موضوع جلسه :فرم یادداشت و اعتبارسنجی سمت سرور

---

1. ◆ساخت فرم برای ایجاد یادداشت(note_create.view.php)

در این فایل یک فرم HTML ساخته میشه که شامل یک textareaبرای وارد کردن متن یادداشت هست:

```
<form method="POST" action="/notes/create">
    <textarea
        id="body"
        name="body"
        placeholder="note text"
        rows="3"
        required
    ><?= isset($_POST['body']) ? $_POST['body'] : ''
?></textarea>

    <?php if(isset($error['body'])): ?>
        <p style="color:red"><?= $error['body'] ?></p>
    <?php endif; ?>

    <button type="submit">Submit</button>
</form>
```

◆نکته:

- از requiredاستفاده کرده تا سمت **کاربر** جلوی ارسال خالی گرفته بشه.
- مقدار قبلی textarea هم با استفاده از isset($_POST['body'])نگه داشته میشه تا اگه اروری اومد، اطلاعات پاک نشن.

---

2. ◆اعتبارسنجی سمت سرور(note_create.php)

اگرچه سمت کلاینت (required) کار می‌کنه، اما اعتبارسنجی اصلی و امن باید سمت سرور انجام بشه:

```
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $error = [];

    if (strlen($_POST['body']) === 0) {
        $error['body'] = 'Body is required';
    }

    if (strlen($_POST['body']) > 1000) {
        $error['body'] = 'Body is too long';
    }

    if (empty($error)) {
        $config = require 'config.php';
        $db = new Database($config['database']);

        $db->query(
            "INSERT INTO notes (body, user_id) VALUES
(:body, :user_id)",
            ['body' => $_POST['body'], 'user_id' => 3]
        );

        header("Location: /notes");
        exit;
    }
}
```

---

3. تست کردن با curl

جفری برای تست درخواست POST از ترمینال استفاده کرد:

```
curl -X POST http://localhost/laracast-php/notes/create -d
"body=cmd" -d "user_id=3"
```

🔹 **توضیح:**

- `-X POST` → مشخص می‌کنه درخواست از نوع POST هست.
- `-d` → داده‌ای که باید ارسال بشه (مثل `$_POST['body']`).
- این مثل اینه که فرم را پر کنیم و دکمه ارسال را بزنیم.

---

4. 🔒 چرا باید اعتبارسنجی سمت سرور باشه؟

چون ممکنه کاربر `required`را حذف کنه یا با curl و ابزارهای دیگه فرم جعلی ارسال کنه.
پس باید همیشه روی `$_POST`در PHP اعتبارسنجی کنیم.

---

5. ⚠نکته مهم درباره فضای بین تگ‌های `<textarea>`

اگر اینو بنویسی:

```
<textarea> </textarea>
```

این یک فاصله داخل مقدار textarea حساب میشه، و `strlen($_POST['body'])` برابر ۱ میشه، یعنی یادداشت خالی نیست! پس باید تگ‌ها کاملاً خالی باشن:

```
<textarea></textarea>
```

✅خلاصه

- `required`سمت کاربر خوبه، اما کافی نیست.
- اعتبارسنجی اصلی در PHP باید انجام بشه.
- خطاها در یک آرایه ذخیره می‌شن و اگر مشکلی نباشه، یادداشت در دیتابیس ذخیره می‌شه.
- مقدارهای قبلی فرم رو با `isset($_POST['body'])`دوباره توی فرم نمایش می‌دیم.
- حداکثر طول یادداشت هم کنترل میشه.
- `curl`ابزار قدرتمندی برای تست POST و GET هست.

توضیحات خودم :

ابتدا به صفحه note_create.php رفته و یک یادداشت خالی رو ذخیره کرد و گفت این درست نیست برای اینکه سمت کاربر این قضیه رو کنترل کنیم در تعریف textarea در انتهایش required میگذاریم.

```
<textarea id="body"
          name="body"
          placeholder="note text"
          rows="3"
          class="block w-full rounded-md bg-white px-3 py-1.5 text-base text-gray-900 outline-1 -outline-offset-1 outline-gray-300 placeholder:text-gray-400 focus:outline-2 focus:-outline-offset-2 focus:outline-indigo-600 sm:text-sm/6"
          required></textarea>
```

فقط بین <textarea></textarea> نباید فاصله باشه حتی اگه یک space هم باشه دیگه خالی محسوب نمیشه.

