WEB TECHNOLOGIES

Express JS

Prof. Vinay Joshi and Dr. Sarasvathi V
Department of Computer Science and Engineering.

Acknowledgement
The slides are created from various internet resources with valuable contributions from multiple professors

WEB TECHNOLOGIES

Express JS

ROUTING

EXPRESSJS: ROUTING

Routing

- Determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
- How your application responds to different HTTP requests.

- Each route can have one or more handler functions, which are executed when the route is matched.
- Route definition takes the following structure:
- app.METHOD(PATH, HANDLER)
- Where:
- app is an instance of express.
- METHOD is an HTTP request method, in lowercase.
- PATH is a path on the server.
- HANDLER is the function executed when the route is matched.

EXPRESSJS: ROUTING

Routing

- Routing refers to the mechanism for serving the client the content it has asked for.
- It is the most important aspects of your website or web services.
- Routing in Express is simple, flexible, and robust.
- A route specification consists of
  - An HTTP method (GET, POST, etc.),
  - A path specification that matches the request URI,
  - And the route handler.

EXPRESSJS: ROUTING

Routing

- The handler is passed in a request object and a response object.

- The request object can be inspected to get the various details of the request, and

- The response object's methods can be used to send the response to the client

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Routing

- Create an application using its root-level exported function
- ***const app = express();*** Set up routes using this app.
- To set up a route, use a function to indicate which HTTP method; for e.g., to handle the GET,PUT, DELETE etc.,
- To this function, pass the pattern to match and a function to deal with the request if it does match.

```
const app = express();

app.get('/hello', (req, res) => {
  res.send('Hello World');
});
```

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Request Matching

- The request’s method is matched with the route method (e.g., the get function was called on app, indicating it should match only GET HTTP methods)

- The request URL with the path spec matches ('/hello’), then the handler is called.

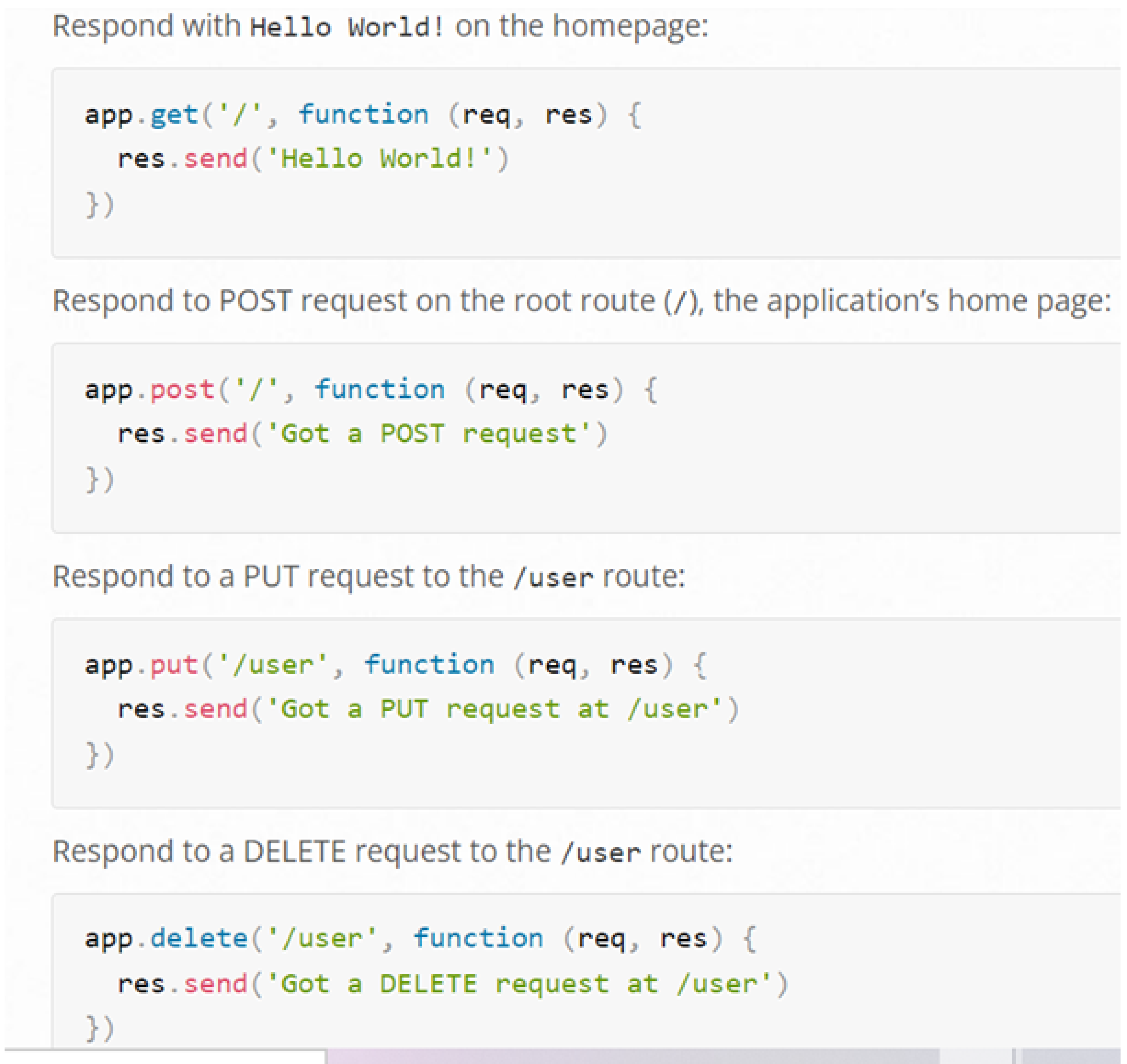
- In other words, “If you receive a GET request to the URL */hello*, then execute *this piece of code.”*

```
const app = express();

app.get('/hello', (req, res) => {
  res.send('Hello World');
});
```

