INF115 Lecture 14: *Web Applications*

Adriaan Ludl
Department of Informatics
University of Bergen

Spring Semester 2021

Chapter 12: *Web Applications*

Learning Goals:

- **Technologies** that the **internet** and web are based on.
- **Structure of HTML** documents.
- Understand the **connection** between **information systems**, **database systems** and **web applications**.
- Create **simple PHP scripts** based on examples.
- Techniques for **securing** web applications.

What is PHP?

PHP stands for the *recursive initialism:*

PHP: Hypertext Preprocessor

https://en.wikipedia.org/wiki/PHP

What is PHP?

PHP is a programming language / scripting language

- Used to **create web applications**
- PHP script **runs** on web servers
- PHP scripts generate ***dynamic*** web pages
- The **content** of web pages is **often retrieved from databases**
- PHP and MySQL are often used together

What is the difference between PHP and SQL?

- PHP is a **general programming language**
- SQL is only used to write queries to a database
- A **PHP script** (program) **describes step-by-step calculations** (using variables, choices, and loops)
- **An SQL query is declarative** - describes *what* more than *how* ("high level")

Static vs Dynamic Web Pages

A **static** web page (sometimes called a flat page or a stationary page) is a web page that is delivered to the user's web browser exactly as stored,

in contrast to **dynamic** web pages which are generated by a web application.

Consequently, a **static web page** displays the same information for all users, from all contexts [subject to browser capabilities, …].

https://en.wikipedia.org/wiki/Static_web_page

What is PHP?

PHP is a programming language / scripting language

- Used to **create web applications**
- PHP script **runs** on web servers
- PHP scripts generate ***dynamic*** web pages
- The **content** of web pages is **often retrieved from databases**
- PHP and MySQL are often used together

What is the difference between PHP and SQL?

- PHP is a **general programming language**
- SQL is only used to write queries to a database
- A **PHP script** (program) **describes step-by-step calculations** (using variables, choices, and loops)
- **An SQL query is declarative** - describes *what* more than *how* ("high level")

Caveats

- This is an introduction to the basics of PHP.

- Learning PHP takes time.
- The textbook is not a reference work on PHP.

- The possibility of typing **errors** is greater in PHP than in SQL.
- PHP scripts are usually **much longer** than SQL queries.
- It is **more difficult** to find errors in PHP than in SQL.

- To learn how to program, refer to other books / courses.

An introduction to PHP

You will see examples of what PHP can be used for:

