COMP4021
Internet Computing

MVC in Express Part 1

Gibson Lam

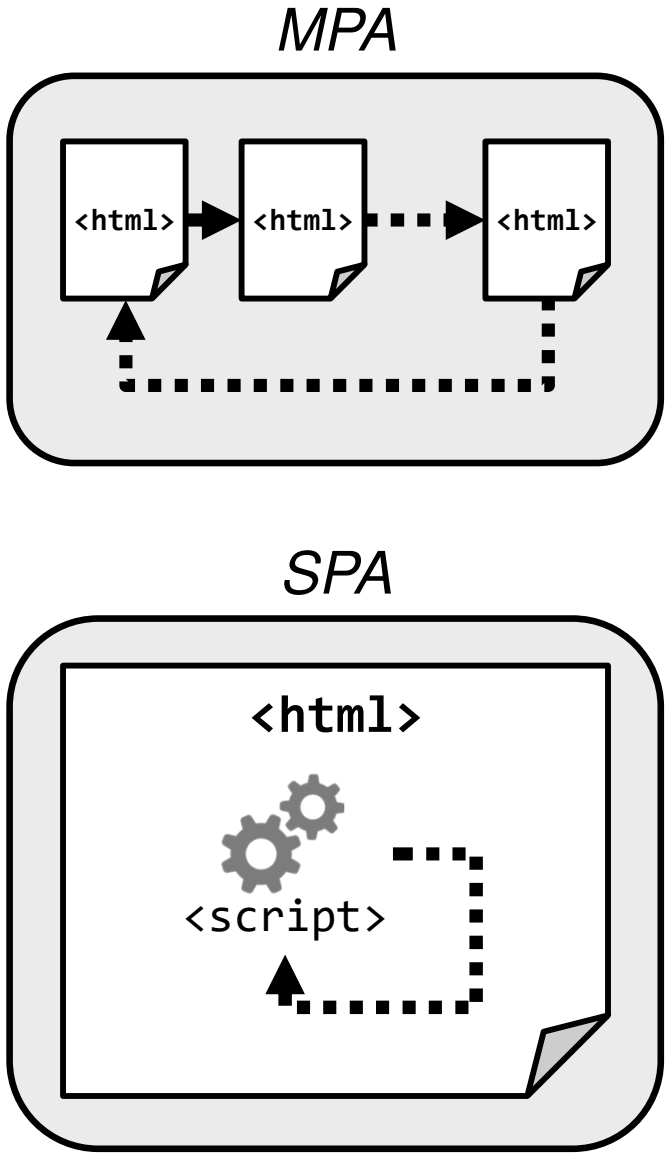
This Presentation

- You have so far focused on building SPAs (single page applications) using AJAX in the browser and Express in the server
- In this presentation, we will look at a software design pattern called Model-View-Controller (MVC)
- We will then use MVC to build a MPA (multi page application)

Single and Multi Page Applications

You have seen this before

- When building a web application using the basic approach, it is called a multi-page application (MPA)
- If the dynamic approach is used, it will become a single-page application (SPA)

