

Datenbank-Praktikum

Wintersemester 2002/03

3. Teilaufgabe

In dieser abschließenden Teilaufgabe des Datenbank-Praktikums dient die in Teil 2 geladenen Datenbank als Basis für eine grafische Applikation (Frontend). Die Applikation ist datenbankunabhängig implementiert und greift über eine definierte Schnittstelle auf die Daten zu. Ihre Aufgabe ist die Implementierung dieser Schnittstelle, um der Applikation den Zugriff auf die Datenbank zu ermöglichen. Ziel ist es, Ihnen die Technik von 3-Ebenen-Anwendungen (Frontend, Mittelschicht, Backend/Datenbank) näher zu bringen, weitere Möglichkeiten des datenbankunabhängigen Zugriffs mittels JDBC kennenzulernen sowie erweiterte SQL-Konzepte anzuwenden.

Die Applikation

Die bereitgestellte Anwendung bietet 3 Sichten zum Zugriff auf die Daten. Die Abfrage-sicht erlaubt die Ausführung beliebiger SQL-Anfragen mit grafischer Darstellung der Ergebnisse. Die Facts-Sicht listet relevante Daten zu einem ausgewählten Land auf und in der Country-Navigator-Sicht kann der Benutzer ausgewählte statistische Attribute zu ausgewählten Ländern sich ausgeben lassen.

Die Anwendung selbst hat keinen direkten Zugriff auf die Datenbank, sondern lädt zur Laufzeit ein datenbankspezifisches Modul, welches eine vorgegebene Schnittstelle (Interface) implementiert und als Mittelschicht den Zugriff auf die Datenbank realisiert. Dieses Modul soll von Ihnen implementiert werden.

Die Applikation wird durch `java wfb.frontend.Wfb` gestartet und erwartet den Namen einer Property-Datei als einzigen Parameter (siehe API-Dokumentation zu `java.util.Properties` für eine Syntaxbeschreibung von Property-Dateien). Diese Datei muss wenigstens die Einträge `modul`, `flagge` und `karte` enthalten. `Modul` enthält den vollständigen Klassennamen des von Ihnen erstellten Moduls (siehe Kapitel *Das Modul*). `Flagge` und

`Karte` enthalten jeweils die Pfade zu den Karten- und Flaggenbildern (ohne Verzeichnis `figures`). Alle weiteren Parameter, die Sie für Ihr Modul benötigen, können Sie in diese Datei eintragen (JDBC-Treiber-Klassenname, Logininformationen, ...).

Das Modul (Mittelschicht)

Das von Ihnen zu implementierende Modul stellt die Verbindung zwischen Anwendung und Datenbank her. Dazu muss es die Schnittstelle `wfb.definitions.WfdbInterface` implementieren.

Methoden der Schnittstelle

Die API-Beschreibung der Methoden der Schnittstelle und der anderen Klassen finden Sie unter `jdbc/index.html`. Im Nachfolgenden werden die einzelnen Schnittstellen-Methoden genauer erläutert.

init

Hier sollte die Datenbankverbindung für die anderen Methodenaufrufe erstellt werden, sowie weitere Aktionen, die zur Initialisierung notwendig sind, ausgeführt werden. Alle notwendigen Parameter sollen aus dem übergebenen Property-Objekt entnommen werden.

finish

Damit die Mittelschicht alle Ressourcen kontrolliert wieder freigeben kann, wird diese Methode bei Beendigung der Anwendung aufgerufen. Hier sollten speziell die Datenbankobjekte wieder freigegeben werden.

getCountries

Diese Methode soll eine Liste der in der Datenbank enthaltenen Ländernamen, die mit dem übergebenen Pattern übereinstimmen, zurückliefern. Beachten Sie, dass im Falle von `pattern=null` die komplette Liste zurückgeliefert wird. Das Pattern kann SQL-Wildcards enthalten.

Hinweis: der Patternvergleich kann mittels des SQL-Operators `like` durchgeführt werden.

getCountryData

Als Ländername wird ein Wert aus `getCountries` übergeben. Für dieses Land soll ein `CountryData`-Objekt zurückgeliefert werden (Klasse `wfb.definitions.CountryData`), das mit den Daten des angegebenen Landes initialisiert ist.