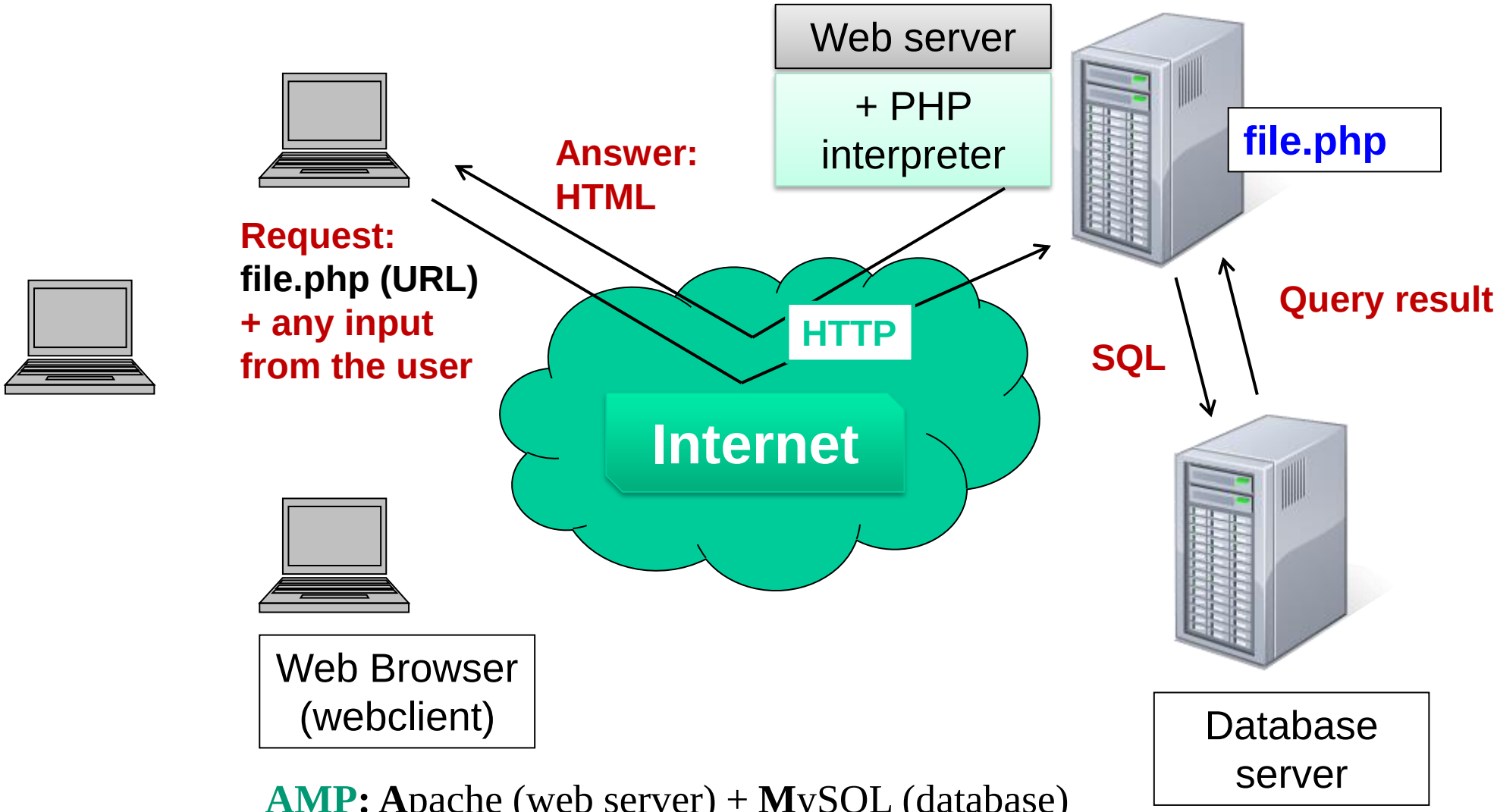
- Creating dynamic web pages
- Retrieving data from a MySQL database
- The PHP scripts send SQL to the database

You will read complete PHP code and (maybe) make some small changes .

This will give you some "more hooks to hang database fabric on" … and

- Show that databases are part of an information system
- Show how SQL is used in programming

Web Applications with PHP



AMP: Apache (web server) + **M**ySQL (database) + **P**HP (programming language)

Program 1: Today's date

```
<html>
<body>

<?php

  // Use the DATE function to retrieve the current date
  // of the form dd.mm.yyyy. Saves in the variable $date.
  $dato = DATE("d.m.Y");

  // Print the current date in HTML h1 header.
   echo "<h1>Dagens dato: $dato</h1>";
?>

</body>
</html>
```

Today's date **: 13.04.2021**

XAMPP

XAMPP is a packaged solution that provides a database + web server on your own machine.

- Normally we need to copy PHP scripts onto a web server to test / run.
- With XAMPP, saving the files to subdirectory htdocs (eg C: \ xampp \ htdocs).
- First: Download and install XAMPP + launch the XAMPP console and from here launch the Apache web server.