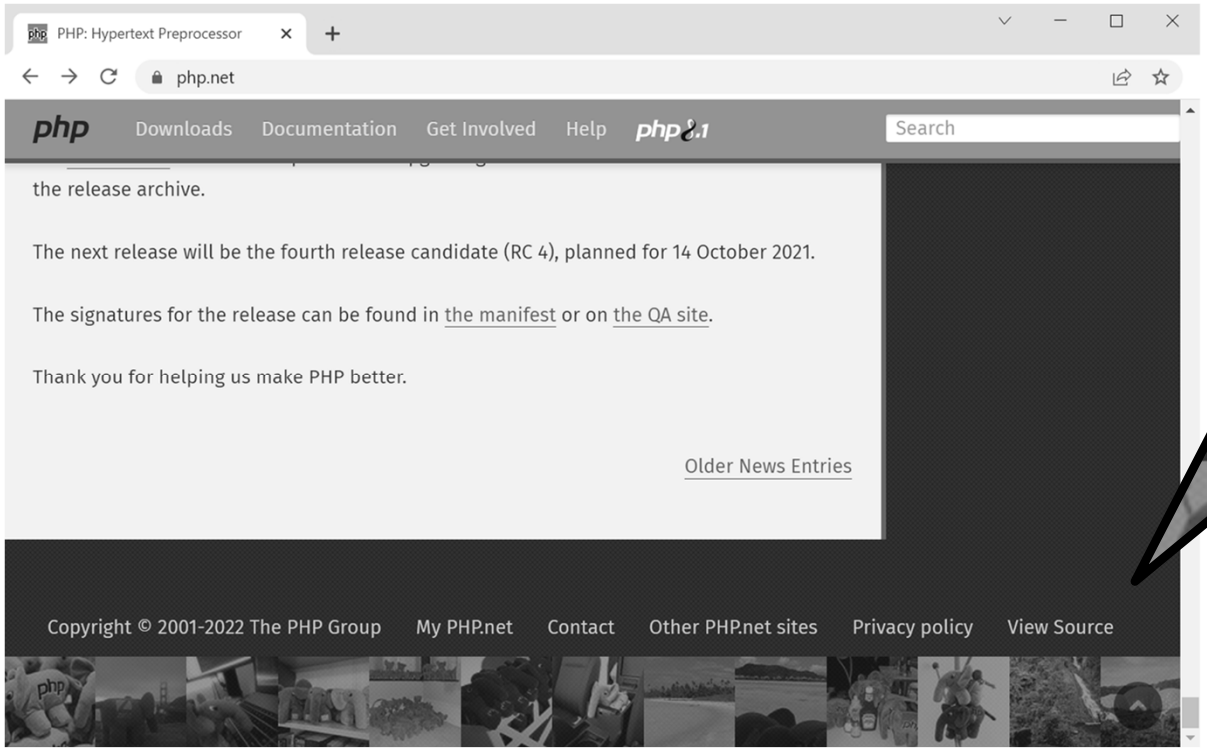


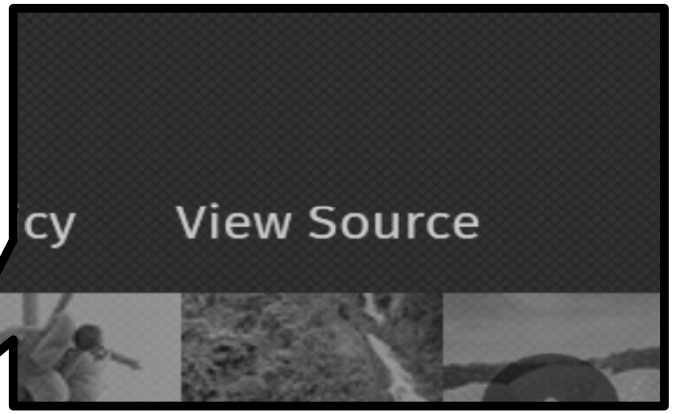
Building MPAs

- Web pages in a MPA typically contain server-side instructions
- It can be done by embedding server-side code inside web pages, e.g. using PHP
  - You can see an example on the PHP homepage, see next slide
- This allows quick development but things may become hard to manage when the application gets bigger and more complex

The PHP Homepage

- You can find a long PHP page example in the PHP homepage (https://www.php.net)
- Simply scroll to the bottom of the page and select ‘View Source’ (PHP source, not HTML!)