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Routing



Respond with `Hello World!` on the homepage:

```
app.get('/', function (req, res) {
  res.send('Hello World!')
})
```

Respond to POST request on the root route (`/`), the application's home page:

```
app.post('/', function (req, res) {
  res.send('Got a POST request')
})
```

Respond to a PUT request to the `/user` route:

```
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user')
})
```

Respond to a DELETE request to the `/user` route:

```
app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user')
})
```

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Request Matching

- The method and the path need not be very specific. E.g., app.get(), app.post(), app.put(), etc.,

- If you want to say “any method,” you could use app.all().

- The path specification can also take regular expression-like patterns (like '/*.do') or regular expressions themselves.

- Regular expressions in paths are rarely used. Route *parameters in the path are used often.*

- They are named segments in the path specification that match a part of the URL.

```
const app = express();

app.get('/hello', (req, res) => {
  res.send('Hello World');
});
```

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Route Parameters

- If a match occurs, the value in that part of the URL is supplied as a variable in the request object.

```
app.get('/customers/:customerId', ...
```

- The customer ID will be captured and supplied to the handler function as part of the request in req.params, with the name of the parameter as the key.

- req.params.customerId can have any value for each of these URLs and can have multiple parameters.

- For e.g., /customers/:customerId/orders/:orderId, to match *a customer's order.*

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Route Lookup

- Multiple routes can be set up to match different URLs and patterns.

- The router tries to match all routes in the order in which they are installed.

- If two routes are possible matches to a request, it will use the first defined one.

- Can define routes in the order of priority.

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Route Lookup

When you add patterns its recommended to add more generic pattern *after the specific paths.*

- *For e.g, if you want to match everything that* goes under /api/, that is, a pattern like /api/*,

- Add this route only *after all the* specific routes that handle paths such as /api/issues.

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Handler Function

- Once a route is matched, the handler function is called.

- The parameters passed to the handler are a request object and a response object.

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Request Objects

1. To access a parameter value we use ***req.params.***
   req.param(name [, defaultValue])

EXPRESSJS: ROUTING

Request Objects

2*. **req.query***: This holds a parsed query string.

- It's an object with keys: as the query string parameters and
- Values as the query string values.
- Multiple keys with the same name are converted to arrays, and
- Keys with a square bracket notation result in nested objects

(e.g., order[status]=closed can be accessed as req.query.order.status).

```
{ key: value }
```

```
?key1=value1&key2=value2&key3=value3
```

EXPRESSJS: ROUTING

Request Objects

3. ***req.header, req.get(header)***:
   - The get method gives access to any header in the request.
   - The header property is an object with all headers stored as key-value pairs.

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Request Objects

4. ***req.path:***
   - The path for which the middleware function is invoked; can be any of:
     - A string representing a path.
     - A path pattern.
     - A regular expression pattern to match paths.
     - An array of combinations of any of the above.

EXPRESSJS: ROUTING

Request Objects

5. ***req.url, req.originalURL***:
   - Contain the complete URL, including the query string.
   - If any middleware modifies the request URL, originalURL retains the original URL as it was received, *before the modification.*

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Request Objects

6. ***req.body:***
   - Contains key-value pairs of data submitted in the request body.
   - By default, it is undefined, and is populated when you use body-parsing middleware is installed to read and optionally interpret or parse the body.