Save the code from the previous slide to file:

- C: \ xampp \ htdocs \ dagsdato.php

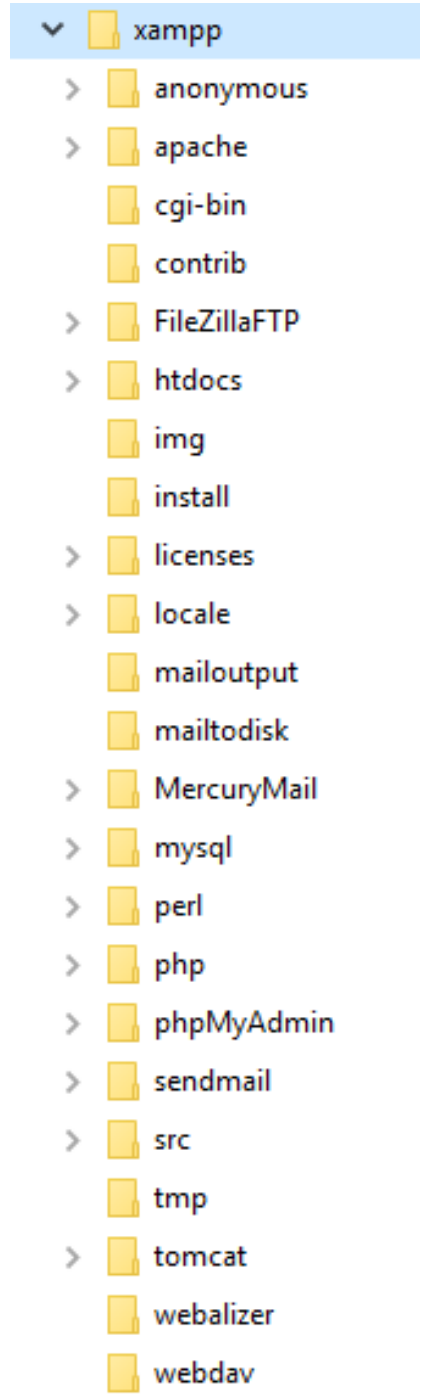
Open the browser at address (URL):

- http: //localhost/dagensdato.php

The PHP interpreter (in XAMPP) executes the script.

The result is a (dynamic) HTML page:

<h1> Today's date: 13.04.2021 </h1>



Quizz on *Web Applications* (part 1)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

Link:

- https://mitt.uib.no/courses/27455/quizzes

Look at the PHP configuration

The phpinfo function shows how PHP is installed:

```
<html>
<body>
<?php
  phpinfo();
?>
</body>
</html>
```

Variables

First: Hi Ola

Then: Hi Kari

- All variable names start with $
- The dot merges text values
- Variables can change value (they can vary)

```
$name = 'Ola';
echo '<h1>Hi ' . $name . '</h1>';
$name = 'Kari';
echo '<h1>Hi ' . $name . '</h1>';
```

Single and double quotes

A text enclosed in double apostrophes may contain variables

- The **PHP engine replaces** such variables with their value.
- So it is **Ola** and not **$name** that is printed.
- Using double quotes makes it easy to print variable contents.

```
<html>
<body>
<?php
  $navn = 'Ola';
  echo "<h1>Hei $navn </h1>";
?>
</body>
</html>
```

Some operators

Arithmetic operators

- `+ - * / %` — `e.g.` 10%3 gives 1 (rest operator of Euclidean division)

Comparison operators

- `> >= == != < <=` — == tests for equality

Increment and decrement

- `++ --` — +1 and -1

Logical operators

- `&& || !` — AND – OR – NOT

Some built-in functions

Text

- str_pad, strlen, substr, substr_replace, trim, ucfirst, …

Date and time

- date, getdate, gettimeofday, strtotime, …

Mathematics

- abs, ceil, cos, exp, floor, log, pi, rand, round, sin, sqrt, tan, …

…

Examples:

```
$s = substr('abcdef', 1, 3);  // gives $s == 'bcd'
$today = getdate();
$avst = sqrt(exp($x2-$x1,2) + exp($y2-$y1,2));
```

15 minute break!
Lecture resumes at 11:10

Program 2: Cinema ticket price (selection)

Reading user input will come later.

