banana

What is Mysql

- Relational Database
- Supports standard SQL (Structured Query Language)
- Websites like Facebook, Twitter, Wikipedia

Connecting to MySQL

- **MySQLi**
- **PDO**

Syntax: MySQLi, Procedural way

```
$link = mysqli_connect("hostname", "username", "password",
                       "database");
```

Syntax: MySQLi, Object Oriented way

```
$mysqli = new mysqli("hostname", "username", "password",
                     "database");
```

PDO

- **PDO**

Syntax: PHP Data Objects (PDO) way

```
$pdo = new PDO("mysql:host=hostname;dbname=database",
                    "username", "password");
```

Connecting to MySQL

Driver name	Supported databases
PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D

http://php.net/manual/en/pdo.drivers.php

Connecting to MySQL with PDO

```
define('DB_SERVER', 'localhost:3307');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', 'banana');
define('DB_NAME', 'pos_db');

try{
    $pdo = new PDO("mysql:host=" . DB_SERVER . ";dbname=" .
                DB_NAME, DB_USERNAME, DB_PASSWORD);
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
                PDO::ERRMODE_EXCEPTION);
    echo 'Connect Successfully';
} catch(PDOException $e){
    die("ERROR: Could not connect. " . $e->getMessage());
}
```

Insert to MySQL with PDO

```
try{
    $sql = "INSERT INTO customers (first_name, last_name, email)
                    VALUES ('Peter', 'Parker', 'peterparker@mail.com')";
    $pdo->exec($sql);
    echo "Records inserted successfully.";
} catch(PDOException $e){
    die("ERROR: Could not able to execute $sql. " . $e->getMessage());
}
```

Select from MySQL with PDO

```
$sql = "select * from customers";
$smt = $pdo->query($sql);
if($smt->rowCount() > 0){
   while($row = $smt->fetch()){
          echo  $row['id'] . ', ';
          echo $row['first_name']  . ', ';
          echo $row['last_name'] . ', ';
          echo $row['email'] . "</br>";
   }
}
unset($smt);
```

SQL Query - sub query

```
SELECT product_id,product_name, MAX(product_price)  FROM
products
```

```
SELECT product_id,product_name, product_price  FROM products
WHERE product_price = (
    SELECT MAX(product_price)
    FROM products
)
```

SQL Query - join

```
SELECT * FROM products p INNER JOIN orders o ON
    p.product_id=o.product_id
```

```
SELECT * FROM products p LEFT JOIN orders o ON
    p.product_id=o.product_id WHERE ISNULL(o.product_id)
```

Sql Query join

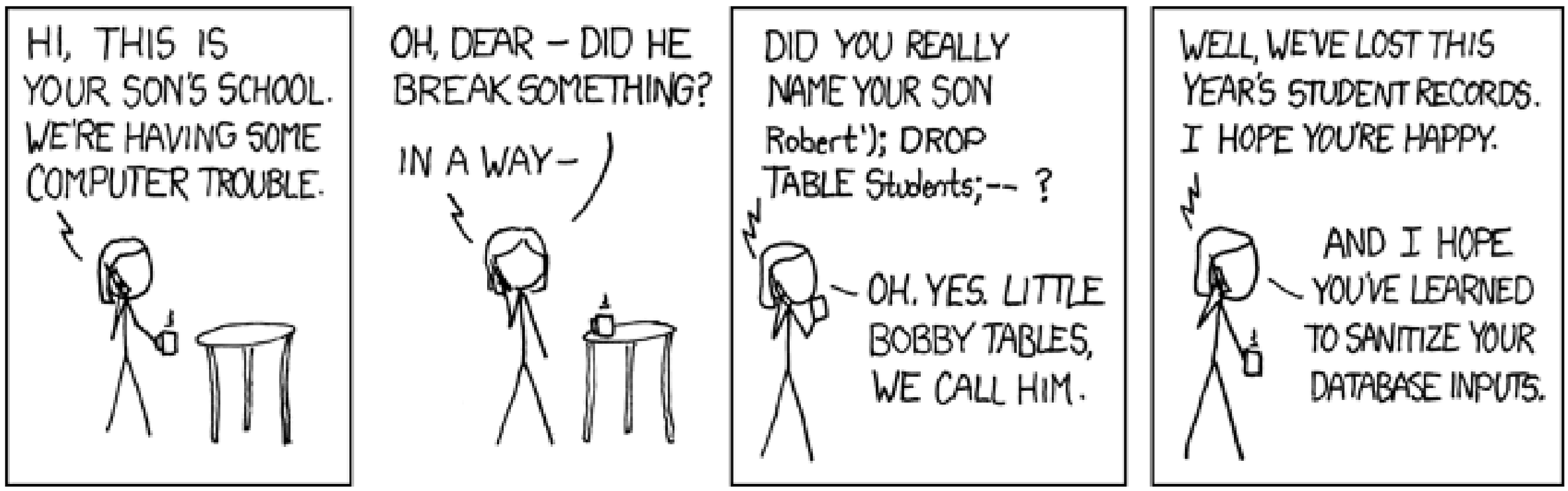
```
SELECT * FROM products p LEFT JOIN orders o ON
p.product_id=o.product_id   WHERE    customer_id='1'
AND  ISNULL(o.product_id)
```

```
SELECT * FROM products p LEFT JOIN orders o ON
p.product_id=o.product_id  AND  customer_id='1'
WHERE  ISNULL(o.product_id)
```

