

Pengembangan Aplikasi Web

Pertemuan Ke-6 (Keamanan Aplikasi *Web*)

Noor Ifada

`noor.ifada@{trunojoyo.ac.id, if.trunojoyo.ac.id}`

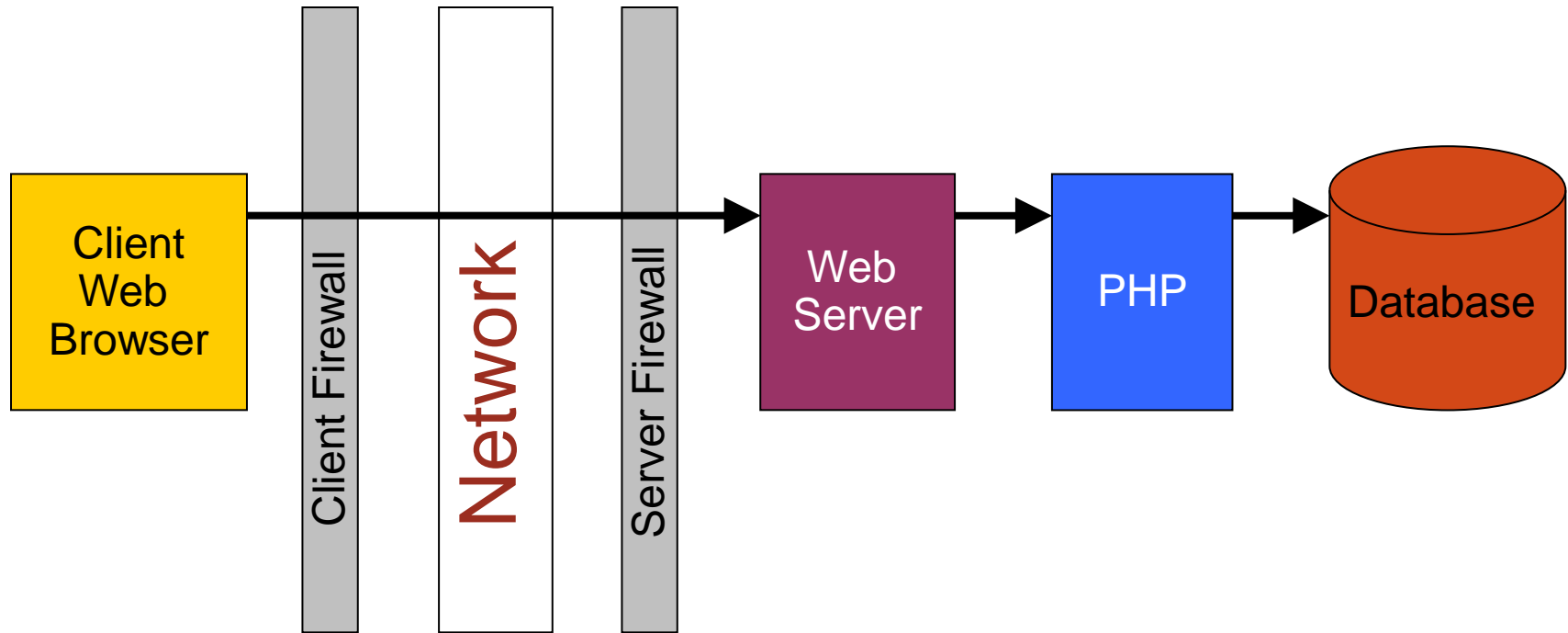
Sub Pokok Bahasan

- Prinsip Keamanan
- Keamanan *End-to-End*
- Keamanan Basisdata
- Keamanan Aplikasi *Web*
- *Session*
- Menggunakan *Session* untuk *Authorization*

