Schlussbewertung

Version 1.0
05.07.05

für das
Projekt

„DB-Goody“

im
Fach
Software-Projekt II
SS 2005

bei
Prof. Knabe

von
Ronald Scuda (s717000),
Thomas Koch (s718815),
Philipp Skujin (s717328),
Matthias Schönau (s716865) und
Torsten Schrape (s710219)

Revisionen

Version:	Datum:	Erläuterungen:
0.1	21.06.05	Ronald: Erstellung
0.2	27.06.05	Ronald: Änderung
0.3	05.07.05	Ronald: Änderung

Inhaltsverzeichnis

1.Einleitung

In der täglichen Arbeit der Administratoren tritt häufig das Problem auf, dass Formulare zur Dateneingabe (DML) für Datenbanken erstellt werden müssen. Diese Formulare dienen dazu, korrekte Datensätze in die Datenbank einzutragen. Dabei wird auf korrekte Eingaben sowie auf Abhängigkeiten zwischen den Tabellen geprüft. Man will vermeiden, dass die Administratoren die Dateneingaben per Hand als SQL- Befehl eingeben müssen oder die Dateneingabeformulare selber programmieren. Bei der Eingabe können Fehler auftreten, welche mit einem Formular zur Dateneingabe vermieden werden können.

Die Anwendung „DB-Goody" erstellt Dateneingabeformularen an Hand einer Strukturbeschreibung. Die Strukturbeschreibung beinhaltet die Tabellen und deren Spalten, die individuellen Beziehungen zwischen den einzelnen Tabellen und Spalten, sowie die Wertebereiche der Datenfelder. Die eingegebenen Strukturdaten werden in einer XML- Datei gespeichert. Basierend auf den Strukturdaten kann der Anwender sich die Dateneingabeformulare als dynamische Webseiten im JSP- Format generieren lassen.

Die Anwendung hat folgende Funktionen:

- Importieren und Ergänzen der Struktur einer bestehenden Datenbank
- Anlegen von Formularen auf Basis einer Datenbankstruktur
- Generierung von dynamischen Dateneingabeformularen

Achtung!

Die Anwendung ist nicht für das Datenbankenmanagement (DDL) gedacht, sondern beschränkt sich auf die Generierung von Formularen zur Datenmanipulation. Es wird keine Garantie auf die 100%ige Funktion der generierten Formulare gegeben. Die generierten Formulare müssen meisten noch individuell angepasst werden.

2.Zuständigkeiten

In der folgenden Tabelle sind die Zuständigkeiten der einzelnen Teammitglieder dargestellt. Die Angaben wurden so genau wie möglich durch Angabe des Paketes, manchmal auch der Klasse, gemacht.

Zuständigkeiten	M. Schönau	T. Koch	P. Skujin	T. Schrape	R. Scuda
GUI - Design			X		
GUI – ui			X		
GUI – helper			X		
GUI – helper - action			X		
GUI – helper - listener			X		
GUI – helper - model			X		
BL – Datenbank		X			
BL – App	X				
BL – Generieren		X		X	
BL – Generieren - GenFactory	X				
• BL – Generieren - GenManager				X	
• BL – Generieren - GenJavaScript				X	
• BL – Generieren - JSPTags				X	
• BL – Generieren - GenHtmltags				X	
• BL – Generieren - Gentags				X	
• BL – Generieren - GenJSPSQLAction				X	
BL – Generieren - GenFormTags				X	

Zuständigkeiten	M. Schönau	T. Koch	P. Skujin	T. Schrape	R. Scuda
BL – Generieren - GenSQL		X			
BL – Generieren - GenMenu		X			
BL – Generieren - GenJSPSQLResult		X			
BL – Generieren - GenDisplayTag		X			
BL – err	X				X
• BL – err – app					X
• BL – err - db	X				
• BL – err - gen		X			
• BL – err - ui	X				
• BL – err - vo					X
• BL – err - xml					X
BL – Vo	X	X			X
• BL – Vo - Anwendung	X				
• BL – Vo - Datenbank		X			
• BL – Vo - Abhängigkeiten					X
• BL – Vo - Wertebereich	X				
Persistenz – XML	X				X
• Persistenz – XML - Dateistruktur					X
• Persistenz – XML – Reader / Writer	X				
Persistenz – Datenbank		X			
Persistenz – Properties	X				

