Vertiefendes Übungsprojekt - SQL II

Albert Weichselbraun

Ablauf

- Ablauf der Lehrveranstaltung
- Vorstellung des Projektthemas
- Projektgruppen
- Vorstellung der Arbeitsumgebung (Software, Locations)
- Walkthrough
  - Datenbankentwurf
  - Formulare
  - PHP
- Security

Ablauf der Lehrveranstaltung

Software Projekt

- Projektphasen entsprechend SA/SD Modell
- Präsentation / Diskussion
  - Problemdefintion/Use Cases
  - ER-Modell/DB-Design
- Implementierung
- Abnahme und Präsentation
  - Benutzerdokumentation
  - Work Reports (wer hat was gemacht)

Wichtige Termine

- 11. Mai 2011: Vorbesprechung, Gruppeneinteilung und Projektvergabe
- 18. Mai 2011: Präsentationen/Besprechung: Problemdefinition und Use Case Diagramme; ER-Modelle, DB-Design
- 25. Mai 2011: Interface/Implementierung
- 1. Juni 2011: Interface/Implementierung
- 8. Juni 2011: Interface/Implementierung
- 15. Juni 2011: Abschlusspräsentation und Projektübergabe

Beurteilungsschema

- 20 % Mitarbeit, Präsentation
- 80 % Projekt (Usability, Datenbankdesign/Performance, Security)
  - 30 % Projektdokumentation
  - 40 % Ausführung
  - 10 % Endpräsentation

Vorstellung der Arbeitsumgebung

- Software im Schulungsraum (dia, kate, psql ...)
- Verlinken des Arbeitsverzeichnisses:

  `ln -s ~aweichse/public_html/projects/2011s/sql2/{gruppe} sql2`

- Struktur:
  - Application : *./index.php*
  - Administration: *./admin*
  - Dokumentation : *./projekt*
- Zugriff:
  - über das Web:
    http://xmdimrill.ai.wu.ac.at/weichselbraun/projects/2011s/sql2/{gruppe}
  - von Extern: ssh j0625050@xmdimrill.ai.wu.ac.at

Walkthrought

- Problemdefintion
- Durchführbarkeitsstudie
  → UML Use-Case Diagramme + Bewertung
  → muss klar werden *was* zu tun ist!
- Analyse (ER-Modell)
- Design (Datenbank, Interface), Business Logic
- Implementierung:
  - Datenbank + Constrains erstellen
  - Formulare (vorzugewiese HTML + CSS, eventuell Swing)
- Dokumentation (Benutzer, Projekt, Installation, Work Reports)

Datenbankentwurf

- Ausgangspunkt: ER-Modell
- Alle Tabellen sind zu normalisieren
- Indices für abfragerelevante Felder (!= Datum, ...)
  - `UNIQUE` in Tabellendefition
  - `CREATE INDEX`
- Business Logic
  - Constraints
    - `NOT NULL`, `CHECK`, Fremd-/Primärschlüssel
    - `ON DELETE`, `ON UPDATE`
  - Datentypen

Benutzerinterface

- Text Encoding: UTF8

```
<meta http-equiv="Content-Type"
      content="text/html; charset=utf8" />
```

- Layout: Cascade Style Sheets (CSS)
- HTML-Formulare <FORM>, <INPUT>, <SELECT>
- Übergabe der Variablen an/von PHP (POST/GET)

HTML-Head

```
<html>
  <head>
    <link rel="stylesheet" href="style.css"
        type="text/css" />
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf8" />
    <title>PHP Test</title>
  </head>

body       { background: #FFFFFF; }
a          { color: #0000FF; }
a:active   { color: #000000;}
h1         { font-family: helvetica; color: blue;}
input.url  { background: lightgrey; }
```

HTML-Formulare

```
  <body>
    <h1>TestFormular</h1>

    <form>
      Name: <input name="name" />
      Url : <input name="url" />  <br />
      <input type="submit" value="Abschicken"
          name="fertig" />
    </form>

  </body>

</html>
```

Übergabe von Variablen

```
<html>
  <head>
    <link rel="stylesheet" href="style.css"
        type="text/css" />
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf8" />
    <title>Summenberechnung</title>
  </head>

<body>
  <h3>Summe:</h3>
  <form action="<?=$_SERVER['PHP_SELF']?>">
    <input name="a1" value="<?=$_REQUEST['a1']?>" />
    <input name="a2" value="<?=$_REQUEST['a2']?>" />
    <br />
    <input type="submit" value="Summe" />
  </form>
```