بعد به ترمینال رفته و از دستور curl استفاده کرد با استفاده از این دستور میتونیم یک درخواست post یا get به ادرس مشخص بدیم به شرط اینکه پروژه در حال اجرا باشه و سرور xampp روشن باشه پارامتر x–ادرسی که باید postبشه رو میدیم و در d–مقادیری که باید ارسال بشن.

curl -X POST http://localhost/laracast-php/notes/create -d "body=cmd" -d "user_id=3"

رو به cmd در فرم متن note_create.view.phpدستور بالا مثل این میمونه که در صفحه در فایل $_POST[‘body’] پرکردیم و دکمه ارسال رو زدیم و textareaعنوان یادداشت در شده هست.cmdبرابر با note_create.php

پس این دستور یک درخواست postبه آدرس مشخص میفرسته و در بدنه ان فیلد body هم برابر با cmd هست.درواقع یک خطی که با curl مینویسیم سمت سرور یعنی فایل note_create.php کدی شبیه این هست:

$body = $_POST['body'];

// حالا مثلا insert در دیتابیس:

$db->query("INSERT INTO notes (body) VALUES (:body)", ['body' => $body]);

برای مدیریت خطا در سمت سرور کد زیر را مینویسیم

```
if($_SERVER["REQUEST_METHOD"] == "POST"){
    $error =[];
    if(strlen($_POST["body"]) == 0){
       $error["body"] = "Body is required";
    }
    if(empty($error)){
       $db = new Database($config['database']);
       $db->query("INSERT INTO notes (body,user_id) VALUES (:body ,:user_id)" , ['body' => $_POST['body'], 'user_id' => 3 ]);
    }
{
```

بعد در فایل note_create.view.php در زیر جعبه متن کد زیررا مینویسیم:

```
<?php if(isset($error['body'])): ?>
                <p><?=$error['body'] ?></p>
            <?php endif; ?>
```

باید تعداد کاراکترهایی که کاربر یادداشت میکنه هم کنترل بشه مثلا بیشتر از ۱۰۰۰ نشه پس کد زیر مینویسیم:

```
if(strlen($_POST["body"]) > 1000){
       $error["body"] = "body is too long ";
    }
```

بعد بین textarea کد زیر را برای اینکه اگر فرم خطا داشت مقدار قبلی پاک نشه مینویسیم :

```
<?= isset($_POST['body']) ? $_POST['body'] : '' ; ?>
```

؟ سوال ۱:

در فرم HTML زیر، چرا از `<?= isset($_POST['body']) ? $_POST['body'] : '' ?>` **داخل** `<textarea>` **استفاده می‌کنیم؟**

```
<textarea name="body">
    <?= isset($_POST['body']) ? $_POST['body'] : '' ?>
</textarea>
```

الف) برای اینکه textarea **همیشه خالی باشه**
ب) برای اینکه اگر فرم خطا داشت، مقدار قبلی پاک نشه
ج) برای اینکه تگ textarea **معتبر باشه**
د) برای جلوگیری از XSS

✅جواب درست: گزینه ب — (این کد باعث می‌شه وقتی فرم با خطا ارسال بشه (مثلاً یادداشت خالی باشه)، مقدار قبلی داخل `textarea` **باقی بمونه و پاک نشه. این تجربه کاربری رو بهتر می‌کنه.**

بریم سراغ ؟ سوال ۲:

وقتی کاربر فرم یادداشت را با یک متن خالی ارسال کند، چه اتفاقی در سرور می‌افتد؟ (با توجه به این کد)

```
if(strlen($_POST["body"]) == 0){
    $error["body"] = "Body is required";
}
```

الف) یادداشت ذخیره می‌شود
ب) خطای PHP رخ می‌دهد

ج) پیغام خطا "Body is required" در صفحه نشان داده می‌شود
د) کد بالا باعث می‌شود فرم دوباره ارسال شود

✅عالیه !**کاملاً درسته**
جواب درست :**ج) پیغام خطا "Body is required" در صفحه نشان داده می‌شود**

این خط از کد:

```
$error["body"] = "Body is required";
```

باعث می‌شه متغیر `$error['body']` مقدار بگیره و چون در `view`چک می‌کنیم:

```
<?php if(isset($error['body'])): ?>
    <p><?= $error['body'] ?></p>
<?php endif; ?>
```

پس خطا در صفحه نمایش داده میشه.