EXPRESSJS: ROUTING

Response Objects

The ***res*** object represents the HTTP response that an Express app sends when it gets an HTTP request.

```
res.send('<p>some html</p>')
```

1. ***res.send(body)***: Sends the HTTP response.
   - If the body is an object or an array, it is automatically converted to a JSON string with an appropriate content type.

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Response Objects

The ***res*** object represents the HTTP response that an Express app sends when it gets an HTTP request.

```
res.status(400).send('Bad Request')
```

2. ***res.status(code)***: This sets the response status code.
   - If not set, it is defaulted to 200 OK.
   - One common way of sending an error is by combining the status() and send() methods in a single call like res.status(403).send("Access Denied").

EXPRESSJS: ROUTING

Response Objects

3. ***res.json(object)***: Sends a JSON response
   - This method sends a response (with the correct content-type) that is the parameter converted to a JSON string using JSON.stringify().
   - The parameter can be any JSON type, including object, array, string, Boolean, number, or null

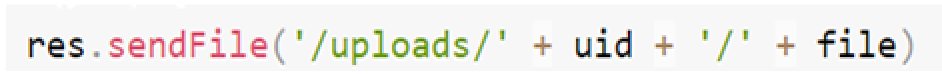
```
res.json({ user: 'tobi' })
```

WEB TECHNOLOGIES

EXPRESSJS: ROUTING

Response Objects

4. ***res.sendFile(path)***:
   - This responds with the contents of the file at path.
   - The content type of the response is guessed using the extension of the file.



```
res.sendFile('/uploads/' + uid + '/' + file)
```

WEB TECHNOLOGIES
EXPRESSJS: ROUTING

Request and Response Objects

for a complete list, please refer to the Request documentation of Express at
Request Objects: http://expressjs.com/en/api.html#req
Response Objects: http://expressjs.com/en/api.html#res

EXPRESSJS: ROUTING

Middleware

- An Express application is a series of middleware function calls.
- Router is nothing but a middleware function.
- Middleware functions have access to the request and response object (req,res), and the next middleware function in the application's request response cycle.
- The next middleware function is commonly denoted by a variable named ***next***.
- ***next*** is the only built-in middleware (other than the router) available as part of Express.

EXPRESSJS: ROUTING

Middleware

- Middleware can be at
  - The application level (i.e.,, applies to all requests) or
  - The router level (applies to specific request path patterns).

- The Middleware at the application level can be used like this: ***app.use(middlewareFunction);***

- The static middleware that knows the location of static files to serve.

EXPRESSJS: ROUTING

Middleware

- In order to use the same middleware in a route-specific way, you define as: ***app.use('/public', express.static('static'));***

- This would have mounted the static files on the path /public and all static files would have to be accessed with the prefix /public,

- For e.g.,: /public/index.html.

EXPRESSJS:URL BINDING

- URL binding in Express.js refers to the process of associating specific actions or functionalities in your web application with specific URLs (Uniform Resource Locators).

- It involves defining the behaviour or response that your server should provide when a client (such as a web browser) makes a request to a particular URL.

- URL binding allows for dynamic routing by using parameters in the URL, enabling the server to extract and process information from the URL. These parameters can be used to customize the response based on the client's request.

EXPRESSJS:URL BINDING

Static vs. Dynamic Routes

Static Routes:
Defined paths that do not change.
Example-: app.get('/about',…)

Dynamic Routes:
Paths with parameters that can change.
Ex-: app.get('/:id',….)

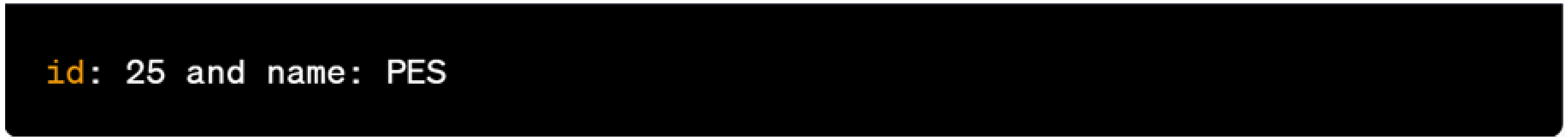
EXPRESSJS:URL BINDING

Dynamic Routes in Express.js

Code Example:

```
app.get('/things/:name/:id', function(req, res) {
  res.send('id: ' + req.params.id + ' and name: ' + req.params.name);
});
```

To test the above code, go to http://localhost:3000/things/PES/25



```
id: 25 and name: PES
```

THANK YOU

Vinay Joshi and Dr.Sarasvathi V
Department of Computer Science and Engineering

vinayj@pes.edu
sarsvathiv@pes.edu

Acknowledgement
The slides are created from various internet resources with valuable contributions from multiple professors