

Backend Selektion=8;

Utbildare: Mahmud Al Hakim

NACKADEMIN

Lektionstillfällets mål

- **API:er del 2 av 2**
- Utveckla en egen tjänsteserver och backendtestning

Kort summering av föregående lektion

- Vi har gått igenom grunderna i API:er
- RESTful API:er
- JSON
- Arbeta mot befintliga servrar

API – Application Programming Interface



Utveckla en egen tjänsteserver

- Videobutiken* behöver hjälp med att utveckla ett eget API.
- Ägaren vill dela information om butikens filmer.
- Målet är att samarbeta med ett antal godisbutiker runt om i Sverige för att kunna sälja butikens filmer via deras system.
- Vi ska nu skapa ett RESTful API som levererar information om alla filmer i JSON-format.

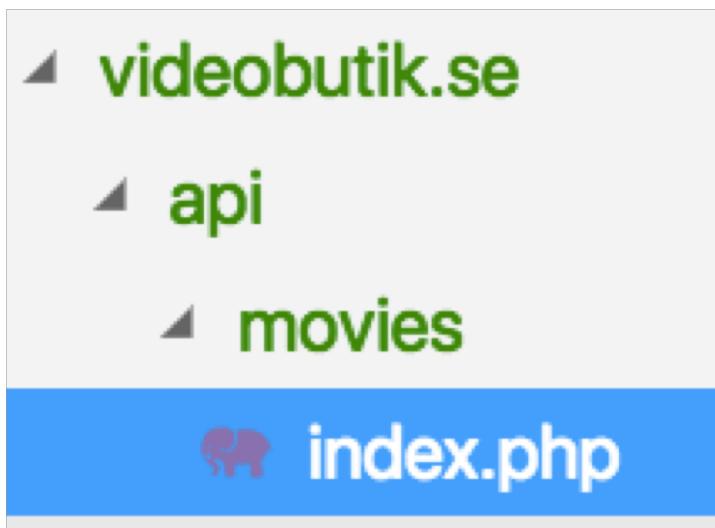
* Se lektion 6 för mer info om videobutiken.

Arbetsprocessen – Pseudokod

1. Ange innehållstypen JSON
2. Logga in i databasen
3. Skapa en SQL-sats för att hämta alla filmer
4. Kör SQL
5. Spara alla filmer i en array
6. Skapa en JSON-string med hjälp av funktionen `json_encode`
<http://php.net/manual/en/function.json-encode.php>
7. Skicka JSON till klienten

Skapa en lämplig endpoint

- Skapa en mapp som heter **api** i applikationens rot t.ex **videobutik.se**
- Skapa en undermapp som beskriver resultatet t.ex. **movies**.
- Skapa filen **index.php** i denna mapp.
- Vår endpoint blir **videobutik.se/api/movies**



Steg 1

```
<?php
```

```
// 1. Ange innehållstypen JSON
```

```
header("Content-Type: application/json; charset=UTF-8");
```

Steg 2 – Logga in i databasen

```
$servername = "localhost";
$username   = "root";
$password   = "root";
$database   = "filmdb";

try{
    $db = new PDO("mysql:host=$servername;dbname=$database;charset=utf8",
                  $username, $password);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // echo "Connected successfully";
}

catch(PDOException $e){
    echo $e-> getMessage();
}
```

Steg 3

// 3. Skapa en SQL-sats för att hämta alla filmer

```
$sql = "SELECT * FROM film";
```

Steg 4

// 4. Kör SQL

```
$stmt = $db->prepare($sql);  
$stmt->execute();
```

Steg 5

// 5. Spara alla filmer i en array

```
$array = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Steg 6

```
// 6. Skapa en JSON-string med hjälp av funktionen json_encode  
// http://php.net/manual/en/function.json-encode.php  
// json_encode – Returns the JSON representation of a value  
  
