

INSTALLATION OF SLIM

```
Select C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.21996.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vin>cd..

C:\Users>cd..

C:\>cd xampp

C:\xampp>cd htdocs

C:\xampp\htdocs>cd api

C:\xampp\htdocs\api>cd src

C:\xampp\htdocs\api\src>composer require slim/slim:3.*
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been created
Running composer update slim/slim
Loading composer repositories with package information
Updating dependencies
Lock file operations: 5 installs, 0 updates, 0 removals
  - Locking nikic/fast-route (v1.3.0)
  - Locking pimple/pimple (v3.5.0)
  - Locking psr/container (1.1.2)
  - Locking psr/http-message (1.1)
  - Locking slim/slim (3.12.5)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 5 installs, 0 updates, 0 removals
  - Downloading psr/http-message (1.1)
  - Downloading psr/container (1.1.2)
  - Downloading pimple/pimple (v3.5.0)
  - Downloading nikic/fast-route (v1.3.0)
  - Downloading slim/slim (3.12.5)
  - Installing psr/http-message (1.1): Extracting archive
  - Installing psr/container (1.1.2): Extracting archive
  - Installing pimple/pimple (v3.5.0): Extracting archive
  - Installing nikic/fast-route (v1.3.0): Extracting archive
  - Installing slim/slim (3.12.5): Extracting archive
Generating autoload files
1 package you are using is looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.

C:\xampp\htdocs\api\src>
```

localhost/api/public/getName/John/Wick

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'Explore'. A search bar and 'Sign In'/'Create Account' buttons are on the right. A blue banner at the top states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.' The left sidebar shows a 'History' panel with a list of requests, including the current one: 'GET 127.0.0.1/api/public/getName/John/Wick'. The main panel displays the request details for the GET method. The URL is '127.0.0.1/api/public/getName/John/Wick'. The 'Params' tab is active, showing a table with two columns: 'Key' and 'Value'. The 'Body' tab is also visible, showing a response in 'Pretty' format: '1 Hello, John Wick'. The status bar at the bottom indicates '200 OK 28 ms 271 B'.

GET 127.0.0.1/api/public/getName/John/Wick

127.0.0.1/api/public/getName/John/Wick

Params

Key	Value
Key	Value

Body

1 Hello, John Wick

200 OK 28 ms 271 B

JSON POSTNAME PAYLOAD

localhost/api/public/postName

The screenshot shows the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'Explore'. A search bar and 'Sign In'/'Create Account' buttons are on the right. A blue banner at the top states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.' The left sidebar shows a 'History' panel with a list of requests, including the current one: 'POST 127.0.0.1/api/public/postName'. The main panel displays the request details for the POST method. The URL is '127.0.0.1/api/public/postName'. The 'Body' tab is active, showing a JSON payload: '{ "lname": "hortizuela", "fname": "manny" }'. The status bar at the bottom indicates '200 OK 55 ms 287 B'. The response is shown in 'Pretty' format: '1 { "status": "success", "data": null }'.

POST 127.0.0.1/api/public/postName

127.0.0.1/api/public/postName

Body

```
{
  "lname": "hortizuela",
  "fname": "manny"
}
```

Body

1 { "status": "success", "data": null }

200 OK 55 ms 287 B

RESPONSE PAYLOAD

localhost/api/public/printName

The screenshot shows the Postman application interface. On the left, the 'History' panel lists several requests. The main workspace displays a POST request to '127.0.0.1/api/public/postPrint'. The 'Body' tab is selected, showing a JSON payload:

```
{  "lname": "hortizuela",  "fname": "manny"}
```

. Below the request, the 'Response' tab shows a successful status (200 OK) and a JSON response:

```
{  "status": "success",  "data": [    {      "fname": "manny",      "lname": "hortizuela"    }  ]}
```

. The status bar at the bottom indicates '200 OK 33 ms 323 B'.

The screenshot shows the SQLyog Community interface. The 'New Connection' panel on the left shows a connection to 'root@localhost' for the 'demo' database. The main workspace displays a query result table for the 'names' table. The table has columns 'id', 'lname', and 'fname'. The result shows one row with 'id' 1, 'lname' 'hortizuela', and 'fname' 'manny'. The status bar at the bottom indicates '1 row(s) Ln 1, Col 1'.

id	lname	fname
1	hortizuela	manny

UPDATE PAYLOAD

localhost/api/public/postUpdate

The Postman interface shows a POST request to `127.0.0.1/api/public/postUpdate`. The request body is a JSON object: `{ "id": "1", "lname": "wick", "fname": "john" }`. The response is `{ "status": "updated successfully", "data": null }`.

History:

- POST 127.0.0.1/api/public/postUpdate
- POST 127.0.0.1/api/public/postName
- GET 127.0.0.1/api/public/getName/Bula/Lord
- POST 127.0.0.1/api/public/getName/Bula/Lord
- POST 127.0.0.1/api/public/getName/Bula/Lord
- POST 127.0.0.1/api/public/postDelete
- POST 127.0.0.1/api/public/postUpdate
- POST 127.0.0.1/api/public/postUpdate
- POST 127.0.0.1/api/public/postPrint
- POST 127.0.0.1/api/public/postName
- POST 127.0.0.1/api/public/postName
- POST 127.0.0.1/api/public/postName
- POST 127.0.0.1/api/public/postName
- GET 127.0.0.1/api/public/getName/John/Wick

The SQLyog Community interface shows a query result for the `names` table. The result is a single row with the following data:

id	lname	fname
1	Wick	John

Database: demo Table: names

1 row(s) Connections: 1

DELETE PAYLOAD

localhost/api/public/postDelete

The image shows the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'Explore'. A search bar is present. A notification banner states: 'You are using the Lightweight API Client, sign in or create an account to work with collections, environments and unlock all free features in Postman.' The left sidebar shows a 'History' list of requests. The main workspace displays a 'POST' request to '127.0.0.1/api/public/postDelete'. The 'Body' tab is selected, showing a JSON payload:

```
{  "id": "1"}
```

. The 'Send' button is visible. Below the request, the 'Body' tab shows the response:

```
{  "status": "deleted successfully",  "data": null}
```

. The status bar at the bottom indicates '200 OK', '45 ms', and '300 B'.

The image shows the SQLyog Community interface. The title bar reads 'SQLyog Community 64 - [New Connection/demo - root@localhost*]'. The menu bar includes 'File', 'Edit', 'Favorites', 'Database', 'Table', 'Others', 'Tools', 'Powertools', 'Transactions', 'Window', and 'Help'. The toolbar contains various icons for database operations. The left sidebar shows a 'New Connection' dialog with a tree view of the database structure. The main workspace displays a 'Query 2' editor. Below the editor, the 'Table Data' tab is selected, showing a table named 'names' with columns 'id', 'lname', and 'fname'. The table structure is:

id	lname	fname
(Auto)	(NULL)	(NULL)

. The status bar at the bottom indicates 'Ready', '0 row(s)', 'Connections: 1', and an 'Upgrade to SQLyog Ultimate' link.