Server Side-Client Side

Database Applications and the Web Client Side and Server Side Scripting

The Web

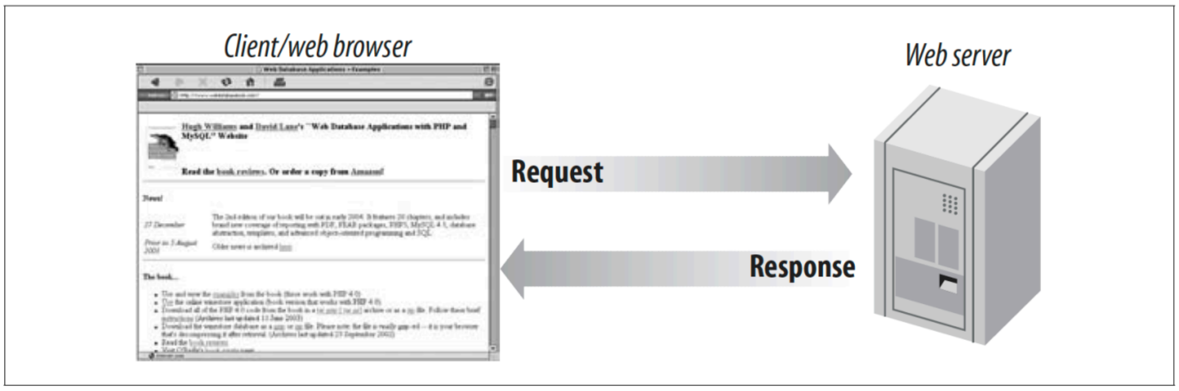


Figure 1-1. A two-tier architecture where a web browser makes a request and the web server responds

Three Tier

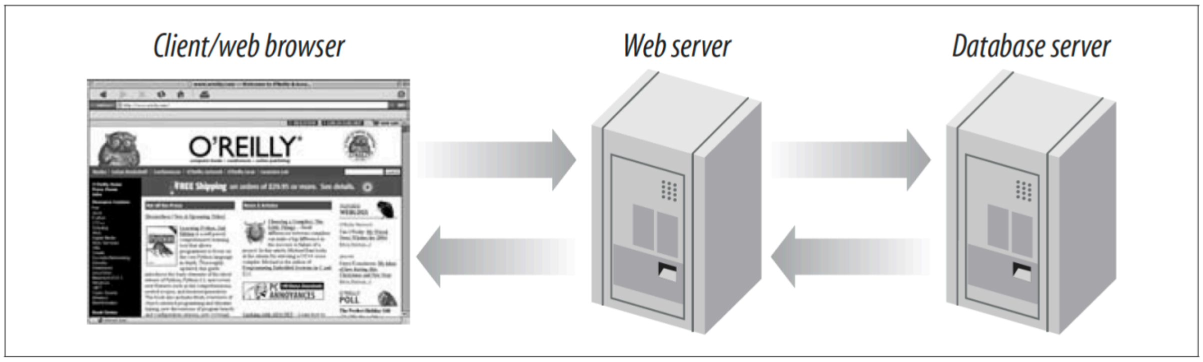


Figure 1-2. A three-tier architecture where a web browser requests a resource, and a response is generated from a database