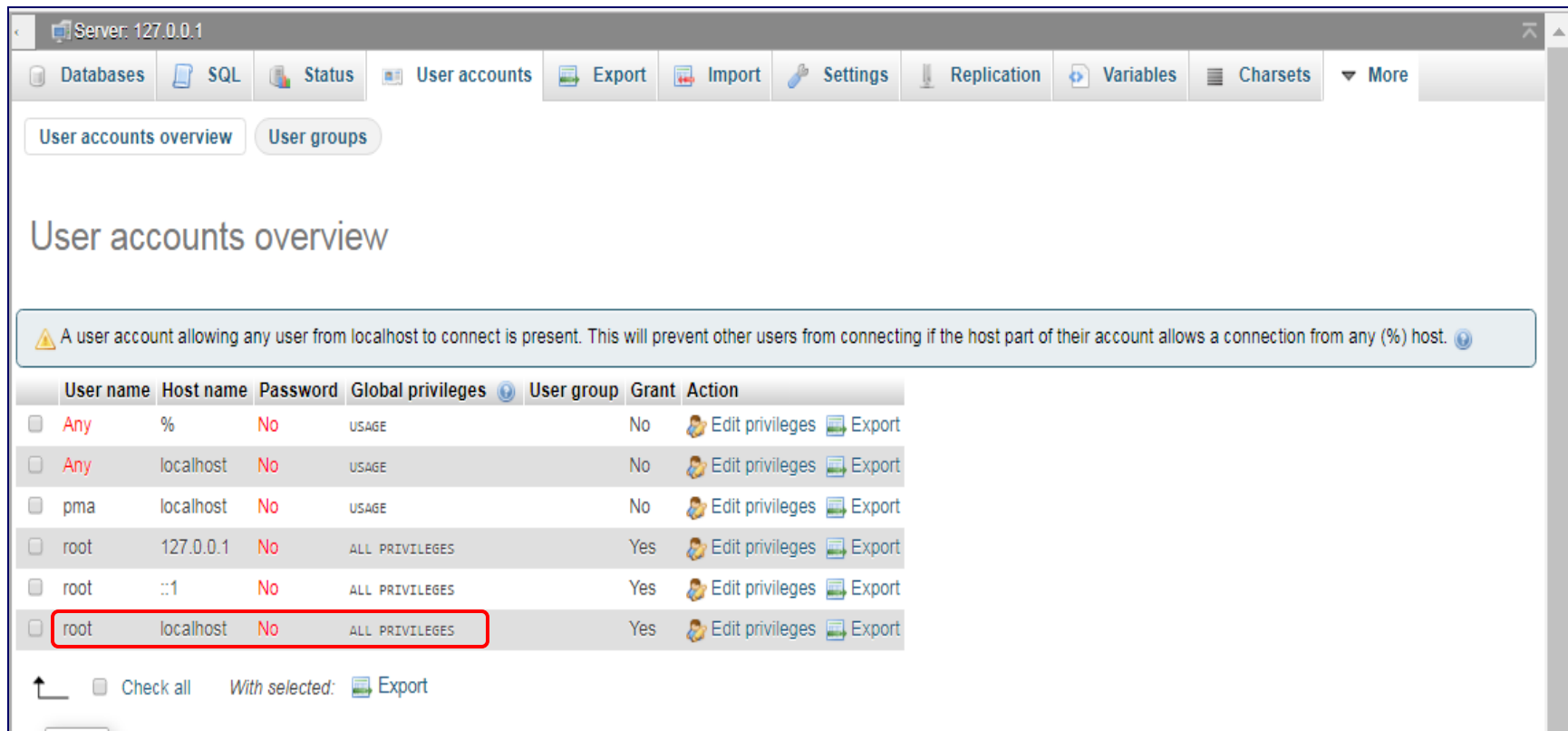
Prinsip Keamanan

- **Confidentiality**
 - Mencegah kebocoran data (data tidak dapat diakses oleh pihak yang tidak berwenang)
- **Integrity**
 - Melindungi konsistensi data
- **Availability**
 - Memastikan bahwa data selalu tersedia ketika dibutuhkan
- **Authenticity**
 - Memvalidasi identitas *user* (pihak yang ingin mengakses data)
- **Non-repudiation**
 - Mencegah penolakan transaksi