PHP Web Application Security

- SQL Injection
- Cross-Site Scripting (XSS)
- Cookies HttpOnly
- Cross-Site Request Forgery (CSRF)

SQL Injection



SQL injection techniques

1. Boolean based blind

   e.g. **100 or 1=1**

2. Union query

   e.g. **1 union select @@version , database() , user()**

3. Batched queries

   e.g. **1; DROP table students;**

   e.g. **Robert');drop table students;--**

Escaping input prevents SQL Injection

```
$password=$_POST['password'];
$newstr = filter_var($password, FILTER_SANITIZE_STRING);

function sanitize($string){
    $string=strip_tags($string);
    $string=htmlspecialchars($string);
    $string=trim(rtrim(ltrim($string)));
    $string= addslashes ($string);
    return $string;
}
$password= sanitize ($_POST['password']);
```

Use PDO Prevents SQL Injection

```
$sql = "INSERT INTO customers (first_name, last_name, email)
        VALUES (:first_name, :last_name, :email)";
$stmt = $pdo->prepare($sql);

// Bind parameters to statement
$stmt->bindParam(':first_name', $first_name, PDO::PARAM_STR);
$stmt->bindParam(':last_name', $last_name, PDO::PARAM_STR);
$stmt->bindParam(':email', $email, PDO::PARAM_STR);

// Set the parameters values and
$first_name = "Ron";
$last_name = "Weasley";
$email = "ronweasley@mail.com";
$stmt->execute();
echo "Records inserted successfully.";
```

Cross-Site Scripting (XSS)

- Reflected XSS

- Stored XSS

- DOM-based XSS

Prevent XSS

- Strip HTML Tags

```
$str = '<p>Test paragraph.</p><!-- Comment -->
        <a href="#fragment">Other text</a>';
$string = strip_tags($str);
//Test paragraph. Other text
```

- Strip HTML Tags

```
$str='<p>Test paragraph.</p><!-- Comment -->
        <a href="#fragment">Other text</a>';
$string = strip_tags($str);
$string = htmlspecialchars ($str);

//“<p>Test paragraph.</p><!-- Comment -->
        <a href="#fragment">Other text</a>”
```