Three Tier Architecture

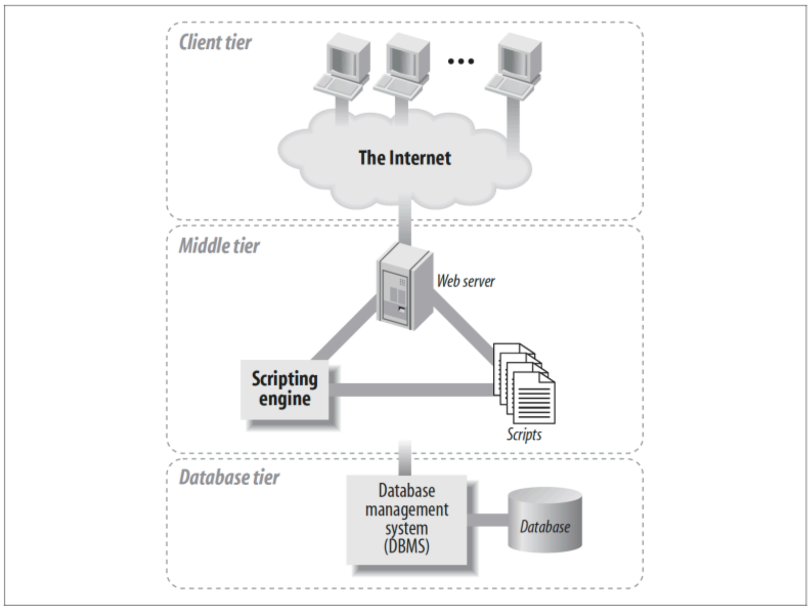
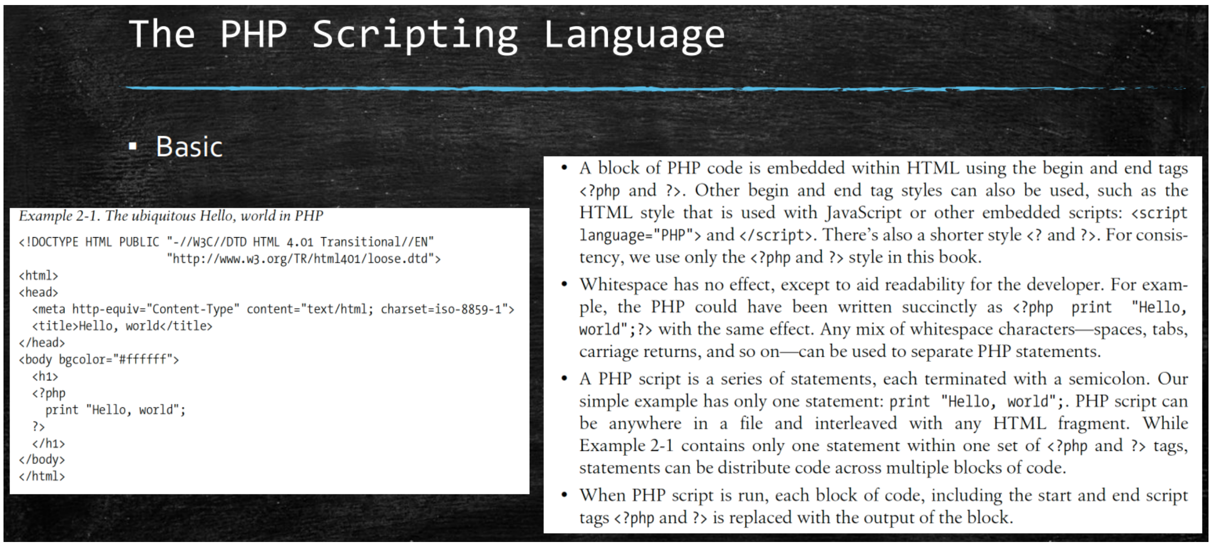


Figure 1-3. The three-tier architecture model of a web database application

Server Side Scripting



The PHP Scripting Language

- Basic

Example 2-1. The ubiquitous Hello, world in PHP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                      "http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <title>Hello, world</title>
</head>
<body bgcolor="#ffffff">
  <h1>
  <?php
    print "Hello, world";
  ?>
  </h1>
</body>
</html>
```

- A block of PHP code is embedded within HTML using the begin and end tags `<?php` and `?>`. Other begin and end tag styles can also be used, such as the HTML style that is used with JavaScript or other embedded scripts: `<script language="PHP">` and `</script>`. There's also a shorter style `<?` and `?>`. For consistency, we use only the `<?php` and `?>` style in this book.
- Whitespace has no effect, except to aid readability for the developer. For example, the PHP could have been written succinctly as `<?php print "Hello, world";?>` with the same effect. Any mix of whitespace characters—spaces, tabs, carriage returns, and so on—can be used to separate PHP statements.
- A PHP script is a series of statements, each terminated with a semicolon. Our simple example has only one statement: `print "Hello, world";`. PHP script can be anywhere in a file and interleaved with any HTML fragment. While Example 2-1 contains only one statement within one set of `<?php` and `?>` tags, statements can be distribute code across multiple blocks of code.
- When PHP script is run, each block of code, including the start and end script tags `<?php` and `?>` is replaced with the output of the block.

Ex syntax:
Client-Side Scripting:
Client-side scripting refers to scripts that run on the user's web browser, typically using JavaScript. These scripts are responsible for enhancing the user interface and interactivity of a website. Here's a simple example of client-side scripting using JavaScript to display an alert when a button is clicked:

html
Copy code
```
<!DOCTYPE html>
<html>
<head>
  <title>Client-Side Scripting Example</title>
</head>
<body>
  <button id="myButton">Click Me</button>