Keamanan *End-to-End*



Keamanan Basisdata



Server: 127.0.0.1

Databases SQL Status User accounts Export Import Settings Replication Variables Charsets More

User accounts overview User groups

User accounts overview

⚠ A user account allowing any user from localhost to connect is present. This will prevent other users from connecting if the host part of their account allows a connection from any (%) host. ⓘ

	User name	Host name	Password	Global privileges ⓘ	User group	Grant	Action
<input type="checkbox"/>	Any	%	No	USAGE		No	Edit privileges Export
<input type="checkbox"/>	Any	localhost	No	USAGE		No	Edit privileges Export
<input type="checkbox"/>	pma	localhost	No	USAGE		No	Edit privileges Export
<input type="checkbox"/>	root	127.0.0.1	No	ALL PRIVILEGES		Yes	Edit privileges Export
<input type="checkbox"/>	root	::1	No	ALL PRIVILEGES		Yes	Edit privileges Export
<input type="checkbox"/>	root	localhost	No	ALL PRIVILEGES		Yes	Edit privileges Export

⬆ ☐ Check all With selected: Export

```
1 <?php
2 $dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
3
4 // Use the connection ...
5
6 ?>
```

Keamanan Aplikasi Web

- Gunakan mekanisme pengamanan *password* yang baik
- Waspada terhadap berbagai bentuk serangan keamanan aplikasi *web*

Mekanisme Pengamanan Password

- Tidak menyimpan data *password* dalam bentuk “*clear text*” di dalam basisdata
- Gunakan fungsi/function *hashing* untuk enkripsi data: **md5**, **sha2**, ...

Stored password: **Today123**

SHA2

fa82bc6820fcc9098dff6ea2ec5b829e14944500bab3f4be19d029d41cb8031

```
SELECT * FROM admin_password
WHERE username = :username
and password = SHA2(:password, 0)
```

Don't Match!

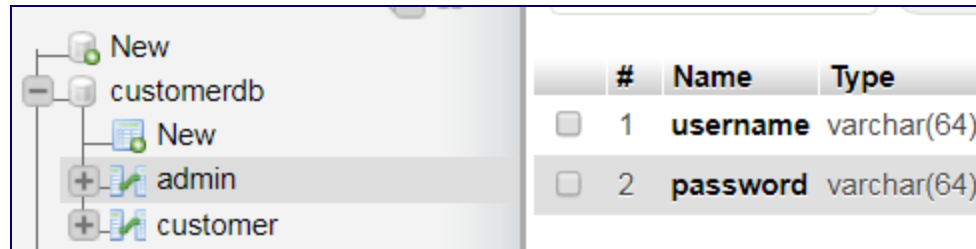
Entered password: **today123**

SHA2

685cb7bec82cc19f15e9a1246a3ad813912a36905f248d26925bd8680a65d5a4

Mekanisme Pengamanan *Password* [2]

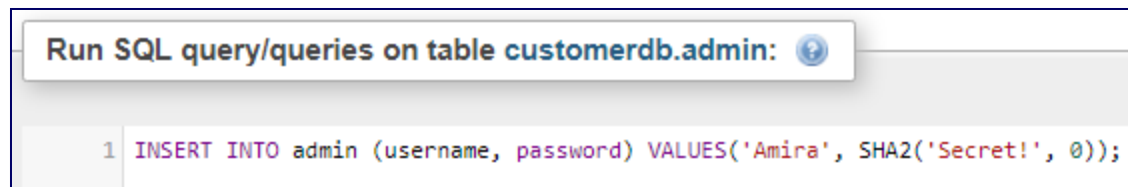
- Buat tabel untuk menyimpan data *user* dan *password*. Contoh: tabel **admin**