$json = json_encode($array);
```

Steg 7

// 7. Skicka JSON till klienten

```
echo $json;
```

// OBS! Ingen extra info får skickas till klienten

```
// t.ex. echo "<pre>";
```

Testa ditt nya API med Postman

<https://www.getpostman.com/downloads>

The screenshot shows the Postman application interface. At the top, there are tabs for 'Runner' and 'Create'. The main header includes 'My Workspace' with a dropdown, 'Invite', and various status icons like a refresh circle, a bell, and a heart. On the right, there's an 'Upgrade' button.

In the center, there are two requests listed:

- A green 'Untitled Request' with a 'GET' method and the URL 'https://webacademy.se/videobutiken/api/movies/'.
- The second request, also a green 'GET https://webacademy.se/videobutiken/api/movies/' entry, is highlighted with a red border.

Below the requests, the URL 'https://webacademy.se/videobutiken/api/movies/' is displayed again. The request details show it's a 'GET' request to 'https://webacademy.se/videobutiken/api/movies/'. There are 'Send' and 'Save' buttons.

The 'Params' tab is selected in the request details. It has a table with columns: KEY, VALUE, and DESCRIPTION. One row is present with 'Key' in the KEY column and 'Value' in the VALUE column, both under the DESCRIPTION column.

At the bottom, the 'Body' tab is selected. It shows the response body in JSON format:

```
1 [ { 2   "filmid": "1", 3   "titel": "Braveheart", 4   "kategori": "Action", 5   "huvudroll": "Mel Gibson", 6   "pris": "20" 7 }, 8 ]
```

Other tabs at the bottom include 'Cookies', 'Headers (6)', and 'Test Results'. The status bar at the bottom right shows 'Status: 200 OK', 'Time: 282 ms', and 'Size: 4.91 KB'.

Övning

- Skapa följande endpoint **videobutik.se/api/users**
- Skapa ett API som skickar en lista över alla kunder i JSON-format.

En app som använder vår tjänsteserver

- En godisbutik ska nu testa vårt API.
- Godisbutiken behöver visa en prislista över alla filmer.
- Godisbutiken lägger till en administrationsavgift på priset.
- Avgiften är 5 kr per film.
- Hjälp godisbutiken att skapa listan.
- Visa en ID, titel och pris (inkl. avgiften).

Arbetsprocessen

1. Ange en endpoint
2. Hämta data i JSON-format
3. Konvertera JSON till en PHP-Array
4. Skapa en tabell

Steg 1

// 1. Ange en endpoint

```
$url="http://webacademy.se/videobutiken/api/movies";
```

Steg 2

// 2. Hämta data i JSON-format

```
$json = file_get_contents($url);

if(empty($json)){
    echo "<h2>Problem med att hämta data!</h2>";
    exit;
}
```

Steg 3

// 3. Konvertera JSON till en PHP-Array

```
$array = json_decode($json, true);
```

```
// Testa arrayen. Vid fel kontrollera JSON-strängen
if(!is_array($array)){
    echo "<h2>Problem med att skapa en array!</h2>";
    exit;
}
```

Steg 4

```
// 4. Skapa en tabell
$table = "<table class='table table-hover table-sm'>";
$table .= '<tr class="thead-dark" style="cursor: pointer">
            <th>ID</th><th>Titel</th><th>Pris</th></tr>';
foreach ($array as $key => $row) {
    $id = $row['filmid'];
    $titel = $row['titel'];
    $pris = $row['pris'] + 5;

    $table .= "<tr><td>$id</td><td>$titel</td>
                <td>$pris kr</td></tr>";
}
$table .= "</table>";
```

Övning

- Använd API:et som du skapat under föregående övning
videobutik.se/api/users
- API:et skickar en lista över alla kunder i JSON-format.
- Skapa ett skript som visar kunderna i en HTML-tabell.

Summering av dagens lektion

- Vi har jobbat med att utveckla en egen tjänsteserver.
- Reflektioner kring dagens lektion?

NACKADEMIN

Framåtblick inför nästa lektion

Under nästa lektion kommer vi att jobba med praktiska övningar.

NACKADEMIN