  <script>
    // Client-side JavaScript
    document.getElementById("myButton").addEventListener("click", function() {
      alert("Button clicked!");
    });
  </script>
</body>
</html>
```
In this example, when the "Click Me" button is clicked, a JavaScript function runs in the user's browser, showing an alert.

Server-Side Scripting:
Server-side scripting refers to scripts that run on the web server to generate dynamic content before sending it to the user's browser. Common server-side scripting languages include PHP, Python, Ruby, and more. Here's a simple example using PHP to generate a dynamic web page:

php
Copy code
```
<!DOCTYPE html>
<html>
<head>
  <title>Server-Side Scripting Example</title>
</head>
<body>
  <h1>Welcome to our website, <?php echo "John"; ?></h1>
  <p>Today's date: <?php echo date("Y-m-d"); ?></p>
</body>
</html>
```
In this PHP example, the server processes the PHP code before sending the HTML to the user's browser. The resulting page includes dynamic content, such as the user's name and the current date, which was generated on the server.

Both client-side and server-side scripting are essential in web development, with client-side handling user interactions and interface enhancements and server-side managing dynamic data generation, database interactions, and server-related tasks.

WEB DATABASE: CRUD

INTRODUCTION : Concept

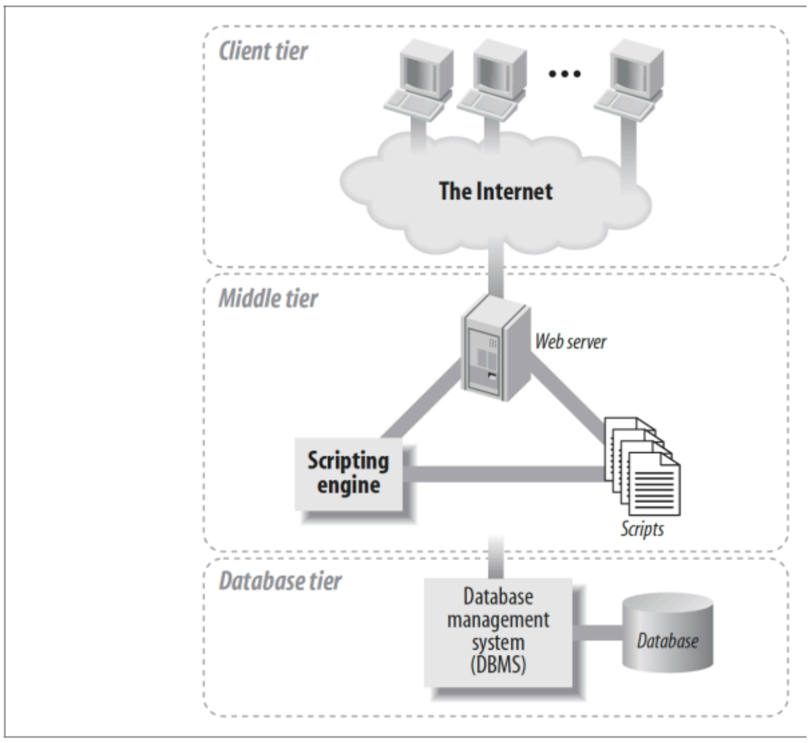
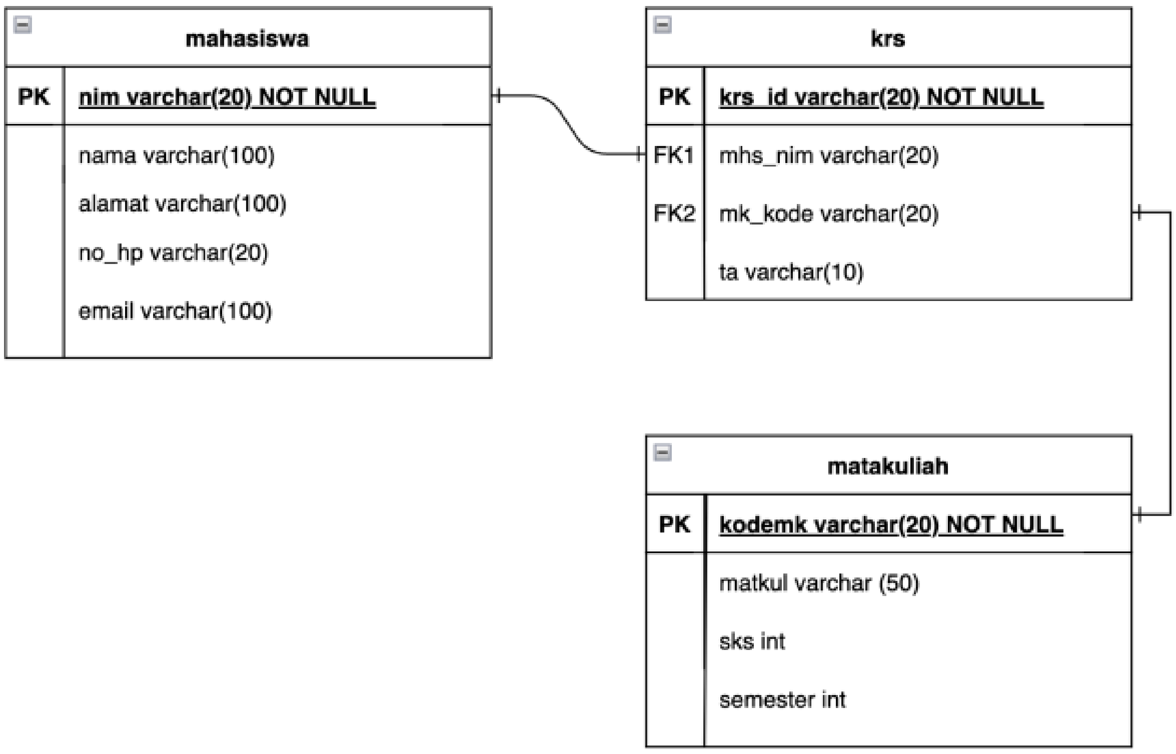


Figure 1-3. The three-tier architecture model of a web database application

RANCANGAN DATABASE



INSERT

Consider an example of the first approach using the *customer* table:

```
INSERT INTO customer VALUES (1,'Williams','Lucy','E',3,
'272 Station St','Carlton North','VIC','3054',12,'(613)83008460',
'2002-07-02');
```

If you want to insert more than one row, you can write more than one INSERT statement. Alternatively, you can write one INSERT statement and separate each row with a comma. Consider an example that uses the latter approach and inserts the details for two customers:

```
INSERT INTO customer VALUES (1,'Williams','Lucy','E',3,
'272 Station St','Carlton North','VIC','3054',12,'(613)83008460',
'2002-07-02'), (2,'Williams','Selina','J',4,'12 Hotham St',
'Collingwood','VIC','3066',12,'(613)99255432','1980-06-03');
```

Data can also be inserted using a second approach. Consider this example:

```
INSERT INTO customer SET cust_id = 1, surname = 'Williams',
   firstname = 'Lucy', initial='E', title_id=3,
   address='272 Station St', city='Carlton North',
   state='VIC', zipcode='3054', country_id=12,
   phone='(613)83008460', birth_date='2002-07-10';
```

VIEW

Consider an example SELECT statement:

```
SELECT surname, firstname FROM customer;
```

SELECT statements can also output data that isn't from a database. Consider the following example:

```
SELECT curtime();
```

To show only the first three regions, you can type:

```
SELECT * FROM region WHERE region_id <= 3;
```

Consider a more complex example:

```
SELECT cust_id FROM customer
  WHERE (surname='Marzalla' AND firstname LIKE 'M%') OR
    birth_date='1980-07-14';
```

UPDATE

Data can be updated using a similar syntax to the INSERT statement. Consider an example:

```
UPDATE customer SET state = upper(state);
```

You can update more than one attribute in a statement. For example, to set both the state and city to uppercase, use:

```
UPDATE customer SET state = upper(state), city = upper(city);
```

The UPDATE statement is also often used with the WHERE clause. For example:

```
UPDATE customer SET surname = 'Smith' WHERE cust_id = 7;
```

This updates the surname attribute of customer #7. Consider a second example:

```
UPDATE customer SET zipcode = '3001' WHERE city = 'Melbourne';
```

This updates the zipcode of all rows with a city value Melbourne.

DELETE

The DELETE statement removes data from tables. For example, the following deletes all data in the *customer* table but doesn't remove the table:

```
DELETE FROM customer;
```

A DELETE statement with a WHERE clause can remove specific rows; WHERE clauses are frequently used in querying, and they are explained later in the section "Querying with SQL SELECT." Consider a simple example:

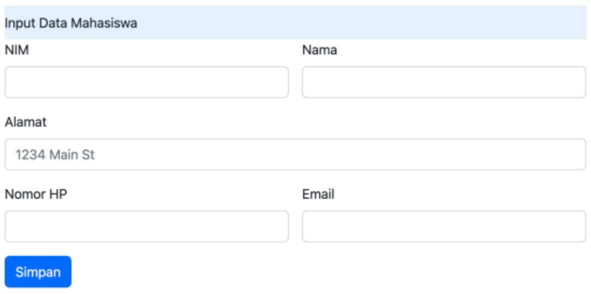
```
DELETE FROM customer WHERE cust_id = 1;
```

This deletes the customer with a cust_id value of 1. Consider another example:

```
DELETE FROM customer WHERE surname = 'Smith';
```

CRUD: CREATE

mahasiswa.php



Beberapa parameter yang diperlukan:

- Form

<form **method**="POST" **action**="simpanmhs.php">

- Input

<input **type**="text" **name**="nim">

Apa perbedaan Method GET dan POST?

SCRIPT INPUT

simpanmhs.php

```php
<?php

include "connection.php";

$nim = $_POST['nim'];
$nama = $_POST['nama'];
$alamat = $_POST['alamat'];
$nohp = $_POST['nohp'];
$email = $_POST['email'];
$query=mysqli_query($conn,"insert into mahasiswa (nim,nama,
alamat, nohp, email) values ( " .$nim." , ' ".$nama." ', '
".$alamat." ', '".$nohp." ', ' ".$email."')");
mysqli_close($conn);

header("location:mahasiswa.php");