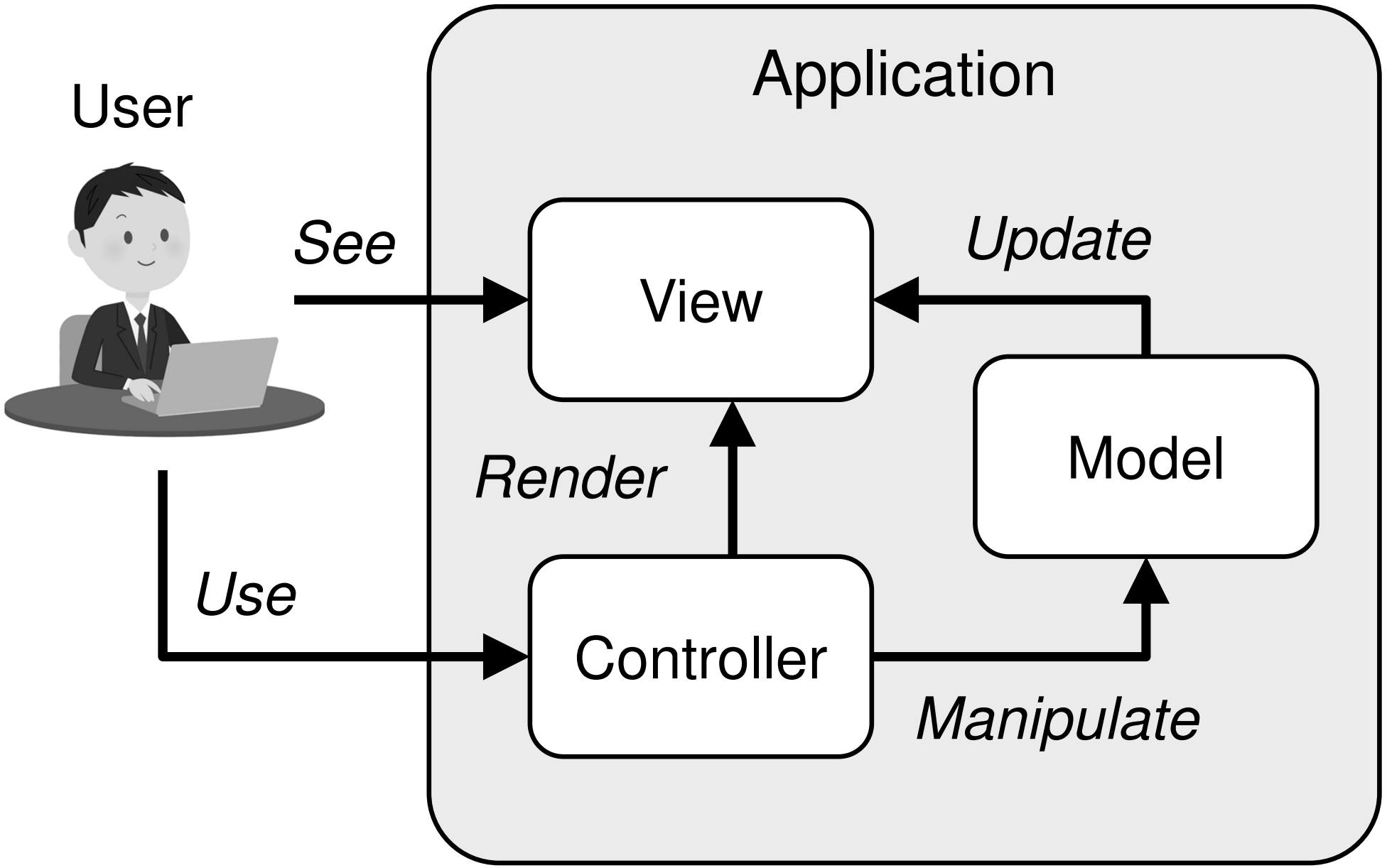




Model-View-Controller

- Another approach to build your MPA is by adopting some software design patterns
- One of the commonly used patterns is Model-View-Controller (MVC)
- It separates the handling of the following components in an application:
  - Model i.e. data
  - View i.e. visual components
  - Controller i.e. application logic