The screenshot shows a database management interface. On the left, a tree view displays the database structure: 'New' (database), 'customerdb' (database), 'New' (table), 'admin' (table), and 'customer' (table). The 'admin' table is selected. On the right, a table shows the structure of the 'admin' table:

#	Name	Type
<input type="checkbox"/> 1	username	varchar(64)
<input type="checkbox"/> 2	password	varchar(64)

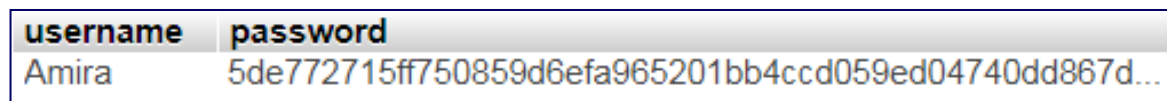
- Tambahkan data ke tabel **admin**



The screenshot shows a SQL query editor with the following text:

```
Run SQL query/queries on table customerdb.admin: ?
```

```
1 INSERT INTO admin (username, password) VALUES('Amira', SHA2('Secret!', 0));
```



The screenshot shows a table with the following data:

username	password
Amira	5de772715ff750859d6efa965201bb4ccd059ed04740dd867d...

Bentuk Serangan Keamanan Aplikasi Web

- SQL Injection
- Script Injection
- XML Attack
- Implementasi sistem keamanan yang lemah
- Developer Problem

SQL Injection

- Jangan menggunakan pernyataan SQL yang menggabungkan masukan *user* secara langsung → gunakan PDO *Prepared Statements*: `prepare()`, `bindValue()`, `execute()`

```
Run SQL query/queries on table customerdb.customer: ?  
  
1 SELECT *  
2 FROM customer  
3 WHERE customerID=1;
```



customerID	firstname	address	balance
1	Almira	Jl. Mawar No. 123, Surabaya	4500000

```
Run SQL query/queries on table customerdb.customer: ?  
  
1 SELECT *  
2 FROM customer  
3 WHERE customerID=1 or true;
```



customerID	firstname	address	balance
1	Almira	Jl. Mawar No. 123, Surabaya	4500000
2	Baharudin	Jl. Mengkudu No. 456, Surabaya	1200000
3	Citra	Jl. Cendana No. 17, Surabaya	0
4	Derry	Jl. Delima No. 50, Surabaya	0
5	Erlina	Jl. Erlangga No. 44, Surabaya	0