Übergabe von Variablen

```
19    <h3>Ergebnis</h3>
20    <?php
21    $a1 = $_REQUEST['a1'];
22    $a2 = $_REQUEST['a2'];
23    if ($a1 and $a2) {
24       $sum = $a1+$a2;
25       echo "Die Summe von $a1 und $a2 ist: $sum";
26    }
27    ?>

29  </body>
30  </html>
```

Sessions

- Benutzer bekommt eine eindeutige Id.
- Variablen serverseitig(!) speicherbar

```
1  <?php
2  session_start();
3  if (!isset($_SESSION['count'])) {
4          $_SESSION['count'] = 0;
5      } else {
6          $_SESSION['count']++;
7  }
8  ?>
9  Diese Seite wurde <?=$_SESSION['count']?> mal
10 angesehen.
```

PHP + PostgreSQL

- Vorgangsweise:
  1. Datenbankverbindung mit pg_connect herstellen.
  2. Query vorbereiten und absetzen.
  3. Auf Fehler prüfen / Resultate ausgeben.
- Wiederkehrende Queries ⇒ pg_prepare + pg_execute.
- Einfache Queries ⇒ pg_query_params.
- Queries *ohne* Benutzereingaben im Querytext ⇒ pg_query.

PHP + PostgreSQL

```
1  <html>
2    <head>
3       <link rel="stylesheet" href="style.css"
4          type="text/css" />
5       <meta http-equiv="Content-Type"
6          content="text/html; charset=utf8" />
7       <title>PHP Test</title>
8    </head>

10 <body>
11   <h3>PHP Test</h3>
12   <?php
13   if ($_REQUEST['company_id' ]) {
14     $company_id= $_REQUEST['company_id'];
15   }

17   $db = pg_connect("host=xmdimrill dbname=weblyzard
18                     user=albert password=xyz");
```

PHP + PostgreSQL

```
20    $result = pg_query_params($db,
21        "SELECT company_id, name, url FROM company
22         WHERE company_id=$1 LIMIT 1",
23        array($company_id));

25    if (!$result) {
26       echo "No records present";
27       exit;
28    }

30    // Output Table Header
31    echo "<table>";
32    echo "<tr><th>Id</th><th>Company Name</th>";
33    echo "    <th>Company Url</th></tr>";
```

PHP + PostgreSQL

```
37    while ($row = pg_fetch_row($result) ) {
38       echo "<tr><td>$row[0]</td></tr>";
39       echo "<td>
40         <input name=\"name\" size=40 value=\"$row[1]\" />
41         </td></tr>";
42       echo "<td>
43         <input name=\"url\" size=40 value=\"$row[2]\" />
44         <a href=\"$row[2]\">[x]</a></td></tr>";
45    }

47    // Table footer
48    echo "</table>";

50 ?>
51 </body>
52 </html>
```

Security

- Why security matters
- Datenbank Passwort
- PHP Security Issues (Global Variables), inc-Files (- .php!)
- SQL Injection

Datenbank Passwort

- Jede Gruppe erhält einen Benutzer und Passwort für die PostgreSQL Datenbank
- Passwort Location: außerhalb des Apache-Trees

```
1    function getDb() {
2      return pg_connect("host=localhost
3          dbname=mdb user=albert password=xyz");
4    }
```

PHP Security Issues

- PHP Include Dateien
  - Endung: `.php`
  - Kritische Dateien: ausserhalb des für den Webserver lesbaren Bereiches
- Übergebene Variablen immer aus dem $_REQUEST Array auslesen.
- `register_globals` aktiviert → Manipulationen einfacher

Dangerous

`register_globals` aktiviert.

```
1  <?php
2     if ($username == 'toni' and $pwd == 'secret')
3      $authorized = true;
4  ?>

6  <?php if (!$authorized): ?>
7  <!-- Formular: Eingabe Benutzer + Passwort -->

9  <?php else: ?>
10 <!-- Geheime Informationen -->

12 <?php endif; ?>
```

Correct

`register_globals` deaktiviert.

```
1  <?php
2     $username = $_REQUEST['username'];
3     $password = $_REQUEST['pwd'];
4     if ($username == 'toni' and $pwd == 'secret')
5      $authorized = true;
6  ?>

8  <?php if (!$authorized): ?>
9  <!-- Formular: Benutzername/Passwort -->

11 <?php else: ?>
12 <!-- Geheime Informationen -->

14 <?php endif; ?>
```

SQL Injection

- Benutzereingaben in Queries
- Gegenmaßnahmen:
  - Prüfen der Benutzereingaben
  - Verwendung von `pg_query_params` (preferred)