**What is redis**

Redis is essentially a very fast, in-memory database used to store and retrieve data quickly. It’s perfect when you need quick access to small pieces of information, like caching results from a slow database or maintaining session data for users in web applications.

**How Redis Works (in Simple Terms):**

1. **In-Memory Storage**: Redis stores all the data in the server’s RAM, making data retrieval extremely fast. However, this means the data size is limited by available memory.
2. **Key-Value Pairs**: Think of Redis as a large dictionary where you have keys (unique identifiers) and values (the data). You can store, update, or delete data using these keys.
3. **Persistence**: Redis can save data to the disk to ensure that it’s not lost if the server is restarted, but it’s mainly used for fast, temporary storage.

**Simple Redis Commands:**

* SET mykey "Hello": Store a key called "mykey" with the value "Hello".
* GET mykey: Get the value of the key "mykey" (would return "Hello").
* DEL mykey: Delete the key "mykey".

**Redis Project: Simple Todo List Application**

To help you understand Redis, we’ll build a simple **Todo List** application using **Node.js** and Redis as the database. This project will allow you to store and retrieve tasks.

**Project Structure:**

* **app.js**: The main file where the server runs.
* **index.html**: Frontend to display the Todo list.

**Step-by-Step Guide:**

**1. Setup Redis**

* Install Redis on your local machine (or use Docker).

bash

Copy code

sudo apt-get install redis

* Start Redis server:

bash

Copy code

redis-server

**2. Install Node.js and Redis Client**

* Install Node.js if you haven’t already.
* Create a new project folder and install the Redis client for Node.js.

bash

Copy code

npm init -y

npm install express redis body-parser

**3. Create app.js**

javascript

Copy code

const express = require('express');

const bodyParser = require('body-parser');

const redis = require('redis');

// Create Express App

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

// Create a Redis client

const client = redis.createClient();

// Connect to Redis

client.on('connect', function() {

console.log('Connected to Redis');

});

// Home route to display Todo list

app.get('/', (req, res) => {

// Get all tasks from Redis

client.lrange('tasks', 0, -1, (err, tasks) => {

if (err) throw err;

res.send(`

<h1>Todo List</h1>

<form action="/add" method="POST">

<input type="text" name="task" placeholder="Add a new task">

<button type="submit">Add Task</button>

</form>

<ul>

${tasks.map(task => `<li>${task}</li>`).join('')}

</ul>

`);

});

});

// Route to add a new task

app.post('/add', (req, res) => {

const newTask = req.body.task;

// Add the task to Redis list

client.rpush('tasks', newTask, (err, reply) => {

if (err) throw err;

console.log('Task added:', newTask);

res.redirect('/');

});

});

// Start the server

app.listen(3000, () => {

console.log('Server running on http://localhost:3000');

});

**4. Create index.html**

No need to create this separately, as we’re directly sending HTML in the response from the server in the / route.

**5. Run the Application**

* Make sure your Redis server is running.
* Run the Node.js app:

bash

Copy code

node app.js

* Open a browser and navigate to http://localhost:3000. You should see a simple interface to add tasks to your todo list.

**How It Works:**

1. When you add a new task, it’s stored in Redis as a list.
2. The client.rpush command adds the new task to the end of the list.
3. When the home page is loaded, client.lrange fetches all tasks from the list and displays them.

**Conclusion:**

This simple project demonstrates how Redis can be used to store data and retrieve it efficiently. The Redis commands rpush and lrange are used to add tasks to a list and get tasks from it. The idea can be extended to other use cases like session storage, caching, etc., sir.