



Softwarepraktikum im Grundstudium im Wintersemester 2002/2003

3. Übung, 7. November 2002

In dieser Übung wird die persistente Speicherung der Aufgabendaten in einer relationalen Datenbank implementiert. Zusätzlich soll die Anwendung um eine Fehlerbehandlung erweitert werden. Alle Gruppen erhalten Zugang zu einer Datenbank, die eine Tabelle `task` mit folgendem Aufbau enthält:

<code>key</code>	<code>int4, serial primary key, wird automatisch gesetzt (getLong)</code>
<code>shorttitle</code>	<code>varchar (getString)</code>
<code>description</code>	<code>varchar (getString)</code>
<code>startdate</code>	<code>date (getDate)</code>
<code>enddate</code>	<code>date (getDate)</code>

Zur Information: Angelegt wurde die Tabelle mit folgendem SQL-Statement:

```
CREATE TABLE tasks (key SERIAL PRIMARY KEY, shorttitle
VARCHAR(50), description VARCHAR(8192), startdate DATE,
enddate DATE);
```

1. Aufgabe (Klasse `SAException`)

Leiten Sie eine Klasse `SAException` von `Exception` ab. Sie implementiert lediglich zwei Konstruktoren, die jeweils über `super` die entsprechenden Konstruktoren von `Exception` aufrufen:

- `public SAException(String message)`: Setzt die Meldung der `Exception` auf `message`
- `public SAException(String message, Throwable t)`: trägt zusätzlich das Fehlerobjekt `t` als Grund für den Fehler ein.

2. Aufgabe (Klasse `StorageAccess`)

Die Klasse `StorageAccess` verkapselt das persistente Abspeichern von Daten der Aufgabenverwaltung. Alle Details der Datenspeicherung werden in dieser Klasse verborgen, damit später leicht andere Realisierungen eingesetzt werden können. In dieser Übung soll das Abspeichern in der relationalen Datenbank PostgreSQL erfolgen. Hierzu enthält die Klasse `StorageAccess` eine Instanzvariable vom Typ `Connection`, die eine Datenbankverbindung verwaltet. Implementieren Sie folgende Methoden der Klasse `StorageAccess`:

- `public void connect()`: Diese Methode stellt die Verbindung zur Datenbank her und speichert diese in der entsprechenden Instanzvariablen ab. Hinweise hierzu finden Sie in den Crashkurs-Folien. Der Connect-String ist wie folgt aufgebaut: `jdbc:postgresql://salieri.informatik.rwth-aachen.de //spws02_gruppennr`. Ihre Gruppennummer sowie Benutzername und Passwort erhalten Sie per Email.
- `public void disconnect()`: trennt die Verbindung zur Datenbank

- `public Tasks getAllTasks():` Liefert alle Aufgaben aus der Datenbank als Objekt der Klasse `Tasks` zurück. Hierzu muss eine Anfrage an die Datenbank gestellt werden und alle Einträge des `ResultSet`s müssen in ein Objekt vom Typ `Tasks` eingefügt werden.
- `public void createTask(Task task):` Fügt das übergebene Aufgabenobjekt mit Hilfe eines `SQL-INSERT`-Statements in die Datenbank ein. Definieren Sie zunächst mit Hilfe der Funktion `prepareStatement` der `Connection` ein `PreparedStatement` mit Parametern. Setzen Sie dann mittels der `get-` und `set-`Funktionen des `Statements` deren Werte und führen die Abfrage mit `executeUpdate` aus.

An allen Stellen, an denen `Exceptions` auftreten können, sollen diese abgefangen werden. Im jeweiligen `catch`-Block soll eine `SQLException` erzeugt und geworfen werden. Dabei ist als Meldung eine Beschreibung der fehlgeschlagenen Operation und als Grund die aufgefangene `Exception` zu übergeben.

3. Aufgabe (Test)

Schreiben Sie ein Kommandozeilenprogramm, das eine Instanz der Klasse `StorageAccess` erzeugt, die Verbindung zur Datenbank herstellt, zwei Aufgabenobjekte füllt und mithilfe der Klasse `StorageAccess` abspeichert. Dann sollen alle Aufgaben aus der Datenbank angezeigt werden.

4. Aufgabe (Abstützen der Webapplikation auf die Datenbank)

Ändern Sie die Webapplikation so ab, dass die Aufgaben über die Klasse `StorageAccess` in der Datenbank gespeichert werden. Hierzu muss in den Klassen `ShowTasksPrepare` und `NewTaskExecute` die Klasse `StorageAccess` zum Lesen und Abspeichern der Aufgaben eingesetzt werden. Dafür soll in den jeweiligen `doGet` bzw. `doPost` Methoden wie in Aufgabe 3 verfahren werden. `ShowTasksPrepare` soll dann wie bisher einen `Iterator` an `ShowTasksView` übergeben. Die Objekte der Klasse `StorageAccess` sollen nur in lokalen Variablen gespeichert werden. Die Initialisierung in der `init()`-Methode ist nicht mehr notwendig.

5. Aufgabe (Fehlerbehandlung)

Erstellen Sie eine Java Server Page `error.jsp`, die ein im `request` als Attribut übergebenes Objekt vom Typ `throwable` (Oberklasse von `Exception`) ausgibt. Neben der Fehlermeldung kann ein `throwable`-Objekt ein weiteres `throwable`-Objekt enthalten. Dieses gibt den Grund des Fehlers an. Auf dieses kann mittels `getCause()` zugegriffen werden. Auch dieses `throwable`-Objekt kann wieder ein weiteres enthalten usw. Ist keins mehr enthalten, gibt die Funktion `getCause()` null zurück. Geben Sie in einer Schleife die Meldungen des übergebenen Objektes und die aller untergeordneten Objekte aus.

Fangen Sie in allen bisher erstellten `Servlets` alle auftretenden `Exceptions` ab und benutzen Sie im Fehlerfall einen `RequestDispatcher`, um auf die Seite `error.jsp` weiterzuleiten. Übergeben Sie die abgefangene `Exception` als Attribut im `Request`. Verfahren Sie so auch bei allen zukünftig erstellten `Servlets`.

Testen Sie Ihre Fehlerbehandlung anhand der Eingabe eines ungültigen Datums beim Anlegen einer neuen Aufgabe.

Testat

Aufgaben 1-3: Donnerstag, 14.11.2002 von 10.00 - 12.00

Aufgaben 4-5: Donnerstag, 21.11.2002 von 10.00 - 12.00

Nächste Besprechung: Donnerstag, 21.11.2002, 14.15 Uhr