?>
```

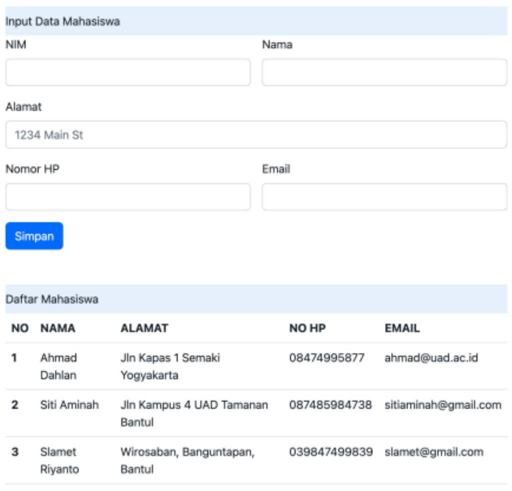
Beberapa parameter yang diperlukan:

- Connection
- Method POST (mengapa POST?)

Apa fungsi header(location:...) pada script di samping?

CRUD: READ
VIEW DATA

mahasiswa.php



Input Data Mahasiswa

NIM	Nama
Alamat: 1234 Main St	
Nomor HP	Email

Simpan

Daftar Mahasiswa

NO	NAMA	ALAMAT	NO HP	EMAIL
1	Ahmad Dahlan	Jln Kapas 1 Semaki Yogyakarta	08474995877	ahmad@uad.ac.id
2	Siti Aminah	Jln Kampus 4 UAD Tamanan Bantul	087485984738	sitiaminah@gmail.com
3	Slamet Riyanto	Wirosaban, Banguntapan, Bantul	039847499839	slamet@gmail.com

Beberapa parameter yang diperlukan:

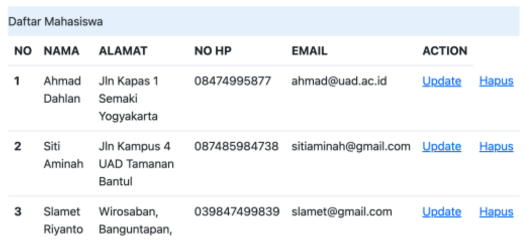
- Connection to database
- mysqli_query
- mysqli_fetch_array

Bagaimana jika menggunakan foreach?

CRUD: UPDATE DELETE

UPDATE DELETE DATA

mahasiswa.php



Daftar Mahasiswa

NO	NAMA	ALAMAT	NO HP	EMAIL	ACTION	
1	Ahmad Dahlan	Jln Kapas 1 Semaki Yogyakarta	08474995877	ahmad@uad.ac.id	Update	Hapus
2	Siti Aminah	Jln Kampus 4 UAD Tamanan Bantul	087485984738	sitiaminah@gmail.com	Update	Hapus
3	Slamet Riyanto	Wirosaban, Banguntapan,	039847499839	slamet@gmail.com	Update	Hapus

Beberapa parameter yang diperlukan:

- Pengiriman variable

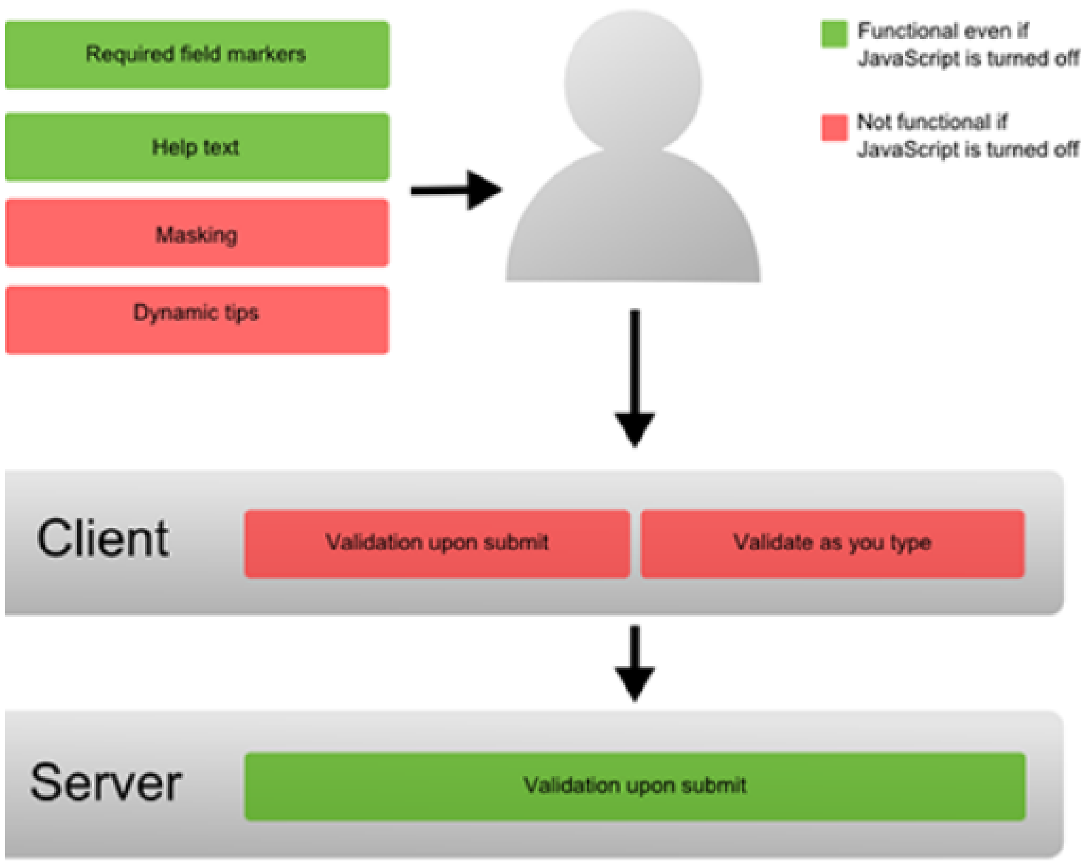
Contoh:

```
<a href='updatemhs.php?id=$row[nim]'>Update</a></td>
```

Bagaimana perintah untuk UPDATE pada form yang sama?
Bagaimana perintah untuk DELETE pada FILE yang sama?

VALIDATION

Validation and Error Reporting Principles
Server- Side Validation with PHP
JavaScript and Client- Side Validation
Concept :



VALIDATION PROCESS:
- **Finding errors**
- **Presenting error messages**

FINDING ERRORS:
- **Interactive**
- **Post-validation**

PRESENTING ERROR:
- **Field-by-field**
- **Batched**

SERVER-SIDE VALIDATION WITH PHP

SERVER-SIDE VALIDATION
Mandatory Data
• Validating Strings
• Validating Zip and postcodes
• Validating email addresses
• Validating URLs
• Validating numbers
• Validating credit cards

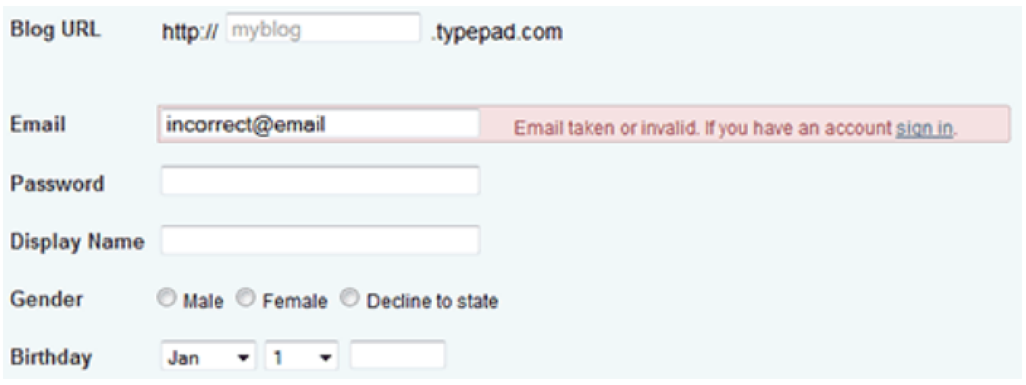
Validating Dates and Times
• Dates
• Times
• Logic, the date function, and MySQL

JAVASCRIPT AND CLIENTSIDE VALIDATION

Besides validation, there are many other common uses of JavaScript in web database applications including:

- Simple interaction with form data. For example, JavaScript is often used to calculate values and display these in an input widget.
- Enhancing user interactions by adding dynamic elements to a web page. Common features include pull-down menus, mouseover changes to the presentation (*rollovers*), and dialog boxes.
- Customizing the browser and using information from the browser to enhance presentation.

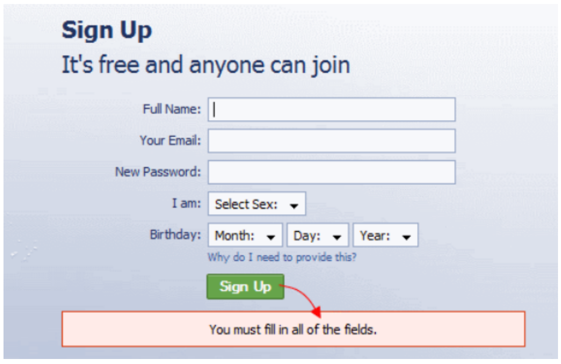
CLIENT-SIDE VALIDATION EXAMPLE
client-side validation, form never gets submitted if validation fails. Validation is being handled in JavaScript methods that you create (or within frameworks/plugins) and users get immediate feedback if validation fails.



Blog URL	http:// myblog .typepad.com	
Email	incorrect@email	Email taken or invalid. If you have an account sign in.
Password		
Display Name		
Gender	○ Male ○ Female ○ Decline to state	
Birthday	Jan ▾ 1 ▾	

WHAT TO VALIDATE

-REQUIRED INFORMATION


Sign Up

It's free and anyone can join

Full Name:

Your Email:

New Password:

I am: Select Sex:

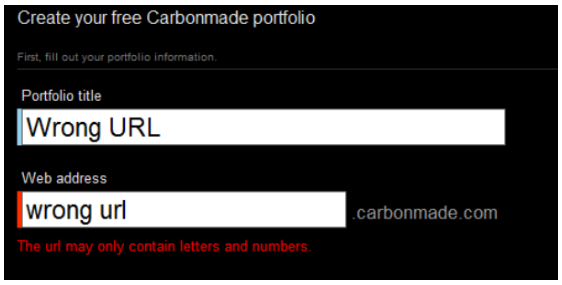
Birthday: Month: Day: Year:

Why do I need to provide this?

Sign Up

You must fill in all of the fields.

-CORRECT FORMAT


Create your free Carbonmade portfolio

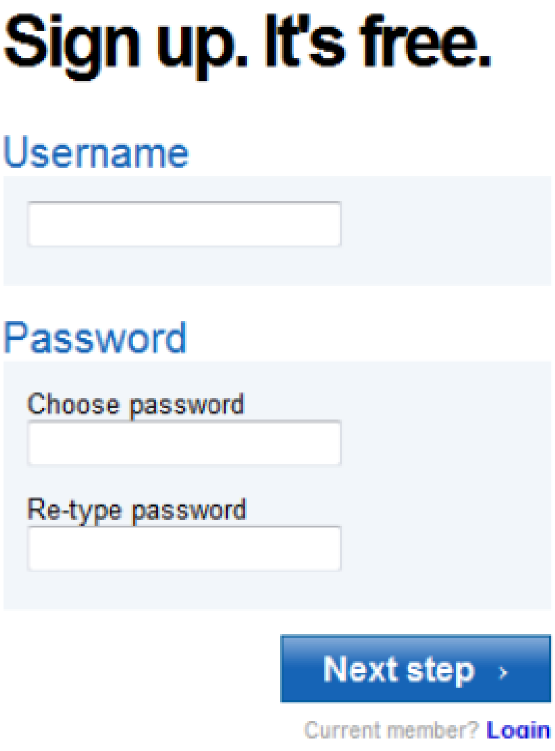
First, fill out your portfolio information.

Portfolio title: Wrong URL

Web address: wrong url .carbonmade.com

The url may only contain letters and numbers.

-CONFIRMATION FIELDS


Sign up. It's free.

Username

Password