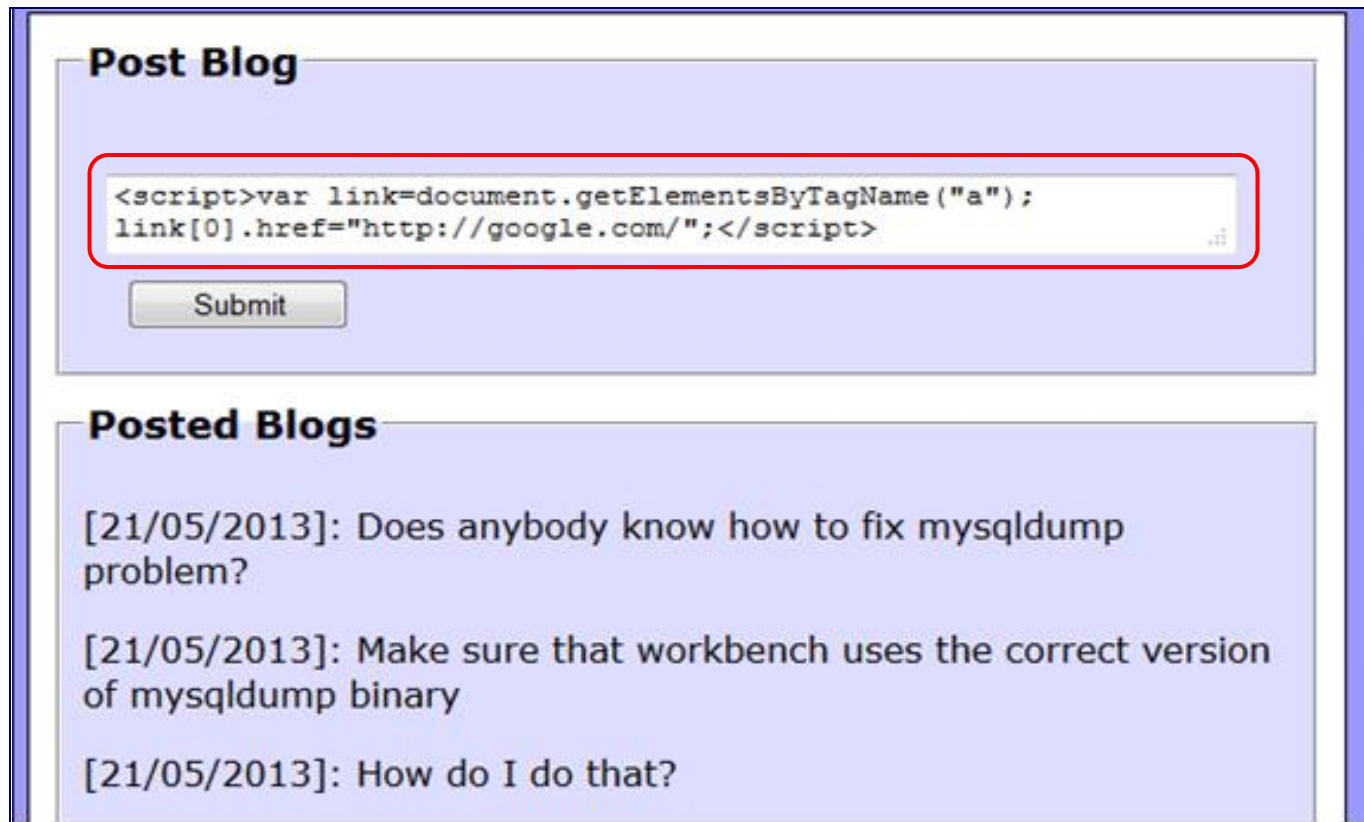
```
Run SQL query/queries on table customerdb.customer: ?  
  
1 SELECT *  
2 FROM customer  
3 WHERE customerID=1 or 1=1;
```



customerID	firstname	address	balance
1	Almira	Jl. Mawar No. 123, Surabaya	4500000
2	Baharudin	Jl. Mengkudu No. 456, Surabaya	1200000
3	Citra	Jl. Cendana No. 17, Surabaya	0
4	Derry	Jl. Delima No. 50, Surabaya	0
5	Erlina	Jl. Erlangga No. 44, Surabaya	0

Script Injection

- Jangan secara langsung menampilkan masukan *user* yang berupa skrip → gunakan fungsi/*function* **htmlspecialchars**



The screenshot shows a web application interface. At the top, there is a section titled "Post Blog" with a light blue background. Inside this section, there is a text input field containing the following JavaScript code: `<script>var link=document.getElementsByTagName("a"); link[0].href="http://google.com/";</script>`. The input field is highlighted with a red border. Below the input field is a "Submit" button. Below the "Post Blog" section, there is another section titled "Posted Blogs" with a light blue background. This section contains a list of three blog posts, each starting with a date in brackets: "[21/05/2013]: Does anybody know how to fix mysqldump problem?", "[21/05/2013]: Make sure that workbench uses the correct version of mysqldump binary", and "[21/05/2013]: How do I do that?".