Zuständigkeiten	M. Schönau	T. Koch	P. Skujin	T. Schrape	R. Scuda
Webstart			X		
Visitor	X				
Enumeration	X				
Tests	X	X			X
• Tests - err					X
• Tests - app	X				
• Tests – db		X			
• Tests - gen					X
• Tests - vo		X			
• Tests - xml	X				
• Tests - general	X				
Dokumentation		X			X
• Dokumentation – Diagramme		X			
• Dokumentation – Dokumente					X

Genauere Angaben, z.B. wer welchen Geschäftsfall, Paket oder Klasse implementiert hat, können wir nicht machen. Jeder hat überall mal eine Änderung vorgenommen und seinen Teil beigetragen. Die Zuständigkeiten der eingesetztzen Techniken orientieren sich an den Aufgaben. Für die GUI wurden keine Tests vorgenommen.

3.Verwendete Technologien

Im folgenden werden alle verwendeten Technologien aufgezählt. Erfahrungen, welche mit ihnen gemacht wurden, sind in Form einer Positives/Negatives- Bewertung angegeben.

Opensource

- Das Projekt „DB-Goody" ist selbst Opensource nach der GNU Public Lizensbestimmungen

Positives:

- Software ist grundsätzlich „frei" verfügbar
- Achtung, unterliegt ebenso Lizenzenbestimmungen

Negatives:

- Dokumentation ist u.U. knapp oder unfrei
- Support ist meist frei (mailinglists), doch damit nicht garantiert
- Ansprüche z.B. auf fehlerfreie Funktion gibt es hier nicht

Generatoren

Positives:

- Handarbeit wird abgenommen
- z.B. für Dokumentation (javadoc, Together für Klassendiagramme)

Negatives:

- manchmal Nacharbeiten per Hand nötig (vor allen Dingen bei Diagrammen)

mySQL und SQL

MySQL ist keine professionelle Datenbank. Sie ist für den Endanwender gedacht.

Positives:

- kostenfrei
- einfache Datenbank
- per Web oder Konsole zu administrieren
- läuft unter Windows auch sehr gut

Negatives:

- es gibt keine Abhängigkeiten, müssen in Logik abgebildet werden
- Prozeduren fehlen
- es können nur SQL- Befehle abgesetzt werden, diese müssen erst zusammengebaut werden
- es gibt keine speziellen Datenbankverbindungen, z.B. für read-only (select)

I18N

Fehlertexte und einiger Konfigurationsparameter der Anwendung „DB-Goody" sind in properties- Dateien (Ressourcebundles) ausgelagert.

Positives:

- einfaches Ändern der Sprache

Negatives:

- nicht zu viel in properties- Dateien auslagern (übertreiben)
- genauer Test, dass es so fuktioniert notwendig, es lief nicht immer sofort

Swing

Die GUI wurde komplett in Swing entwickelt.

Positives:

- ist bei Java schon dabei

Negatives:

- kein elegantes Aussehen --> Verwendung von Look-and-Feel- Managern

Ant

Ant wurde zum Deployen und Kompilieren der Anwendung „DB-Goody" verwendet.

Positives:

- einfaches Deployen mittels Skript
- man vergisst nichts

Negatives:

Junit

Alle Tests wurden mit Junit erstellt.

Positives:

- auch grafisches Version der Tests möglich
- deckt schnell Fehler auf

Negatives:

- man muss sich disziplinieren, erst den Test zu schreiben
- bei Änderungen müssen Test immer nachgebessert werden
- keine Tests der GUI möglich

Java 1.5

Es wurden neue Funktionalitäten der Java 1.5- Version, z.B. Generics, Reflektion, Enumeration, verwendet.

