Creating API Endpoints for Specific Data

In the previous tutorial, you created an API that returned every record from the `security_events` table.

In this tutorial, you will create multiple API endpoints that return different types of data.

This approach is commonly used in modern web applications.

Why Create Multiple Endpoints?

Your current API returns all records:

```
api_events.php
```

This works, but many applications need smaller, more focused datasets.

For example:

```
api_events.php
api_critical_events.php
api_statistics.php
```

Each endpoint has a specific purpose.

Create a Critical Events API

Create:

```
api_critical_events.php
```

Add:

```
<?php

$conn = new mysqli(
    "localhost",
    "root",
    "",
    "project_db"
);

header(
    "Content-Type: application/json"
);

$query = "
    SELECT *
    FROM security_events
    WHERE severity = 'critical'
";

$result =
    $conn->query($query);

$events = [];

while (
    $row =
    $result->fetch_assoc()
) {

    $events[] = $row;

}

echo json_encode(
    $events,
    JSON_PRETTY_PRINT
);
```

```
?>
```

Test the Endpoint

Open:

```
http://localhost/api-demo/api_critical_events.php
```

You should only see critical events.

> **Screenshot Placeholder**
>
> Insert screenshot showing critical event results.

Create a Statistics API

Create:

```
api_statistics.php
```

Add:

```
<?php

$conn = new mysqli(
    "localhost",
    "root",
    "",
    "project_db"
);

header(
    "Content-Type: application/json"
);
```

```
$statistics = [

    "total_events" => 0,
    "critical_events" => 0,
    "high_events" => 0

];

$result =
    $conn->query(
        "SELECT COUNT(*) AS total
         FROM security_events"
    );

$statistics["total_events"] =
    $result->fetch_assoc()["total"];

$result =
    $conn->query(
        "SELECT COUNT(*) AS total
         FROM security_events
         WHERE severity='critical'"
    );

$statistics["critical_events"] =
    $result->fetch_assoc()["total"];

$result =
    $conn->query(
        "SELECT COUNT(*) AS total
         FROM security_events
         WHERE severity='high'"
    );

$statistics["high_events"] =
    $result->fetch_assoc()["total"];

echo json_encode(
    $statistics,
    JSON_PRETTY_PRINT
```

```
);

?>
```

Test the Statistics Endpoint

Open:

```
http://localhost/api-demo/api_statistics.php
```

Example:

```
{
    "total_events": 3,
    "critical_events": 1,
    "high_events": 1
}
```

Read the Statistics API with JavaScript

Create:

```
dashboard.html
```

Add:

```
<!DOCTYPE html>
<html>
<head>
    <title>Dashboard</title>
</head>
<body>

<h1>Security Dashboard</h1>
```

```
<div id="output"></div>

<script>

fetch("api_statistics.php")
    .then(response => response.json())
    .then(data => {

        document.getElementById(
            "output"
        ).innerHTML = `

            <p>
                Total Events:
                ${data.total_events}
            </p>

            <p>
                Critical Events:
                ${data.critical_events}
            </p>

            <p>
                High Events:
                ${data.high_events}
            </p>

        `;

    });

</script>

</body>
</html>
```

Common Endpoint Examples

Many real systems use endpoints such as:

```
/api/users
/api/products
/api/orders
/api/messages
/api/statistics
```

Each endpoint returns only the data required by the application.

Summary

You now have:

```
api_events.php
```

Returns all events.

```
api_critical_events.php
```

Returns only critical events.

```
api_statistics.php
```

Returns dashboard statistics.

This approach keeps APIs organised and makes applications easier to maintain.

Next tutorial: **Adding Parameters to an API Using URL Variables**.

Revision #1
Created 2026-06-10 02:02:28 UTC by Mr Napper
Updated 2026-06-10 02:02:36 UTC by Mr Napper