XML Attack

- XML *attack* adalah *Denial-of-Service* (DoS) *attack*
- Merupakan XML schema yang *well-formed* dan *valid*

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]
<lolz>&lol9;</lolz>
```

Implementasi keamanan yang lemah

- Tidak mengimplementasikan sistem keamanan terbaru (*Web Browser, Web Server, ...*)
- Tidak mengimplementasikan mekanisme *password* yang baik
- Hanya mengandalkan validasi *client-side*

Developer Problem

- *Error reporting* tidak diaktifkan
- Variabel tanpa inisialisasi nilai

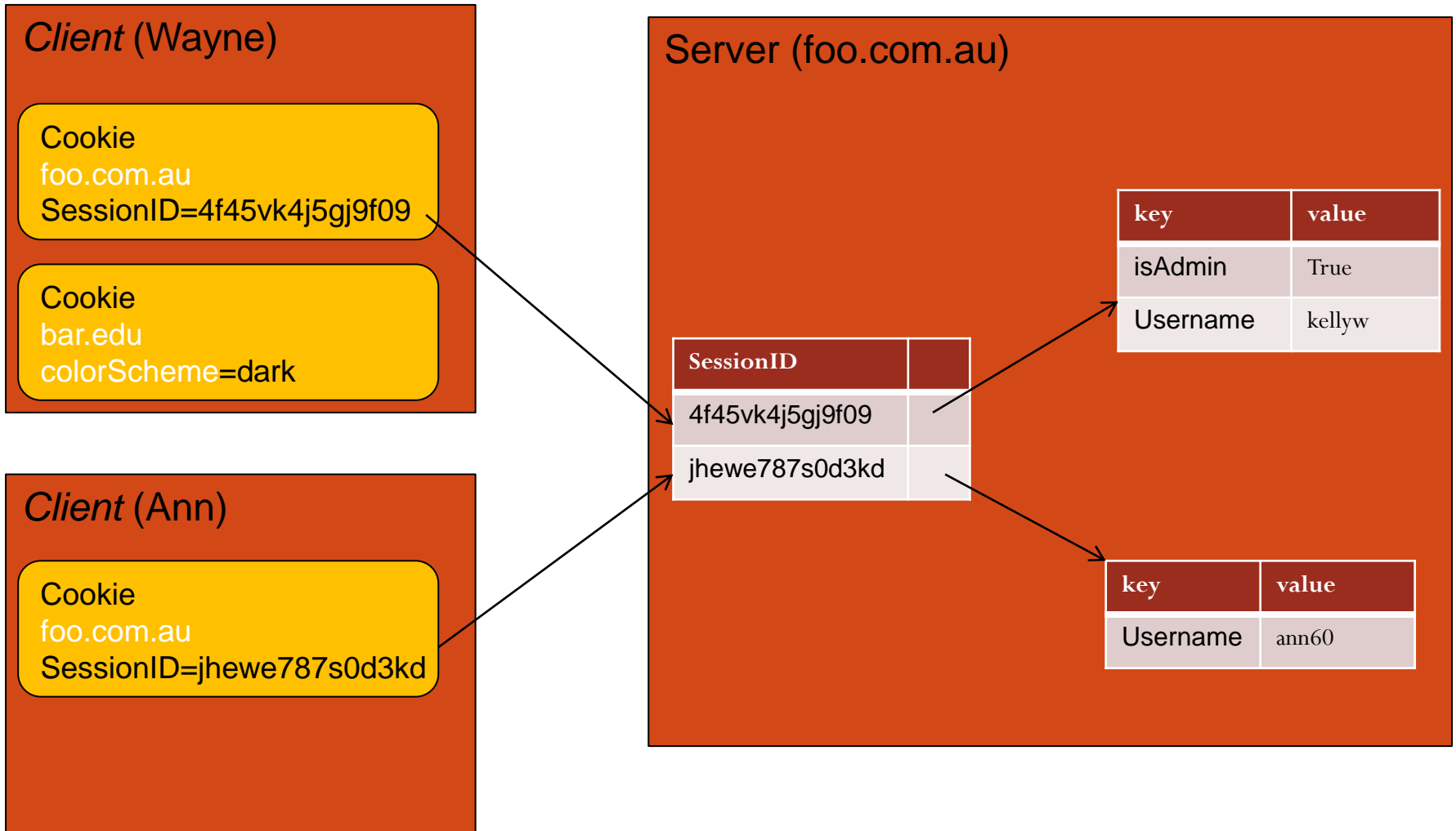
```
<?php
    $intNoValue ;
    if ($intNoValue == 0 )
        echo "<p>Equals Zero</p>";
?>
```

- Variabel global: `php.ini` → `register_global = On`

