CONSTRUCTOR UNIVERSITY

Database Application Development

Ramakrishnan & Gehrke, Chapter 6

PHP and (My)SQL

www.php.net

- PHP calls embedded within HTML as special tag
  - <?php *php-statement-sequence* ?>
- Execution (server-side!) of PHP:
- PHP statements → (HTML) text; complete file forwarded by Web server:

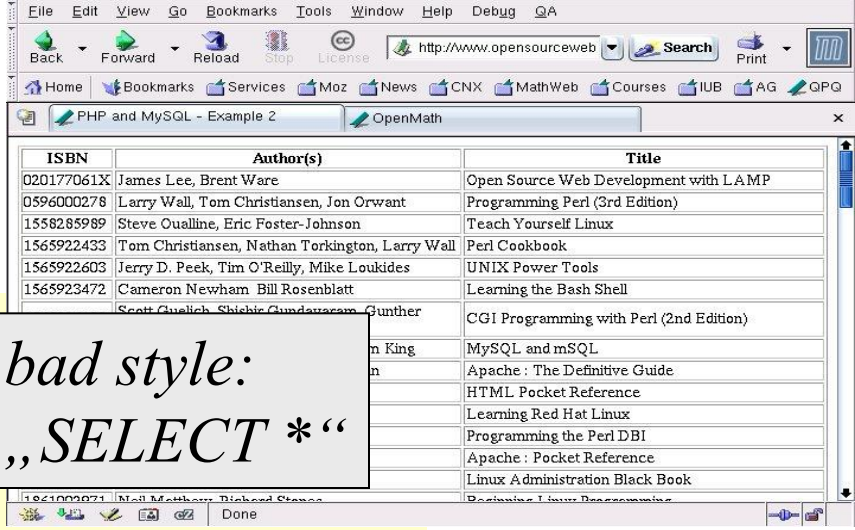
```
<h1><?php echo “Hello World”; ?></h1>   ⟶   <h1>Hello World</h1>
```

- Example: connecting to mysql server on localhost

```
<?php
   $mysql = mysql_connect( “localhost”, “apache”, “DBWAisCool” )
      or die( “cannot connect to mysql” );
?>
```

variables have „$“ prefix

PHP, HTML, and (My)SQL



*bad style: „SELECT *“*

```
<html>
  <head>
    <title>PHP and MySQL Example</title>
  </head>
  <body>
   <?php $mysql = mysqli_connect( “localhost” );
     $result = mysql_db_query( “books”, “SELECT isbn, author, title FROM book_info" )
        or die( “query failed - “ . mysql_errno() . “: “ . mysql_error(); )
    ?>
    <table>
      <tr>  <th>ISBN</th>  <th>Author(s)</th>  <th>Title</th>  </tr>
      <?php while ( $array = mysql_fetch_array($result) ); ?>
      <tr><td><?php echo $array[ "isbn"    ]; ?></td>
            <td><?php echo $array[ "author" ]; ?></td>
            <td><?php echo $array[ "title"    ]; ?></td>
      </tr>
      <?php endwhile; ?>
    </table>
    <?php mysql_close($mysql); ?>
  </body>
</html>
```

open

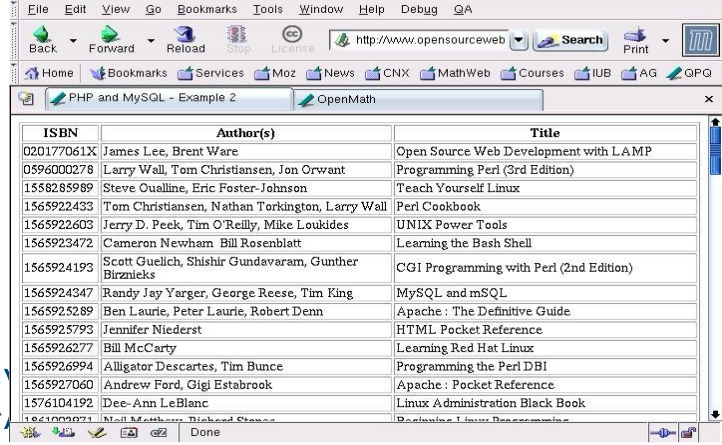
query

iterate over result set

close

Python and (My)SQL

- Different approach: Python prime, not HTML
- Python code example (schematic, using Flask)