حالا بریم سراغ **؟ سؤال ۳**:

چرا بین تگ `<textarea>`نباید حتی یک فاصله (space) خالی هم باشه؟

الف) باعث ارور در PHP میشه
ب) باعث میشه textarea ظاهر نشه
ج) اون space به عنوان مقدار اولیه داخل textarea نمایش داده میشه
د) تاثیری نداره، فقط زیبایی کد به هم میریزه

✅ جواب صحیح :**ج) اون space به عنوان مقدار اولیه داخل textarea نمایش داده میشه**

در HTML ، **هر چیزی که بین تگ‌های** `<textarea> ... </textarea>`**باشه**، به‌عنوان **مقدار اولیه** داخل فیلد نمایش داده میشه even—یک فاصله‌ی خالی! مثلاً:

```
<textarea> </textarea>
```

نتیجه: داخل textarea یه) space فاصله) هست.
اما اینو می‌خوایم:

```
<textarea><?= $_POST['body'] ?? '' ?></textarea>
```

تا اگر مقداری باشه، نمایش بده. اگر نه، هیچی نباشه.

بریم سراغ ? سؤال ۴:

در این خط از کد:

```
$db->query("insert into notes (body,user_id) values
(:body,:user_id)", [
    'body' => $_POST['body'],
    'user_id' => 3
]);
```

چه کاری انجام میشه؟

الف) فقط کوئری آماده میشه ولی اجرا نمیشه
ب) کوئری insert اجرا میشه و اطلاعات وارد دیتابیس میشه
ج) کوئری update اجرا میشه
د) کوئری select اجرا میشه

پاسخ ب) کوئری insert اجرا میشه و اطلاعات وارد دیتابیس میشه ✓

توضیح:

```
$db->query("insert into notes (body,user_id) values
(:body,:user_id)", [
    'body' => $_POST['body'],
    'user_id' => 3
```

```
]);
```

در اینجا:

- `$db->query(...)` هم کوئری رو آماده می‌کنه(prepare) ،
- و هم اون رو با مقدارهای داده‌شده `['body'=>..., 'user_id'=>...]` اجرا می‌کنه .(execute)

بریم سراغ **؟ سؤال ۵**:

در این کد:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // انجام عملیات
}
```

چرا از `$_SERVER['REQUEST_METHOD']` استفاده می‌کنیم؟

الف) چون بررسی کنیم آیا فرم با روش GET ارسال شده
ب) چون بررسی کنیم آیا فرم با روش POST ارسال شده
ج) چون می‌خواهیم مسیر فایل اجرا شده رو چک کنیم
د) برای اتصال به دیتابیس استفاده میشه

جواب درست **ب) چون بررسی کنیم آیا فرم با روش POST ارسال شده ✓**

توضیح:
در PHP ، وقتی می‌خوایم بررسی کنیم که آیا فرم ارسال‌شده با روش POST بوده یا نه، از این شرط استفاده می‌کنیم:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // یعنی فرم با متد POST ارسال شده
}
```

این باعث میشه فقط وقتی که کاربر دکمه «ارسال» رو زده و فرم POST شده، کد اجرا بشه.
این یعنی امنیت و ساختار بهتر کد.

بریم سراغ **؟ سؤال ۶:**

کد زیر چه کاری انجام می‌دهد؟

```
echo htmlspecialchars($_POST['name']);
```

الف) مقدار name را مستقیماً چاپ می‌کند
ب) مقدار name را رمزنگاری می‌کند
ج) مقدار name را چاپ می‌کند ولی هر کدی داخلش را به‌عنوان متن در نظر می‌گیرد
د) مقدار name را از دیتابیس می‌گیرد و چاپ می‌کند

عالی، جواب درست بود ✅ ج(
چون `htmlspecialchars()` جلوی اجرای کدهای HTML یا JavaScript مخرب رو می‌گیره و اون‌ها رو به‌صورت متن عادی نمایش می‌ده) برای جلوگیری از XSS.

