Importing JSON into MySQL with PHP

This tutorial imports a fictional JSON dataset into MySQL using PHP, PDO, validation and a prepared statement.

Use fictional data only. Do not import real student, staff, account, device or school-security information.

Project files

Create this structure inside your XAMPP project:

```
project/
├── includes/
│   └── database.php
├── smart_security_data.json
└── import_json.php
```

`includes/database.php` should create the shared `$pdo` connection used throughout the project. Keep connection details in that one file rather than repeating them on every page.

Understand the expected data

The fictional JSON file contains homes, devices and nested events. Each event will become one row in `security_events`.

Before importing, identify the fields your code expects. A typical event needs:

- a device identifier and type;
- a room;
- an event timestamp and type;
- the data transmitted;
- severity; and

- the access result.

Read and decode the JSON

```
<?php
declare(strict_types=1);

require __DIR__ . '/includes/database.php';

$jsonPath = __DIR__ . '/smart_security_data.json';

if (!is_readable($jsonPath)) {
    exit('The fictional JSON file could not be read.');
}

try {
    $data = json_decode(
        file_get_contents($jsonPath),
        true,
        512,
        JSON_THROW_ON_ERROR
    );
} catch (JsonException $error) {
    exit('The JSON structure is invalid.');
}
```

`JSON_THROW_ON_ERROR` prevents malformed JSON from being treated as valid data.

Prepare one reusable insert

```
$insert = $pdo->prepare(
    'INSERT INTO security_events (
        device_id,
        device_type,
        room,
        event_timestamp,
        event_type,
```

```
        data_transmitted,
        severity,
        access_result
    ) VALUES (
        :device_id,
        :device_type,
        :room,
        :event_timestamp,
        :event_type,
        :data_transmitted,
        :severity,
        :access_result
    )'
);
```

Preparing the statement once separates the SQL instructions from each row's values.

Validate and import each event

```
$allowedSeverities = ['low', 'medium', 'high', 'critical'];
$allowedResults = ['authorised', 'denied', 'blocked'];
$inserted = 0;
$rejected = 0;

$pdo->beginTransaction();

try {
    foreach (($data['devices'] ?? []) as $device) {
        foreach (($device['events'] ?? []) as $event) {
            $deviceId = trim((string) ($device['deviceId'] ?? ''));
            $deviceType = trim((string) ($device['deviceType'] ?? ''));
            $room = trim((string) ($device['room'] ?? ''));
            $timestamp = trim((string) ($event['timestamp'] ?? ''));
            $eventType = trim((string) ($event['eventType'] ?? ''));
            $transmitted = trim((string) ($event['dataTransmitted'] ?? ''));
            $severity = strtolower(trim((string) ($event['severity'] ?? '')));
            $accessResult = strtolower(trim((string) ($event['accessResult'] ?? '')));
```

```
            $validTimestamp = DateTime::createFromFormat(
                'Y-m-d H:i:s',
                $timestamp
            ) !== false;

            if (
                $deviceId === '' ||
                $deviceType === '' ||
                $room === '' ||
                $eventType === '' ||
                !$validTimestamp ||
                !in_array($severity, $allowedSeverities, true) ||
                !in_array($accessResult, $allowedResults, true)
            ) {
                $rejected++;
                continue;
            }

            $insert->execute([
                'device_id' => $deviceId,
                'device_type' => $deviceType,
                'room' => $room,
                'event_timestamp' => $timestamp,
                'event_type' => $eventType,
                'data_transmitted' => $transmitted,
                'severity' => $severity,
                'access_result' => $accessResult
            ]);

            $inserted++;
        }
    }

    $pdo->commit();
} catch (Throwable $error) {
    if ($pdo->inTransaction()) {
        $pdo->rollBack();
    }

    exit('The import could not be completed.');
```

```
}
```

The transaction prevents a database error from leaving an uncertain partial import. Row validation rejects unsuitable values before storage.

Report the result

```
echo '<p>' . $inserted . ' fictional events imported.</p>';
echo '<p>' . $rejected . ' rows rejected.</p>';
```

For a classroom prototype, report useful totals without exposing database errors or connection details to the user.

Check the import

Confirm that:

- valid events are stored once;
- invalid values are rejected;
- the prepared statement is reused;
- a failed transaction is rolled back;
- the result totals match the database; and
- only fictional data was processed.

Revision #3
Created 2026-06-10 01:57:29 UTC by Mr Napper
Updated 2026-08-19 23:03:22 UTC by Mr Napper