The program prints the ticket price.

```
$alder = 23;                    // Simulate input
$pris = 80;                     // Full price
if ($alder < 7) {
  $pris = 0;                    // [0..7> get free entrance
}
else {
  if ($alder <= 12 || $alder > 67)
    $pris = $pris * 0.5;        // Half price
}
echo "Pris: $pris";
```

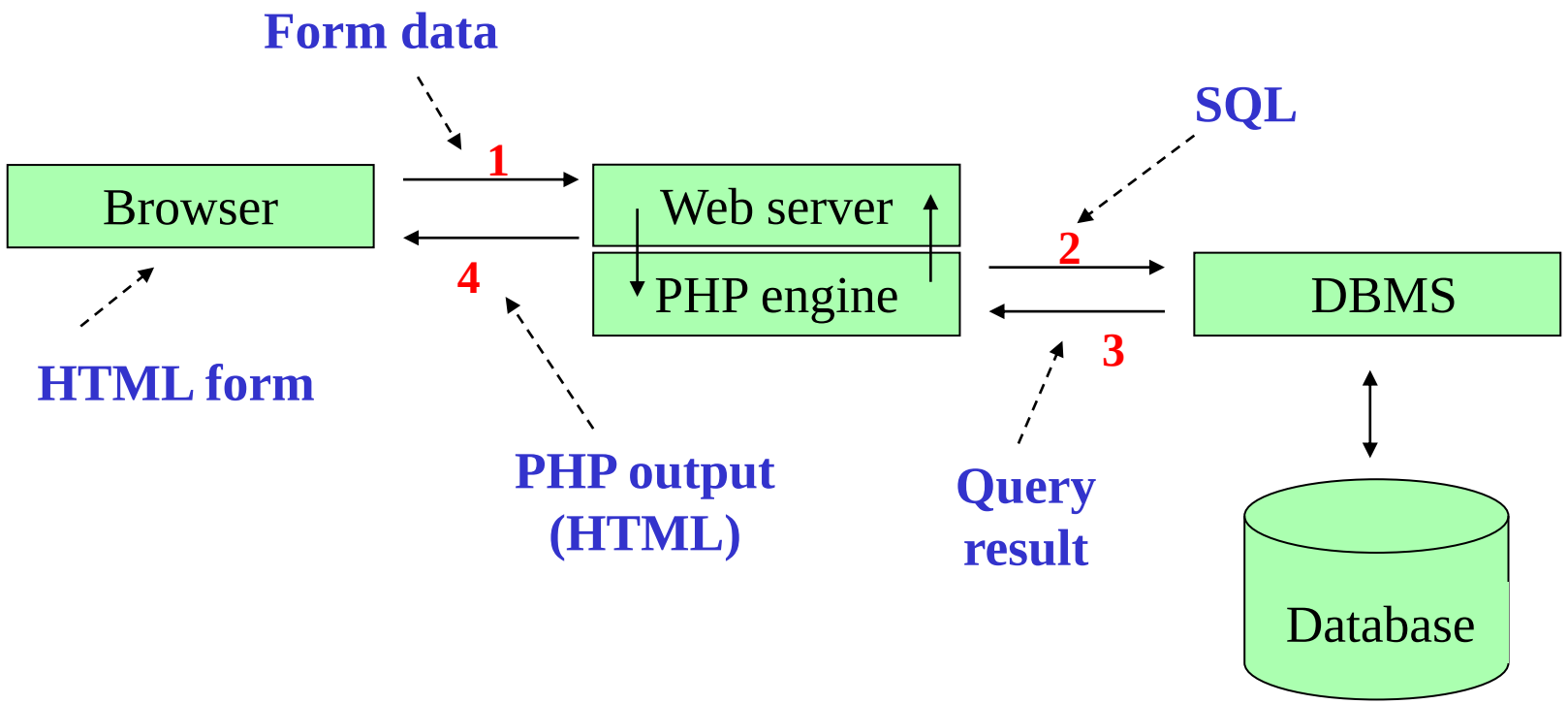
Program 3: List of names (loop, repetition)

The program prints all names in a table (array).

```
$navnTabell =
  array("Per", "Kari", "Ola", "Lise", "Karianne");

echo "<table width=\"1\">";
foreach ($navnTabell as $fornavn)
{
  echo "<tr>";
  echo "<td>$fornavn</td>";
  echo "</tr>";
}
echo "</table>";
```

