Express

Introduction

Lesson Plan

- What is express
- express architecture
- installing express
- express path
- req, res
- hello world
- middlewares
- Routers

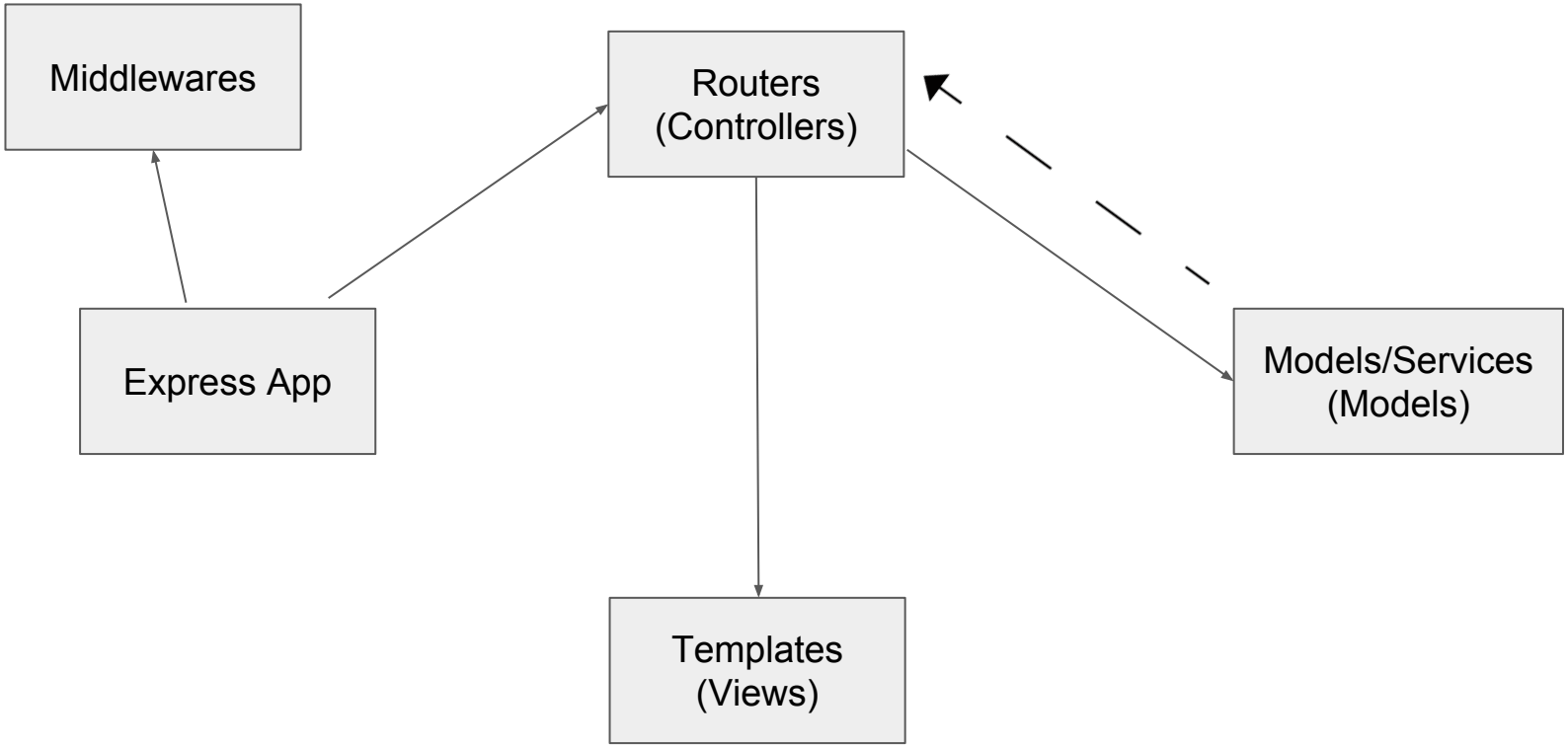
What is express

- framework for creating server applications
- minimalist
- unopinionated
- large community and a lot of packages you can connect to express

express architecture

- think of express as a chain made from chain blocks
- when a request is received by express it will go over the chain that is connected to that route and activate every chain block (chain block = middleware)
- you can add chain blocks to that chain
- you can say a chain blocks only works on certain urls, or on all of them.
- you can decide the methods (get, post, put, delete...) that the chain block will work on.
- From every chain block you can either pass to the next block or end the chain
- Since express is un opinionated there is no official architecture but in this course we will recommend on the following MVC structure.

Express Architecture



Express app

- represents an express application
- used to configure the routes of the application
- used to configure settings for the applications
- we can attach middlewares to the express app
- we can configure to start listening on a port
- lets install express create an express app object and start listening on a port

Middlewares

- middlewares can work on all or part of the routes, and on all or part of the methods
- they add functionality and pass the control to the next middleware or return results
- there is a lot of community middlewares you can attach to your app
- the middlewares are run in the same order they are attached
- We will demonstrate the middlewares later on

Routes

- With routes you can attach a listener to a path
- can be more than one listener
- listener can be attached to a method type like **get, post …** or to all methods with **all**
- For the path you can enter the following
  - string
  - string patterns ?, +, *, and ()
  - regular expressions
- you can define a routing param with :param which will be valid in **req.params**
- you can chain methods together by using the **route**
- you can define a router to separate routes to different apps

Routes EX 1

- lets start by creating an hello world application
- we will connect a route to our express application and send hello world on every route

req object

- the request object represents the incoming request
- **req.body** - key value pairs of data submitted in request body (you will need a middleware for this to work)
- **req.params** contains url params defined in route
- **req.query -** key value of the query params passed

res object

- represents http response that express will send back
- **res.json() -** sends a json response
- **res.status(404) -** sets the status code of the response
- **res.redirect(route)**
- **res.render(view, data)**
- **res.send(body)**
- **res.sendFile**

Routes EX2

- we will get a param with the url and print it in the response

routes ex3

- given a same route we need to attach a post and a get request to that route

routes ex4

- seperate the route to a different file using **Router**
- create a model that queries the todo rest server, and returns the data from there
- create a get method to display all the todo items

body-parser

- used to populate the **req.body** with key value of the data the user sent
- we connect this middleware to the express app.
- need to install it with npm

express.static

- used to serve static files
- you give this middleware the absolute path of the assets directory

EX install middlewares

- install the static middleware and the body-parser middleware

EX writing our own middleware

- lets add a middleware that adds an hello world message to the request

Templates

- A template will later transform to an HTML
- the template syntax depends on the templating engine used
- express is working with the popular template engines
- we can pass variables when rendering the template
- In this course we will use Pug (Jade) templating engine (recommended by express)
- you can set express application configuration using **app.set**
- two config option regarding templates
  - **views**
  - **view engine**

EX templates

- install pug template engine
- create a template for displaying hello world and a route

Student EX

- create a router with /login route
- there should be a post and get for that route.
- the get will render a template with a login form consisting of a mail
- the form will be send to the /login route
- if the mail is valid redirect the user to /welcome page
- else redirect him to an error page