```
<legend>Login</legend>
<div class="field"> <!-- Name field and its error message -->
    <label for="username">Username:</label>
    <input type="text" name="strusername" id="username" />
</div>
<div class="field"> <!-- Password field and its error message -->
    <label for="password">Password:</label>
    <input type="password" name="strpassword" id="password" />
</div>
<div class="field">
    <input class="specialsubmit" type="submit" name="login" value="Login"/>
    <input class="specialsubmit" type="hidden" name="intOkay" value="1"/>
</div>
```

```
<?php
    if (checkUser($strusername $strpassword))
        $intOkay = 1;
    if ($intOkay)
        echo "<p>Valid user</p>";
    function checkUser ($username, $password) {
        return false;
    }
?>
```

Session



Menggunakan *Session* untuk *Authorization*

- Tentukan halaman mana yang dapat dilihat langsung oleh *user* (publik) dan mana yang memerlukan *login* (non-publik)
- Untuk halaman non-publik, tambahkan skrip seperti contoh berikut:

```
<?php require 'admin_permission.inc' ?>
<DOCTYPE html>
<html>

...
```

admin_permission.inc

```
<?php
    session_start();
    if (!isset($_SESSION['isAdmin'])) // isAdmin is just an example
    {
        // if not authorized then redirect to the login page
        header("Location: https://{$_SERVER['HTTP_HOST']}/login.php");
        exit(); // don't proceed with the rest of this page
    }
?>
```


Halaman LOGIN

- Buat *form* LOGIN

Login

Username	<input type="text"/>
Password	<input type="password"/>
	<input type="button" value="Login"/>

- Halaman **Login.php** harus dikirimkan ke dirinya sendiri (*self-submission*)

```
<?php
    if (isset($_POST['login']))
    {
        // validate and process posted username and password here
    }
?>
<html>
    ...
    <form method="POST" action="Login.php">
        // the user name and password fields go here
    ...
```

Halaman LOGIN [2]

Validasi & pemrosesan masukan data *login*:

- a. Cek apakah *username* dan *password* yang dimasukkan adalah benar

fungsi/function

```
if (checkPassword($_POST['username'], $_POST['password']))
```

- Gunakan **PDO Prepared Statement** untuk memvalidasi masukan data dengan yang ada di dalam basisdata. Skrip SQL:

```
SELECT *  
FROM admins  
WHERE username = :username and password = SHA2(:password, 0)
```

- Ketika skrip SQL menghasilkan keluaran setidaknya satu baris, maka hasil validasi bernilai **true**

```
return $query->rowCount() > 0;
```

Halaman LOGIN [3]

Validasi & pemrosesan masukan data login:

b. Jika pengecekan *username* dan *password* yang dimasukkan adalah benar

- Aktifkan/buka session baru untuk mencatat bahwa seorang *user* “admin” telah berhasil *login*

```
session_start();  
$_SESSION['isAdmin'] = true;
```

- Arahkan *user* “admin” ke halaman non-publik

```
header('Location: http://localhost/Private.php');
```

- *Exit* dari halaman **Login**

```
exit();
```

Halaman LOGOUT

- Buat halaman **Logout.php** untuk:
 - Membersihkan data user dari *session*:

```
session_start();  
unset($_SESSION['isAdmin']);
```

- Menampilkan informasi bahwa *user* telah berhasil *logout*