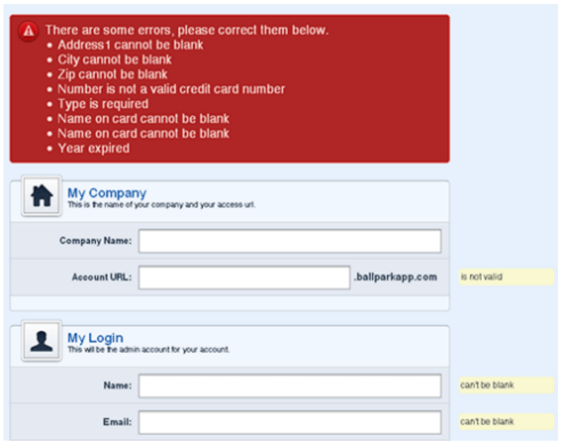
Choose password

Re-type password

Next step ›

Current member? Login

-VALIDATION UPON SUBMIT


There are some errors, please correct them below.

- Address1 cannot be blank
- City cannot be blank
- Zip cannot be blank
- Number is not a valid credit card number
- Type is required
- Name on card cannot be blank
- Name on card cannot be blank
- Year expired

My Company

This is the name of your company and your access url.

Company Name:

Account URL: .ballparkapp.com — is not valid

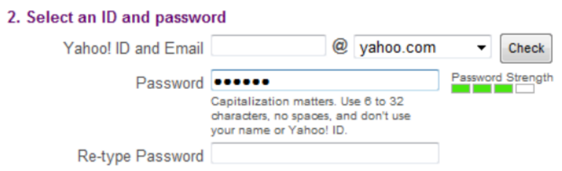
My Login

This will be the admin account for your account.

Name: — can't be blank

Email: — can't be blank

-REAL-TIME VALIDATION (OR INSTANT VALIDATION)


2. Select an ID and password

Yahoo! ID and Email @ yahoo.com Check

Password •••••• Password Strength

Capitalization matters. Use 6 to 32 characters, no spaces, and don't use your name or Yahoo! ID.

Re-type Password

SESSIONS

Session Management
Using Session in Validation
When to Using Sessions?
PHP Session API and Configuration

Concept :

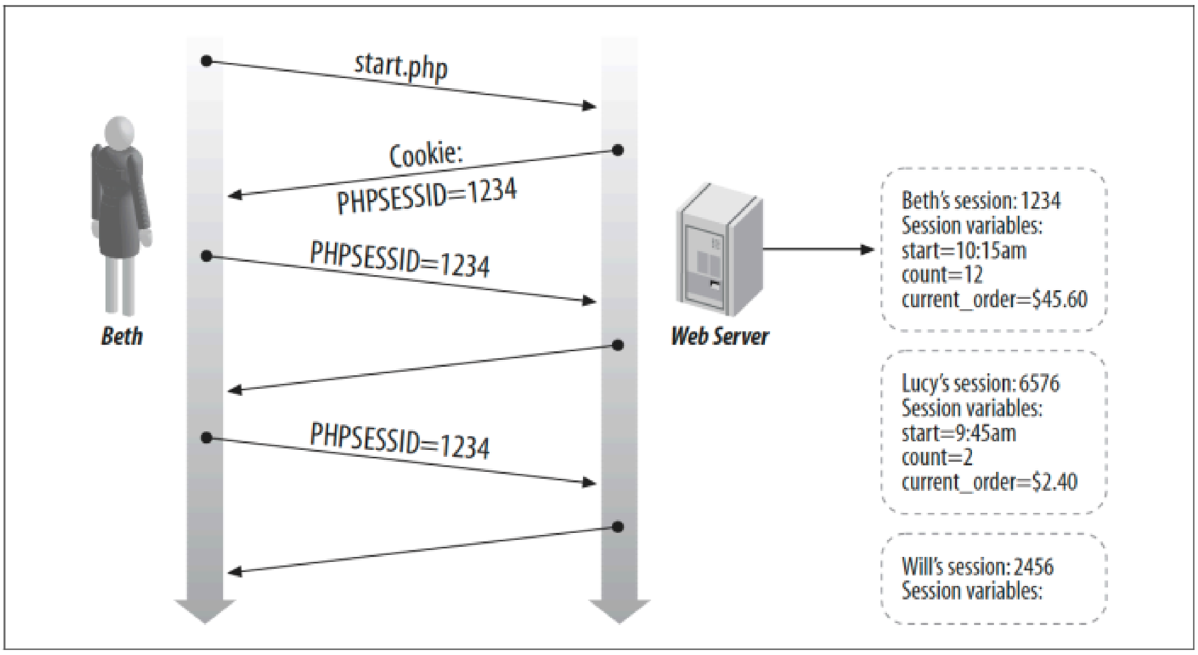


Figure 10-1. Session IDs and session variables

Web Database Applications with PHP and MySQL, Second Edition, 2004 by Hugh E. Williams and David Lane

A session manages the interaction between a web browser and a web server
• A session has two components: session variables and a session identifier (ID)
-The session variables are the state information that's related to a user's interaction with an application.
-The session variables are stored at the web server or database server, and are located using the session ID

• Three characteristics of session management over the Web
-Information or state must be stored. Information that must be maintained across multiple HTTP requests is stored in session variables.
-Each HTTP request must carry an identifier that allows the server to process the request with the correct session variables.
- Sessions need to have a timeout. Otherwise, if a user leaves the web site, there is no way the server can tell when the session should end

• Starting Session
-The session_start( ) function is used to create a new session.
-The first time a user requests a script that calls session_start( ), PHP generates a new session ID and creates an empty file to store session variables.

• Using Session Variable
-PHP provides access to session variables through the superglobal associative array $_SESSION.
-To unset a session variable, you use the unset( ) function