Overview of MVC

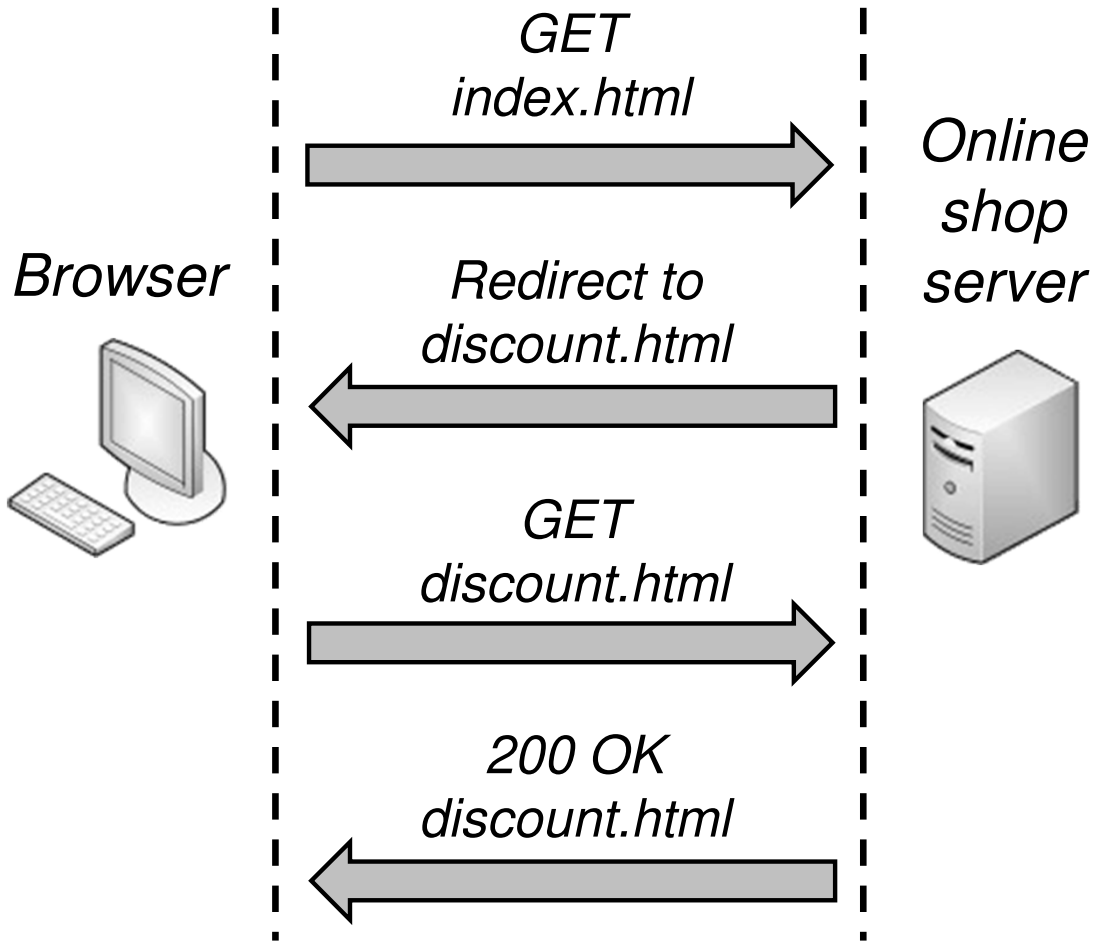


Using MVC in Express

- Although MVC can be used in any language/platform, we will focus on using MVC in Express and Node.js
- Before doing that, you need to learn these two things:
  1. Using HTTP redirect
  2. Using template engines

HTTP Redirect

- HTTP redirect is a method to ask the browser to load another page, rather than the requested page
- For example, an online shop can ask the browser to show the discount page, rather than the requested index page



The Redirect HTTP Response

- The redirection is sent in the HTTP response using the status code 302 and a `Location` header
- Here is an example:

```
HTTP/1.1 302 Found
Location: http://money.com
...More headers and content not shown...
```

The new location can be a relative path or a full URL

The Redirect Status Code

- The status code 302 represents a ‘temporary’ redirection
- There are also several other 3XX status codes representing other types of redirection
- For example:
  - Status code 301 tells the browser that the page has been permanently moved
  - Status code 304 tells the browser that the page is not modified and the browser can use its locally cached version of the page

Using Redirect in Express

- To use HTTP redirect in an Express server, you simply send a ‘redirect’ response (302) using `.redirect()`, like this:

```
app.get("/", (req, res) => {
    res.redirect("http://money.com");
});
```

- You can also redirect the browser to another path:

```
res.redirect("/mainpage");
```

Another Redirect Example

- It is very common to redirect a browser to the sign-in page when the user has not signed into the application:

```
app.get("/secret", (req, res) => {
    if (!req.session.username) {
        res.redirect("/signin");
        return;
    }

    ...Send the secret content...
});
```