HttpOnly Cookie

- Session.cookie_httponly = true

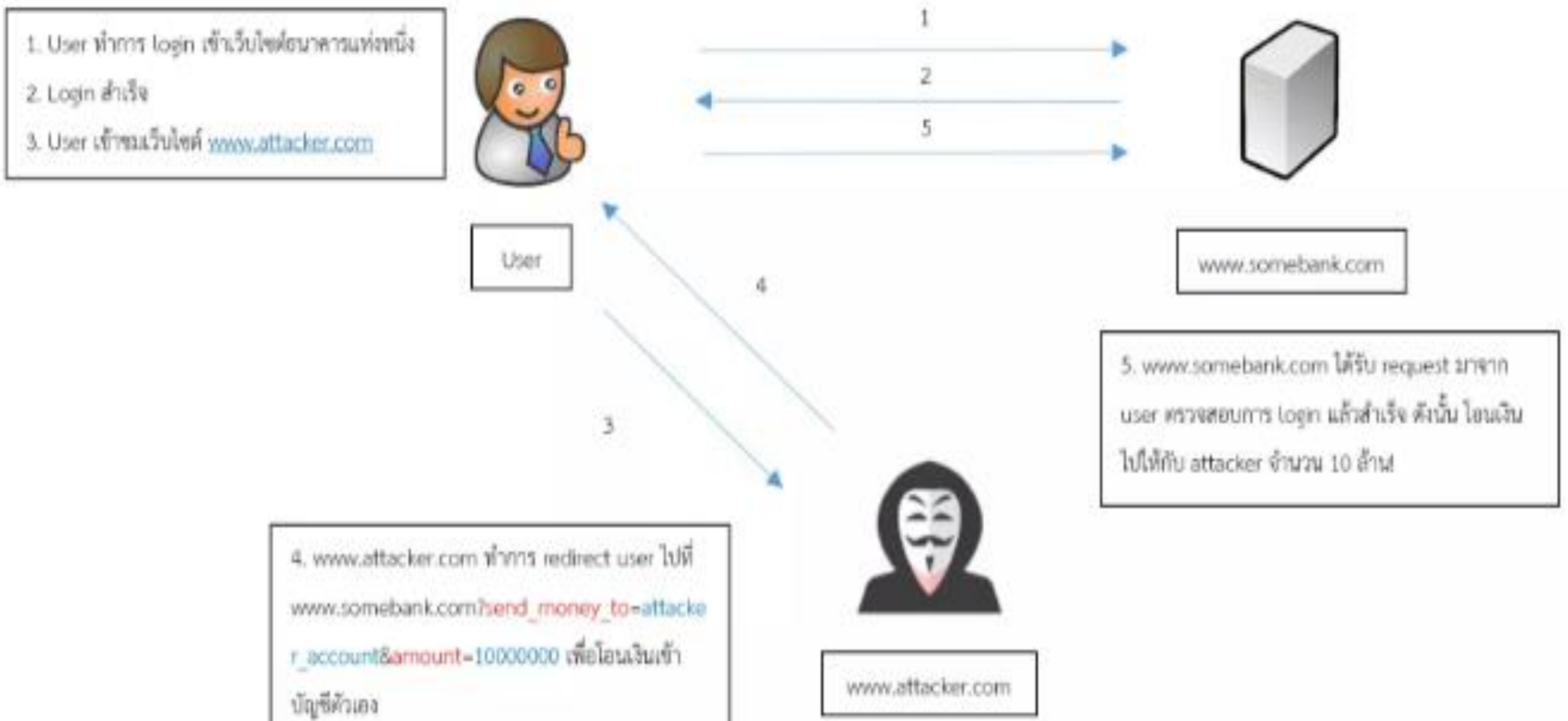
- setcookie(*name,value,expire,path,domain,secure,httponly*);

e.g.

```
// secure
setcookie('Cookie1', rand(100, 999), 0, '/', '', true, true);

// not secure
setcookie('Cookie2', rand(100, 999), 0, '/', '', false, false);
```

CSRF (Cross Site Request Forgery)



Prevent CSRF (Cross Site Request Forgery)

1. **Synchronizer Token Pattern**

```
<form>
<input type="hidden" value="aSecretTokenRandomlyGeneratedForThisSession" />
<input type="password" />
</form>
```

2. **HTTP referer header**

3. **Re-authentication & CAPTCHA**

PHP Ajax

```
$sql = "select * from customers";
$smt = $pdo->query($sql);
if($smt->rowCount() > 0){
    $datas=array();
    while($row = $smt->fetch()){
        $data['id']=$row['id'];
        $data['firstName']=$row['first_name'];
        $data['lastName']=$row['last_name'];
        $data['email']=$row['email'];
        $datas[]=$data;
}
echo json_encode($datas);
unset($smt);
```

End