Positives:

- String- Literale wurden durch Reflektion beseitigt
- Generics vereinfachen Programmierung --> weniger Fehler zur Laufzeit
- durch Enumeration entfallen viele switch- Anweisungen, das Ganze ist sicherer

Negatives:

- Eclipse warnt bei der Verwendung von generischen Variablen, dass man den Typ festlegen soll, dies ist aber nicht immer möglich

Tag- Bibliotheken

Bei der Generierung wurden Tag- Bibliotheken für die Erstellung der SQL- Befehle und jsp-

Seiten verwendet.

Positives:

- einfache Verwendung
- bei den jsp- Seiten kommen valide Seiten heraus

Negatives:

XML

Die Strukturdaten werden als XML- Datei gespeichert.

Positives:

- valide und well-formed Dateien
- feste Struktur durch Schema
- vorhanden Parser in Java
- Visitor- Pattern bietet sich an

Negatives:

- kleine Änderungen schon in der Schreibweise von Attributen, z.B. Gross-Kleinschreibung, haben grosse Auswirkungen

Multex

Multex wurde zu Fehlerbehandlung eingesetzt.

Positives:

- verstecken von Fehlerdetails --> Anwender wird nicht überfordert
- Ausnahmen werden mit verständlichen Texten versehen
- Unterscheidung in Benutzerfehler und Systemfehler

Negatives:

eingesetzte Pattern

Im folgenden werden die verwendeten Pattern mit Bewertung aufgezählt.

Visitor- Pattern

Das Visitor- Pattern wird bei der XML- Verarbeitung verwendet.

Positives:

- vereinfachen der Änderungen, da Methoden zentral

Negatives:

Singleton- Pattern

Einige Objekte der Anwendung "DB-Goody" sind singleton, weil sie nur einmal vorhanden sein sollen.

Positives:

- Sicherstellung, dass von einem Objekt nur eines wirklich existiert
- Zugriff über getInstance()- Methoden

Negatives:

- bei Test kann man nicht einfach Objektvergleiche machen, weil es durch Singleton immer dasselbe ist

MVC- Pattern

Positives:

- Trennung von GUI Datenmodell
- Daten als Vo- Objekte
- Kommunikation über feste Schnittstellen

Negatives:

verwendete Software

Hier wird die verwendete Software mit einigen Bemerkungen beschrieben.

Eclipse

Eclipse ist eine sehr gute IDE. Leider sind einige Plug-Ins noch nicht ausgereift und es kommt zu Problemen. Dafür ist es Opensource.

Together

Together ist ein sehr mächtiges Programm für die Analysphase. Es ist etwas überdimensioniert für unser Projekt. Leider unterstützt es noch keine Generics, sodass die Diagramme nachbearbeitet werden mussten.

SVN

SVN ist der Nachfolger von CVS. Das mergen verschiedener Versionsstände funktioniert sehr gut. Im Team ist es trotzdem aufgetreten, dass Version wieder überschrieben wurden. Die Ursache konnte nicht ermittelt werden. Menschliches Versagen schliessen wir nicht aus.

Tomcat

Der Tomcat wurde nur als Webserver zum Starten der Anwendung "DB-Goody" oder der davon generierten jsp- Dateien verwendet.

MySQL

Siehe verwendete Techniken

Entwicklungsstand

Es wurden alle muss- und soll- Kriterien umgesetzt. Viele kann- Kriterien sind nich nicht oder nur teilweise implementiert. Die Einschätzung der Kapazitäten zur Fertigung zu Beginn des Semesters wurden überschätzt.

Erfahrungen

Im großen und ganzen funktionierte die Teamarbeit ganz gut. Die Kommunikation war bei uns selten ein Problem. Da alle Beteiligen wenig Erfahrungen mit den eingesetzten Techniken hatten, ging die Implementierung langsam voran.