The user has not signed in, secret content not shown

Using Dynamic Content

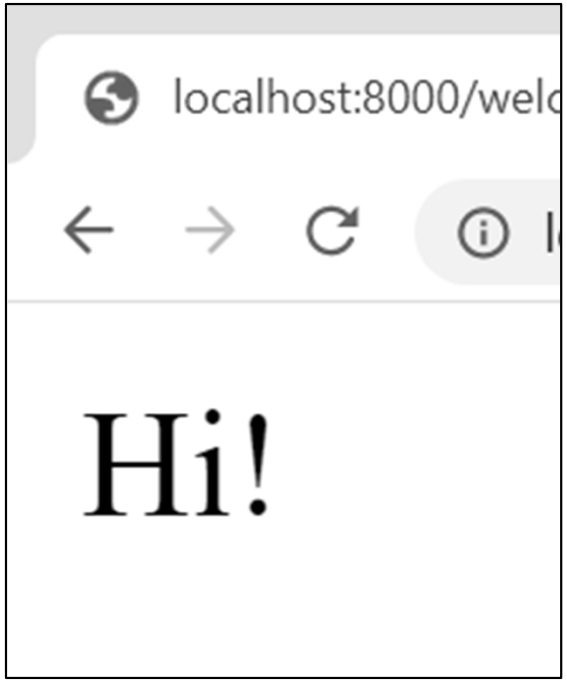
- When programming with an Express server, you have either:
  - Worked with static content (the `static` middleware), or
  - Used JSON (the `json` middleware)
- If you want to return dynamic web page content, similar to what PHP does, both of the above will not work

Using JavaScript Strings

- One simple way to return dynamic content is to build the HTML page on the fly, like this:

```
const welcomeMessage = "Hi!";

res.send("<html>" +
         "<body>" +
         welcomeMessage +
         "</body>" +
         "</html>");
```



Using Template Literals

- You may also use *template literals* (strings enclosed by a pair of backquotes ``) to save a bit of typing, i.e.:

```
const welcomeMessage = "Hi!";

res.send(`<html>
<body>${welcomeMessage}</body></html>`);
```

Anything inside `${ }` *is evaluated as part of the string content*

Template Engines

- If you want to mix code and HTML in the same file, just like what you do in PHP, you will need to use *template engines*
- There are many different template engines you can use in Express
- In the following discussion, we will use EJS because it is simple, and it works just like PHP

Initialization of EJS

- First, you need to install the EJS template engine using npm:

```
C:\Users\Gibson>npm install ejs
```

- You can then initialize EJS in an Express server using this code:

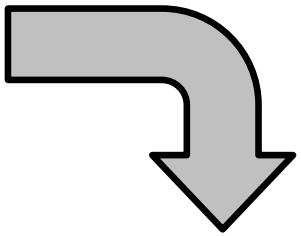
```
app.set("view engine", "ejs")
```

The EJS Language

- EJS works in a similar way to PHP
- You put JavaScript code inside the EJS tag: `<% ... %>`
- You can also use a shorthand to directly show JavaScript content, i.e. `<%= ... %>`
- However, you don't have an `echo` command or a `print` command to show output directly

A Simple EJS Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome!</title>
</head>
<body>
  <%= "Welcome to my page!" %>
</body>
</html>
```



Welcome to my page!

The EJS Views Folder

- After creating the EJS file content, you should save the file using the `.ejs` extension
- In EJS, `.ejs` files are called `views`
- The default folder that you should put your 'views' is in a folder called `views`, under the same folder of your Express server
- If you want to, you can also create sub-folders inside the `views` folder

Rendering an EJS View

- Let's assume you have prepared a view called `welcome.ejs`
- And you have put it inside the `views` folder
- Then, if you want to return the EJS content to the browser, you can 'render' the view using the following code:

```
res.render("welcome")
```

You do not need to use the `.ejs` *extension here*

Passing Variables to Views

- You can make things more interesting by passing variables to the views, such as the code below:

```
res.render("sayhi", { name: "Paul" })
```

- In the above code, a variable called `name` is passed to the view called `sayhi`
- Inside the view, you can then use the variable in the EJS code, which is shown on the next slide

Using Variables in EJS

- The following example shows the content of the name variable passed to it from the server:

```
<!DOCTYPE html>
<html>
<head>
  <title>Greeting!</title>
</head>
<body>
  Hi, <%= name %>!
</body>
</html>
```

sayhi.ejs

Using Views in MVC

- The EJS views form the ‘view’ of MVC
- In the next part of the discussion we will work with MVC’s model and controller
- And we will then build a simple MPA application