Database-driven web solutions with PHP



The web server is a "server" for the browser and a "client" for the DBMS

- **Form data** becomes **SQL** becomes Q**uery result** becomes **HTML** …

Program 4: View from the database

Explanations follow on the next slides … but the typical layout is:

- Open connection, run SQL, show query result, close connection

```
$conn = mysqli_connect("localhost", "bruker", "passord", "db" );
$sql = "SELECT * FROM Vare";
$resultat = mysqli_query( $conn, $sql );

$rad = mysqli_fetch_assoc($resultat);
while ( $rad)
{
  $navn = $rad["Betegnelse"];
  echo "<p>$navn</p>";
  $rad = mysqli_fetch_assoc($resultat);
}
mysqli_close( $conn );
```

Open and close the database connection

To connect to a MySQL server we need:

- The address of the server (possibly localhost)
- Username
- Password

A MySQL server can contain multiple databases:

- Must select database

Open the connection to the database:

```
$conn= mysqli_connect(
  "localhost", "bruker", "passord", "db"
);
```

Close the connection to the database:

```
mysqli_close( $conn );
```

Perform SELECT queries

Sends the SQL query as a parameter to *mysqli_query*:

```
$sql = "SELECT * FROM Vare";
$result = mysqli_query( $conn, $sql );
```

What does *$result* contain?

- It contains the entire query result
- So we can go through the query result row by row with a loop and retrieve values

Processing query results

Assume *$resultat* is a query result with amongst others a column *Betegnelse*:

```
// Get the first row
$rad = mysqli_fetch_assoc($resultat);

while ( $rad ) // As long as there are multiple rows
{
  $navn = $rad["Betegnelse"];  // Get a name
  echo "<p>$navn</p>";         // Print it

  // Fetch the next row
  $rad = mysqli_fetch_assoc($resultat);
}
```

Quizz on *Web Applications* (part 2)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

Link:

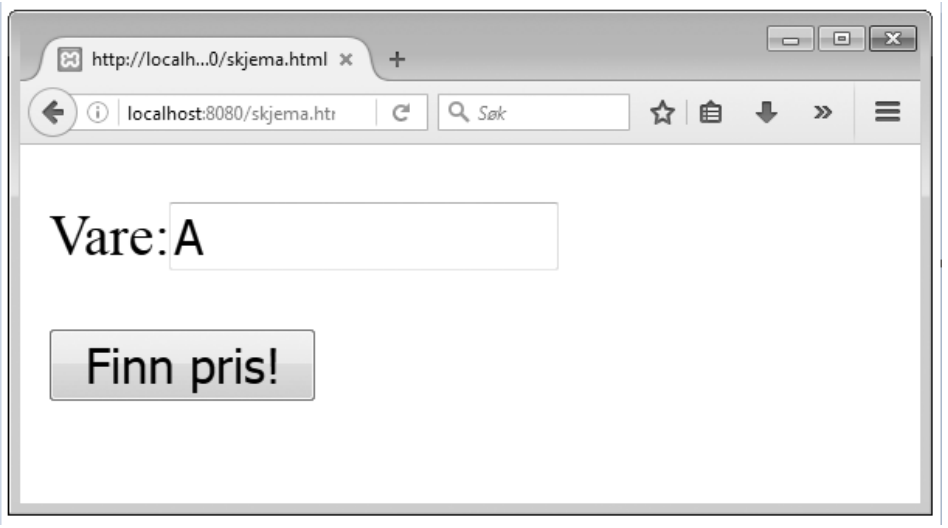
- https://mitt.uib.no/courses/27455/quizzes

An HTML Form

- User writes an **item name** in the text field and clicks "Find price!"
- The name is sent to **varepriser.php** using the **POST** method.

