9. REST

- > REST : Representational State Transfer
- sest is an architectural style that defines a set of constraints to be used for creating web services.

22 mg > Hallet mape \$

-> CRUD operations

POST submits new data to the server PUT updates existing data partially DELETE removes data

Which has some posts and we will use crub operations on this page.

* Creating RESTful APIS

GET /posts index to get data for all posts

POST /posts create to add new post

GET /posts /: id view to get one post (wing id)

PATCH /posts /: id update to update specific post

DELETE / posts /: id destroy to delete specific post

	In demo/ REST_CLASS Teaminal
	HP@DP MINGWEA ~/Desktop/demo/REST_CLASS \$ npm init -y :
	\$ npm install express:
_	\$ npm install ejs
	\$ touch index.js \$ code index.js
	In demo/ REST CLASS / index.js
	const express = require ("express"); const app = express();
24720	const post = 8080;
1204	app. listen (port, () =) i console log(App is listening on port (siport) 3);

classaute.

In Terminal

- s nodemon index.js:

App is listening on post 8080

* Implement: GET /posts :

GET / posts to get data for all posts (index soute)

In demo/ REST_CLASS/ index.js:

- let posts = [

username: "parth",:

content: "I love coding":

3, 1...3, 1...3];

app. use (express. unlencoded ({ extended: true })); app. use (express. static ("public"));

app. set ("view engine", "ejs") / hut syrand)

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

3); Res. render ("index.ejs", { posts });

In demo/REST_CLASS/public / style.css background-color: aqua; - body & h1 { color: maroon; } margin: 0; steel the item;

padding: 0.5 rem 1 rem;

. user {

font-style: italie;

color: blue;
} background-color: wheat;
margin: 1 sem 0; · post } In demo/ REST_CLASS/ views/ index.ejs - (!DOCTYPE html) Klink sel="stylesheet" haef="/style.css".) 41 head > (() 1209 1, "Franch 1") 30 1 m 32, 28A

zizog) rzog immensigneri * (body> < h1> Quora Posts </ h1> (% for (const post of posts) { % <div class="post"> <h3 class = "user"> @ <%= post username %>> </h3> <h4 \(\text{class} = "content" \(\text{} \)</pre> <0/0 = post. content %> K" and </hd> <1div7 (%) } 1/07 4 body > 11 html> the tuent 2 Now A Short A SYLDOG 1 > solites and and solites (bush) X Khool >

Line & " Start = Lookery & "story" = Marine (mark)

* Implement: POST /posts POST / posts to add new post (create (new) route) CALE 3 (15/20 15 3/20 12/10) PUT 0/5 2 soutes: - serve the form GET /posts/new -> Add new post post /posts CALL SALES TO SALES TO SALES THE SAL (3/1/2) In views/index.ejs - Calate a new post 1107 (o) o) o) C That D In views/ new ejs S looted IV - < ! DOCTYPE html> < title > create a New Post </title> </head> < body? < form action="/posts" method="post"> < Imput type = "text" name = "usegname" placeholder = "enter username" >

(627 his latery) File singulary & (69) (textarea name="content" placeholder="enter your content"> (Itextarea) (6A) < 6827 ("Small with & <- button type="submit" > submit Post <1 button> (Iform> (1body > KILLMIZ In REST_CLASS / index. js app. get ("/posts/ new", (reg, res) =) { res. render ("new.ejs"); app. post("/posts", (seq, ses) =) } let { username, content } = req. body; posts. push ({username, content }); Res. redirect ("/posts");
Coult-in get request of express

```
* Implement: GET /posts/:id
   GET /posts/:id to get one post (using id)
              ( show soute)
         In views/index.ejs
    (div class="post")
< a half="http://localhost:8080/poets/</= post.id xy">
        show Details
       41a>
     </div >
                                     < ybody
                                     Slavent
        In views/ show.ejs
 - <! DOCTYPE html)
      < title > Post in Detail <1 title>
      < link sel="stylesheet" heef ="/style.css">
   11 head >
   < body ?
    < h27 Here is your post in Details <1 h27
    > Post id: <-10= post. id -11-7 <1p>
   : whole you say I had not say a find
    (bay 1 - 21209
```

* Create id for post -> When we create new post, we are not setting-up new id, so for that we will write(de) as below: -> ULID Package : Universally Unique Identifier In Terminal - \$ npm install unid < [mi] 12 In index is - const { v4: unidv4 } = require ("unid"); - Replace value of id of array items with unidvac) (reates sandom alphanumeric string with chauds (abach-imla-abc-xxx...) - app. post("/posts", (reg, res) =) { let { username, content } = req. body; posts. push ({ id: unidvac), username, content }); ses . sedirect ("/posts"); 3);

(thous)

* Implement: PATCH / posts/:id PATCH /posts/:id to update specific post (update noute) 2 soutes: - serve update form GET /posts/:id/edit → Update post PATCH /posts/:id 1 may land method - overside In views/ index.ejs - < div class = " post " 7 < a hsef = "http://localhost-2020/post/</-> Edit <1a> (head) 4 div 7 Ch2 > white your post cities app. get ("/posts/:id/edit", (seq, ses) => { services (apply) let { 1d 3 = 289, params; let post = posts. find (citem) => id === item.id); nes nender ("editijs", { post}); 3);

* method - overside · package - In HTML form element has only two method built-in get & post, so if we want to use PATCH, PUT, DELETE, ... ; we have to install a package. In Terminal - \$ npm install method-overside In views/edit.ejs - (! DOCTYPE. html > <title> Edit Post <1title> 1/head> 4/body> < h2> Edit Your Post <1h2> <P> Post id : <10 = post.id 1.7 2/p > LP> Post username: </- = post. id 1.7 </p> 16 1 xus (993) < form method = " post" action = " / posts / c/ = post id 1/> ? - method= PATCH"> <textanea name="content"> 1/textarea> <button type="submit" > Submit </button> 11body7 11 html7

araise In Lindexijs i tresmasignis - const method Overside = sequise ("method-overside"); app. use (methodOverside ("-method")); app. patch ("/posts/:id", (seq, ses) => { let { id } = reg.params; let newContent = seq. body. content; one let post = posts.find ((item) =) id===item.id); post. content = newContent; Pue (do) copy Res Redirect ("/posts"); tud> 1 xobia wit app. lette ("/pate/:id", (see, see) = { (20x many , god = 1 bit to) Pusts = posts . / That ((item) = 1/1 == 1 trans . /); : ("zomo V) + salas 2,294

```
/posts/:id DELETE
                          * Implement:
DELETE /posts/:id to delete specific post
                                                                                                                                                             (destroy route)
                                         (Consider Continued of Continue
                                                  In views/index.egs
                      - < div class="post" > ....
              action = "/posts/<1.= post. id 1.>? - method = Decere"
                                                               < button > Delete Post < 16utton7
                                          </ form >
                                </div>
                                                                              In index. is
                  - app. delete ("/posts/:id", (req, res) => {
                                                    let { id } = req. params;
                                                  posts = posts. filter ((item) = id !== item. id);
                                                 res. redirect ("/posts");
                            4);
```