / DTO*
    4. *Eager Loading*
    5. Library: *sql2o*
2. Grafische Darstellung
    1. Library: *GRAL* zum Zeichnen von Grafen
3. Softwaredesign
    1. Packete: `db`, `gui` und `controller`
4. Tests
    1. Library: *JUnit*

Implementierung

Umgebung

Das Programm wurde unter *OpenJDK 1.7* auf *Linux Mint 17 Cinnamon 64-Bit* getestet.

Änderungen

Während der Implementierung haben sich einige Änderungen ergeben. Im Folgenden findet sich eine grobe Auflistung jener.

1. Es wurde ein zusätzliches Packet `db.dto` eingführt, das *Plain Old Java Objects*[6] enthält, die als *Data Transfer Objects* (DTO) zum Einsatz kommen.

[6] http://de.wikipedia.org/wiki/Plain_Old_Java_Object