- 1 -

- 2 -

getCities

Für das übergebene Land sollen alle Städte und ihre Attribute, jeweils gekapselt in `CityData`-Objekte (Klasse `wfb.definitions.CityData`), in einer Liste zurückgeliefert werden.

getOrganizations

Gibt eine Liste aller Abkürzungen der in der Datenbank enthaltenen Organisationen zurück.

getStatisticAttributes

Hier wird davon ausgegangen, dass Sie für die zeitabhängigen Attribute eine eigene Tabelle angelegt haben. Zurückgegeben werden sollen alle Attribute, die in dieser Tabelle enthalten sind (außer dem Jahr und Landverweis). Verwenden Sie zum Ermitteln der Attribute die Metadaten-Klassen der JDBC-API. Da diese Attributnamen zur Anzeige kommen, sollen nicht die Attributnamen wie sie im Schema angelegt sind zurückgegeben werden, sondern ausführliche/sprechende Namen. Für die Umsetzung der Schemanamen nach Ausgabenamen verwenden Sie bitte eine extra Tabelle, so dass auch weitere Attribute hinzugefügt werden können.

getStatisticYears

Gibt eine Liste aller Jahre zurück, für die zeitabhängige Daten in der Datenbank vorliegen.

getStatisticData

Liefert zeitabhängige Daten für die übergebenen Länder, die den Kriterien des (Nicht)Enthaltens in angegebenen Organisationen genügen. Es sollen nur die angegebenen Jahre und Attribute enthalten sein. Die übergebenen Attributnamen sind die ausführlichen Namen (siehe Methode `getStatisticAttributes`). Beachten Sie, dass die übergebenen Parameter auch null enthalten können (siehe API-Dokumentation)!

Das zurückzuliefernde `StatData`-Objekt wird mit einem Array der Attributnamen initialisiert und für jedes Land und Jahr die Daten mittels `addData` übergeben. Beachten Sie, dass die Reihenfolge der Attributnamen und der Daten übereinstimmen muss.

updateCountryComment

Erweitern Sie nachträglich Ihre Ländertabelle um ein Attribut, welches einen Kommentar zum Land enthält (`ALTER TABLE ...`). Der Inhalt des neuen Attributs wird durch diese Methode für das übergebene Land gesetzt. Die Kommentar-Änderungen sollen in einer Log-Tabelle protokolliert werden. Der Logeintrag soll nicht per Trigger sondern in dieser Methode erfolgen. Sichern Sie durch Verwendung des Transaktionsmechanismus die Atomarität der gesamten Aktion.

- 3 -

executeQuery

Die übergebene Query soll auf der Datenbank ausgeführt und das Ergebnis in einem Objekt zurückgeliefert werden, welches die Schnittstelle `wfb.definitions.SQLResultInterface` implementiert (siehe die API-Dokumentation). Die Namen der zurückgelieferten Attribute erhalten Sie wieder über die Metadaten des Result-Objektes (JDBC-API).

Implementierungsrichtlinien

- Es sollen keine Parameter im Quellcode fest vorgegeben werden. Stattdessen sind alle Parameter aus dem `Property`-Objekt auszulesen, welches beim Aufruf der Methode `init()` übergeben wird (siehe auch Hinweise zur Property-Datei im vorangegangenen Kapitel). Hierzu gehören auch der Klassenname des JDBC-Treibers, die JDBC-URL, Logininformationen usw. Dieses ist eine wesentliche Voraussetzung um ein datenbankunabhängiges Programm zu erstellen. Ein Datenbankwechsel kann dann im günstigsten Fall einfach durch Ändern der Parameter in der `Property`-Datei vollzogen werden.
- Die einzelnen Methoden verwenden meistens die selben Anfragen mit unterschiedlichen Parametern. Verwenden Sie dafür jeweils `PreparedStatement`-Objekte, die sie z.B. bei Aufruf von `init()` erzeugen und in den einzelnen Methoden vor dem Ausführen nur parametrisieren müssen.
- Beachten Sie, dass bedingt durch die grafische Oberfläche auch mehrere Methoden parallel aufgerufen werden können. Dieses wäre auch der Fall, wenn ihr Modul von einem Webserver zum Aufbau dynamischer HTML-Seiten verwendet werden würde. Stellen Sie deshalb durch geeignete Maßnahmen sicher, dass parallele Aufrufe verschiedener bzw. auch derselben Methode problemlos möglich sind.
Hinweis: Synchronisation
- Bei jedem Methodenaufruf soll auf die Daten der Datenbank zurückgegriffen werden (kein Vorhalten der Daten durch die Mittelschicht).
- Alle Methoden greifen ausschließlich auf Views zu (siehe unten)