ISBN	Author(s)	Title
020177061X	James Lee, Brent Ware	Open Source Web Development with LAMP
0596000278	Larry Wall, Tom Christiansen, Jon Orwant	Programming Perl (3rd Edition)
1558285989	Steve Oualline, Eric Foster-Johnson	Teach Yourself Linux
1565922433	Tom Christiansen, Nathan Torkington, Larry Wall	Perl Cookbook
1565922603	Jerry D. Peek, Tim O'Reilly, Mike Loukides	UNIX Power Tools
1565923472	Cameron Newham Bill Rosenblatt	Learning the Bash Shell
1565924193	Scott Guelich, Shishir Gundavaram, Gunther Birznieks	CGI Programming with Perl (2nd Edition)
1565924347	Randy Jay Yarger, George Reese, Tim King	MySQL and mSQL
1565925289	Ben Laurie, Peter Laurie, Robert Denn	Apache : The Definitive Guide
1565925793	Jennifer Niederst	HTML Pocket Reference
1565926277	Bill McCarty	Learning Red Hat Linux
1565926994	Alligator Descartes, Tim Bunce	Programming the Perl DBI
1565927060	Andrew Ford, Gigi Estabrook	Apache : Pocket Reference
1576104192	Dee-Ann LeBlanc	Linux Administration Black Book

```
from flask import Flask, request, render_template
import mysql.connector
import sys
app = Flask( __name__ )
query_components = parse_qs(urlparse(self.path).query)
name = query_components["name"][0]

def booklist() :
    connection = mysql.connector.connect( … )
    cursor = connection.cursor()
    cursor.execute( "SELECT isbn, author, title FROM book_info WHERE author='?' ", author )
    result = cursor.fetchall()
    connection.close()
    return render_template( 'booklist.html', result = result )
```

not in exams

CONSTRUCTOR UNIVERSITY

SQLJ

- SQLJ = Java + embedded JDBC database access, nicely wrapped
  - ISO standard
  - eliminates JDBC overhead
    → compact & elegant database code, less programming errors
- SQLJ program ----[SQLJ translator]----> std Java source code
  - embedded SQL statements → calls to SQLJ runtime library
- (semi-) static query model: Compiler does
  - syntax checks, strong type checks
  - consistency wrt. schema
- Primer: http://archive.devx.com/dbzone/articles/sqlj/sqlj02/sqlj012102.asp

not in exams

CONSTRUCTOR UNIVERSITY

SQLJ Code Example

```
Int sid; String name; Int rating;
#sql iterator Sailors( Int sid, String name, Int rating );
Sailors sailors;

#sql sailors =
   { SELECT sid, sname INTO :sid, :name FROM Sailors WHERE rating = :rating };

while (sailors.next())
{    System.out.println( sailors.sid + ": " + sailors.sname) );
}

sailors.close();
```

SQLJ Iterators

- Named iterator
  - Needs both variable type and name, and then allows retrieval of columns by name
  - See example on previous slide:
    #sql iterator Sailors(Int sid, String name, Int rating);
- Positional iterator
  - Needs only variable *type* (not name), uses FETCH ... INTO construct:

```
#sql iterator Sailors( Int, String, Int );
Sailors sailors;
#sql sailors = { SELECT sid, sname INTO :sid, :name FROM Sailors WHERE rating = :rating };
do
{   #sql { FETCH :sailors INTO :sid, :name };
    if ( ! sailors.endFetch() )
        …; // process sailor
} while (! sailors.endFetch() );
```

Stored Procedures

- Program executed through a single SQL statement
  - Executed in server
- Advantages:
  - Can encapsulate application logic while staying “close” to the data
  - Reuse of application logic by different users
  - Avoid tuple-at-a-time return of records through cursors

SQL/PSM Example

- PSM code:

```
CREATE FUNCTION rateSailor (IN sailorId INTEGER) RETURNS INTEGER
        DECLARE rating INTEGER
        DECLARE numRes INTEGER

        SET numRes = (SELECT COUNT(*)
                      FROM Reserves R
                      WHERE R.sid = sailorId)

        IF (numRes > 10)
        THEN rating =1;
        ELSE rating = 0;
        END IF;

        RETURN rating;
```

- Foreign code:

```
CREATE PROCEDURE TopSailors( IN num INTEGER )
   LANGUAGE JAVA
   EXTERNAL NAME "file:///c:/storedProcs/rank.jar"
```

Calling Stored Procedures from Client

- Embedded SQL:
  - EXEC CALL IncreaseRating(:sid, :rating);
- JDBC:
  - CallableStatement cstmt = con.prepareCall("{call ShowSailors}");
- SQLJ:
  - #sql showsailors = { CALL ShowSailors };

Summary: Connecting PL & DBMS

- Coupling techniques
  - API: library with DBMS calls = layer of abstraction between application and DBMS
  - Embedded SQL: extend PL with SQL statements
  - Stored procedures: execute application logic directly at the server
- Cursor for record-at-a-time traversal