```
// This call either creates a new session or finds an existing one.
session_start();

// Check if the value for "count" exists in the session store
// If not, set a value for "count" and "start"
if (!isset($_SESSION["count"]))
{
  $_SESSION["count"] = 0;
  $_SESSION["start"] = time();
}
```

• Ending Session
-At some point in an application, sessions should be destroyed
-The first time a user requests a script that calls session_start( ), PHP generates a new session ID and creates an empty file to store session variables.
-a call to the session_destroy( ) function should be made to clean-up the session variables and remove the session file.
-Be aware that while a call to session_destroy( ) removes the session file from the system, it doesn't remove the session cookie from the browser.

```
// This call either creates a new session or finds an existing one.
session_start();

// Check if the value for "count" exists in the session store
// If not, set a value for "count" and "start"
if (!isset($_SESSION["count"]))
{
  $_SESSION["count"] = 0;
  $_SESSION["start"] = time();
}
```

CASE STUDY: USING SESSION IN VALIDATION
WHEN TO USING SESSION?

Judul: CASE STUDY: Using Session in Validation

Kapan Menggunakan Session?
Session biasanya digunakan dalam validasi PHP ketika Anda perlu menyimpan data yang berhubungan dengan pengguna selama sesi mereka berlangsung. Contoh situasi yang memerlukan penggunaan session meliputi:

Validasi Login: Ketika seorang pengguna mencoba masuk, Anda dapat menggunakan session untuk menyimpan data seperti ID pengguna atau nama pengguna agar bisa diakses di halaman-halaman selanjutnya.

Form Validation: Saat pengguna mengirimkan formulir dengan data yang mungkin perlu di-validasi atau diperiksa, Anda dapat menggunakan session untuk menyimpan pesan kesalahan atau data yang telah dimasukkan sehingga pengguna dapat melihatnya pada halaman berikutnya setelah validasi.

Keranjang Belanja: Jika Anda menjalankan toko online, Anda dapat menggunakan session untuk menyimpan item yang dipilih oleh pengguna dalam keranjang belanja mereka selama sesi berlangsung.

Larangan Akses: Anda juga dapat menggunakan session untuk melarang akses ke bagian tertentu dari situs web hingga pengguna masuk atau memiliki izin khusus.

Contoh Kode Penggunaan Session dalam Validasi PHP:

php
Copy code
```
<?php
// Inisialisasi session
session_start();

// Contoh penggunaan session untuk validasi login
if(isset($_POST['login'])) {
   $username = $_POST['username'];
   $password = $_POST['password'];

   // Simulasikan validasi login
   if($username == 'pengguna' && $password == 'password') {
      $_SESSION['user_id'] = 123; // Simpan ID pengguna dalam session
      header('Location: welcome.php');
   } else {
      $_SESSION['error_message'] = 'Username atau password salah';
      header('Location: login.php');
   }
}
?>
```
Dalam contoh di atas, session digunakan untuk menyimpan ID pengguna dan pesan kesalahan. Jika login berhasil, ID pengguna disimpan dalam session, dan pengguna diarahkan ke halaman selamat datang. Jika ada kesalahan login, pesan kesalahan disimpan dalam session, dan pengguna diarahkan kembali ke halaman login dengan pesan kesalahan tersebut.

Anda dapat mengakses dan mengelola data session menggunakan variabel global $_SESSION di seluruh halaman web PHP yang terkait dengan sesi yang sama.