Sicherheitsrichtlinie

Um einen sicheren Einsatz des Frontends zu ermöglichen, darf dieses auf die Datenbank nicht mit den Rechten der Praktikumssteilnehmer zugreifen. Stattdessen sollen alle benötigten Daten (und nur diese) in `Views` gekapselt werden, auf die der Nutzer `dbread` mit Passwort `.dbread` lesen, bzw. im Falle des Land-Kommentar-Attributs, auch schreibenden Zugriff erhält. Sie müssen also Ihr Schema um entsprechende `Views` erweitern und dem angegebenen Nutzer die notwendigen Rechte erteilen (`GRANT...`).

- 4 -

Testimplementierung

Damit Sie einen ersten Eindruck von der Funktionalität des Frontends bekommen, haben wir ein Dummy-Modul implementiert, welches ohne auf die Datenbank zuzugreifen Testdaten für die Methodenaufrufe generiert. Dieses Modul hat den Klassennamen `wfb.wfdbimpl.DummyImpl`. Eine Property-Datei für das Frontend, welche das Dummy-Modul referenziert, ist Ihnen als `dummy.properties` gegeben.

Abtestat

Zum Fertigstellungstermin (s.u.) soll das Modul **voll funktionsfähig** gemäß Aufgabenstellung nutzbar sein. Zum Abtestat ist die Applikation mit Ihrem Modul vorzustellen.

Termine:

<i>Beginn der Bearbeitung</i>	ab sofort
<i>Fertigstellung</i>	bis Freitag, 07.02.03 (Ende des Praktikums)
<i>Durchführung der Testate:</i>	n.V. (Datum/Uhrzeit pro Gruppe nach Absprache mit jew. Betreuer), jedoch spätestens zum Semesterende

Mit Semesterende verfallen alle nicht abgeschlossenen Praktika.

Hinweise

- alle Quelldateien des Frontends, der Schnittstelle und des Dummy-Moduls befinden sich unter `3.aufg/src` (Frontend im Paket `wfb.frontend`, Schnittstelle im Paket `wfb.definitions`, Dummy-Modul im Paket `wfb.wfdbimpl`); die Quelldateien können Sie in Ihr home-Verzeichnis kopieren. Beachten Sie, dass die Paketstruktur nicht verändert werden darf. Zum Übersetzen der Klassen müssen Sie sich im Verzeichnis `src` befinden (z.B. `javac wfb.frontend/Wfb.java` und Aufruf mittels `java wfb.frontend.Wfb PROPERTY_DATEI`)
- die API-Dokumentation der Schnittstellenklassen kann im Browser angezeigt werden (Pfad `3.aufg/doc/index.html`)
- Die jpeg-Bilder von Flaggen und Karten befinden sich unter:
`/home/dbprak00/prak/figures`
- **Bitte vermeiden Sie, die Bilder in Ihr Verzeichnis zu kopieren**
- Beachten Sie bitte die **Literaturhinweise auf der Praktikumsseite im WWW**. Sie

sollten sich unbedingt mit der API-Spezifikation (<http://dbs.uni-leipzig.de/de/praktika/docs/api/index.html>) vertraut machen.

- Metadaten zum Datenbankschema erhalten Sie über `Connection.getMetaData()`
- Metadaten über das aktuelle `ResultSet` erhalten Sie mittels `ResultSet.getMetaData()`
- Im Normalfall befinden sich JDBC-Verbindungen im `AutoCommit`-Modus, d.h. jede einzelne Datenbankoperation wird als einzelne Transaktion aufgefaßt und automatisch mit einem Commit versehen. Sollen mehrere Operationen in einer **Transaktion** zusammengefaßt werden, muß die Verbindung mittels `Connection.setAutoCommit(false)` in den 'manuellen' Commit-Modus versetzt werden. Die erste Operation beginnt dann die Transaktion und das Transaktionsende muß vom Programm mittels `Connection.commit()` bzw. `Connection.rollback()` markiert werden. Beachten Sie, dass die Transaktionsverwaltung auf Verbindungsebene abläuft. Dieses muss im Falle paralleler Anfragen berücksichtigt werden.