حالا همون‌طور که خواستی، بریم سراغ ۱۰ سوال تستی پشت سر هم 👇

؟ سؤال ۷

کدام دستور یک جدول جدید به نام `users` در پایگاه داده ایجاد می‌کند؟

الف) `select * from users`
ب) `insert into users values (...)`
ج) `create table users (...)`
د) `drop table users`

؟ سؤال ۸

در PDO ، استفاده از `prepare()` و `execute()` به چه منظوری است؟

الف) اجرای مستقیم کوئری
ب) جلوگیری از SQL Injection
ج) حذف رکوردها
د) گرفتن خروجی به‌صورت JSON

؟ سؤال ۹

چه تفاوتی بین `$_GET` و `$_POST` وجود دارد؟

الف `$_GET` (اطلاعات را به‌صورت رمزنگاری‌شده ارسال می‌کند
ب `$_POST` (از URL استفاده می‌کند
ج `$_GET` (از URL استفاده می‌کند و `$_POST` اطلاعات را در بدنه درخواست می‌فرستد
د) هیچ تفاوتی ندارند

؟ سؤال ۱۰

عبارت زیر در کجا استفاده می‌شود؟

```
<input type="hidden" name="id" value="<?= $note['id']
?>">
```

الف) برای نمایش مقدار به کاربر
ب) برای ارسال داده‌ای پنهان در فرم
ج) برای جلوگیری از XSS
د) برای آپلود فایل

؟ سؤال ۱۱

تابع require_onceدر PHP چه کاری می‌کند؟

الف) یک فایل را چند بار اجرا می‌کند
ب) یک فایل را فقط یک بار وارد می‌کند
ج) فایل را فقط در صورت نیاز وارد می‌کند
د) باعث حذف فایل می‌شود

؟ سؤال ۱۲

کدام خطا در صورت اشتباه در نام متد POST رخ می‌دهد؟

الف) ۴۰۴
ب) ۴۰۳
ج) هشدار Noticeیا Warning
د) ۵۰۰Internal Server Error

؟ سؤال ۱۳

کدام روش برای جلوگیری از نمایش خطاهای PHP در مرورگر استفاده می‌شود؟

الف) استفاده از ini_set("display_errors", "on")
ب) استفاده از error_reporting(E_ALL)
ج) استفاده از ini_set("display_errors", "off")
د) استفاده از var_dump()

؟ سؤال ۱۴

اگر مقدار ورودی کاربر را به‌طور مستقیم در کوئری SQL قرار دهیم، چه اتفاقی ممکن است بیفتد؟

الف) اجرای بهتر برنامه
ب) افزایش امنیت
ج) امکان حملهSQL Injection
د) کاهش عملکرد

؟ سؤال ۱۵

درPHP ، برای بررسی خالی نبودن یک مقدار از چه چیزی استفاده می‌شود؟

الف) `isset()`
ب) `empty()`
ج) `!empty()`
د) `$_SERVER`

؟ سؤال ۱۶

کدام یک از موارد زیر باعث اجرای دستورات فقط در صورت کلیک روی دکمه فرم می‌شود؟

الف) `if ($_SERVER['REQUEST_METHOD'] === 'GET')`
ب) `if ($_POST)`
ج) `if ($_SERVER['REQUEST_METHOD'] === 'POST')`
د) `if (isset($_SERVER))`

✓سؤال ۷

جواب تو :ج
درست بود!
`create table`برای ساخت جدول استفاده میشه.

❌سؤال ۸

جواب تو :الف
جواب درست :ب
`prepare()`و `execute()`در PDO برای جلوگیری از **SQL Injection**هستن.

✅سؤال ۹

جواب تو :ج
درست بود!
`$_GET`داده رو از طریق URL می‌فرسته، `$_POST`توی بدنه درخواست.

✅سؤال ۱۰

جواب تو :ب
درست بود!
`type="hidden"`داده رو توی فرم پنهان نگه می‌داره و برای ارسال پشت‌صحنه استفاده می‌شه.

✅سؤال ۱۱

جواب تو :ب
درست بود!
`require_once`**فقط یک بار فایل رو وارد می‌کنه حتی اگه چند بار صداش بزنی.**

✗سؤال ۱۲

جواب تو :الف
جواب درست :ج
اگه `$_POST['name']`**رو صدا بزنی ولی مقدار نداشته باشه، خطای Notice یا Warning دریافت می‌کنی، نه ۴۰۴.**

✓سؤال ۱۴

جواب تو :ج
درست بود!
اگر ورودی کاربر مستقیماً وارد کوئری SQL بشه، احتمال حمله SQL Injection هست.

✓سؤال ۱۵

جواب تو :ج
درست بود!
`empty($value)!`**بررسی می‌کنه که مقدار خالی نباشه.**

✓سؤال ۱۶

جواب تو :ج
درست بود!
`$_SERVER['REQUEST_METHOD'] === 'POST'` **بررسی می‌کنه فرم ارسال شده یا نه.**