HTML: HyperText Markup Language

```
<!DOCTYPE html>
<html>
<body>
  <form method = "POST" action = "varepriser.php">
    <p>
      Vare:<input type="text" name="vare" size="15"/>
    </p>
    <p>
      <input type="submit" value="Finn pris!"/>
    </p>
  </form>
</body>
</html>
```



Collecting **Input** from the HTML form

We have created a website with an HTML Form

- Contains a text field called **vare** (name attribute)
- The form is sent to **varepriser.php** (action attribute)

In varepriser.php we get input from the user:

```
$varenavn = $_POST ['vare'];
```

The variable **$varenavn** can be used to create a query:

```
$sql = "SELECT * FROM Vare
    WHERE Betegnelse LIKE '$varenavn %' ";
```

Query if the user wrote A in the HTML form:

```
SELECT * FROM Vare WHERE Betegnelse LIKE 'A%'
```

Collecting **Input** from the HTML form

We have created a website with an HTML Form

- Contains a text field called **vare** (name attribute)
- The form is sent to **varepriser.php** (action attribute)

In varepriser.php we get input from the user:

```
$varenavn = $_POST ['vare'];
```

The variable **$varenavn** can be used to create a query:

```
$sql = "SELECT * FROM Vare
    WHERE Betegnelse LIKE '$varenavn %' ";
```

Query if the user wrote A in the HTML form:

```
SELECT * FROM Vare WHERE Betegnelse LIKE 'A%'
```

Collecting **Input** from the HTML form

We have created a website with an HTML Form

- Contains a text field called **vare** (name attribute)
- The form is sent to **varepriser.php** (action attribute)

In varepriser.php we get input from the user:

```
$varenavn = $_POST ['vare'];
```

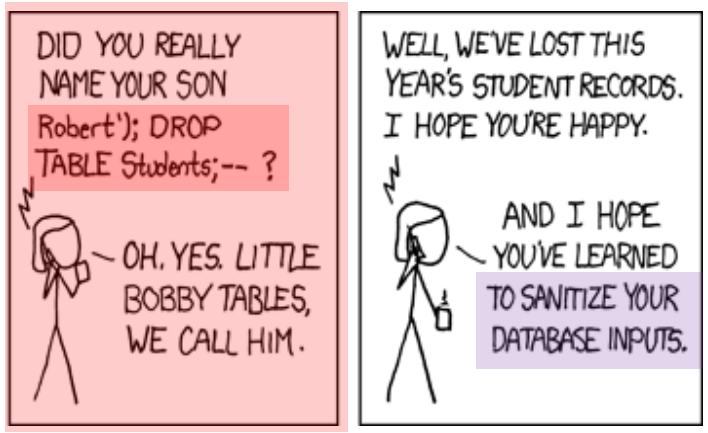
The variable **$varenavn** can be used to create a query:

```
$sql = "SELECT * FROM Vare
    WHERE Betegnelse LIKE '$varenavn %' ";
```

Query if the user wrote A in the HTML form:

```
SELECT * FROM Vare WHERE Betegnelse LIKE 'A%'
```

NB ! Such inputs must first be "**cleaned**" to **prevent hacker attacks**, see section 12.6



Exploits of a Mom (from xkcd.com/327)

Update the database via PHP

We can also use **mysql_query** for updating (**UPDATE**).

- We do not get a query result back, but true / false.

```
$sql = "UPDATE kunde SET etternavn = 'Mo' WHERE knr = 2";
if (mysqli_query($db, $sql)) {
  echo "<p>Etternavn til kunde 2 er endret til Hansen.</p>";
}
else {
  echo "<p>Oppdatering av etternavn feilet.</p>";
}
```

We can perform INSERT and DELETE in a similar way.

We can use input from the user in the SQL query.

Summary: *Web Applications*

- **Technologies** that the **internet** and web are based on.
- **HTML structure and syntax.**
- Understand the **connection** between *information systems*, *database systems* and *web applications*.
- Create **simple PHP scripts** based on examples.
- Techniques for **securing** web applications.

Third Hand-In Assignment

You will connect to and query a database using PHP.

We recommend using XAMPP to setup a PHP development environment:

https://www.apachefriends.org/index.html