

Spring과 React

# Init

- <http://start.spring.io/> 에 접속하여 spring boot init project를 만듦.
- H2랑 Mongo는 강 쓸수도 있을거 같아서 넣어 놨음.

**SPRING INITIALIZR** bootstrap your application now

Generate a Maven Project ▾ with Java ▾ and Spring Boot 1.5.6 ▾

## Project Metadata

Artifact coordinates

Group

org.pangyo

Artifact

amicus

## Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Web ×

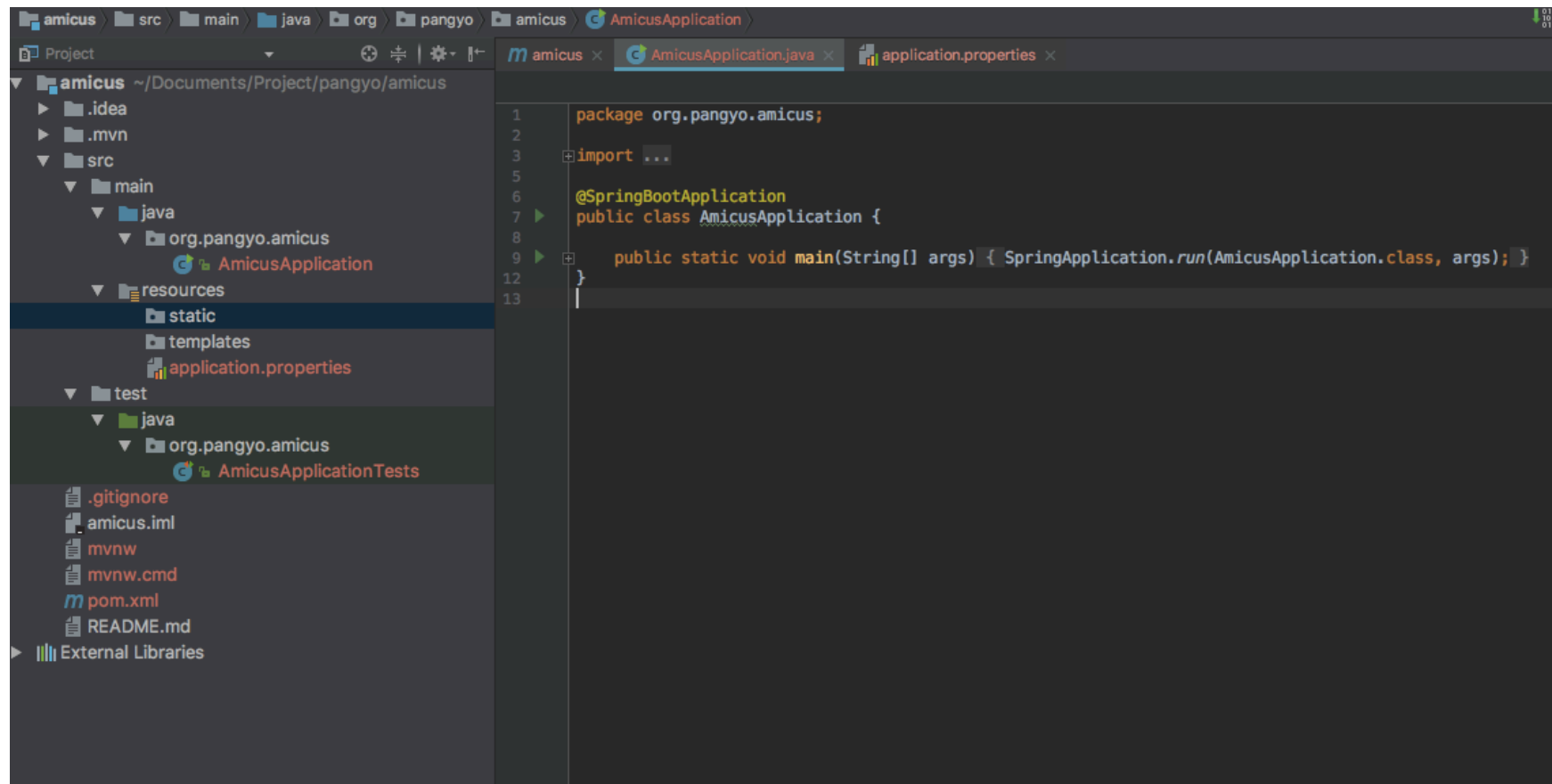
JPA ×

H2 ×

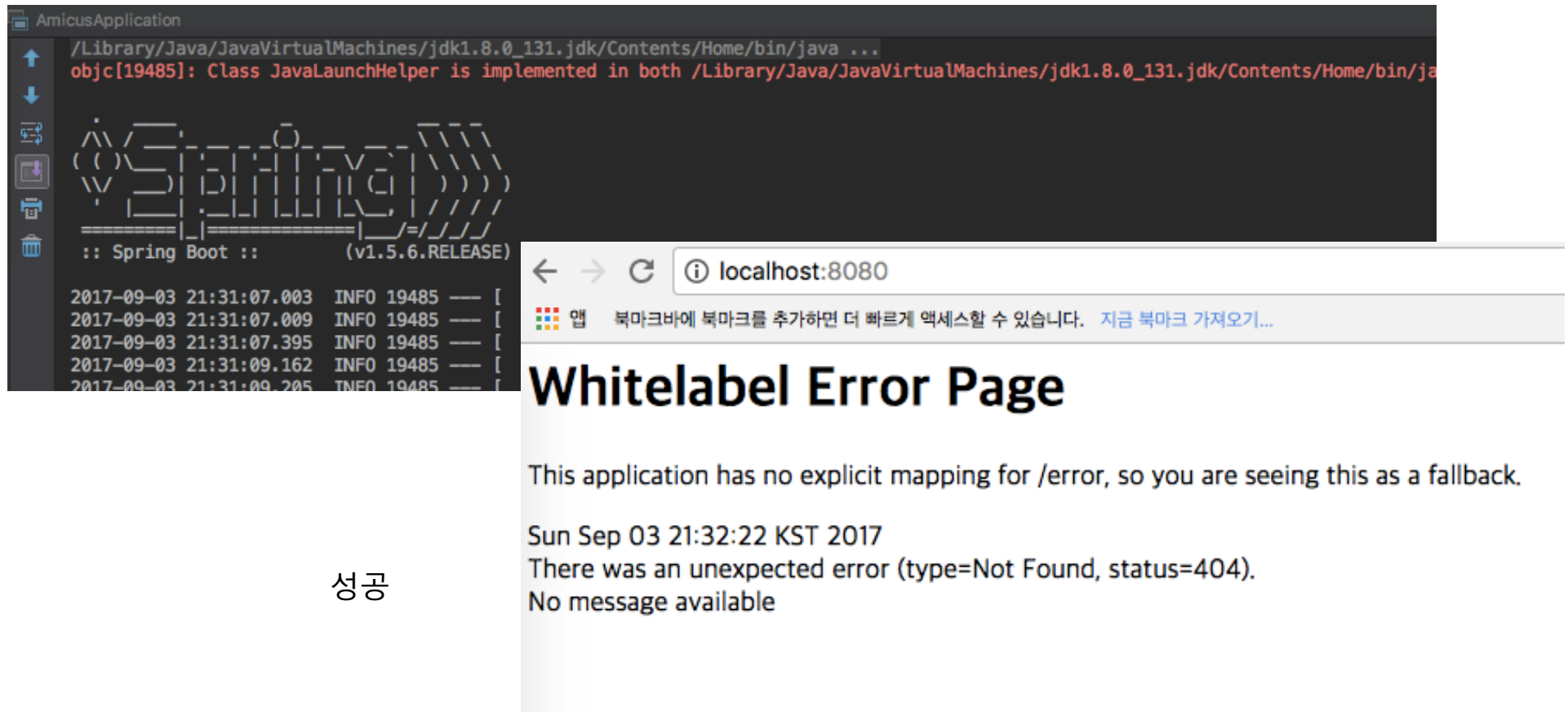
MongoDB ×

Generate Project % + ↗

# 프로젝트 셋업



# 실행



The image shows a terminal window on the left and a web browser on the right. The terminal window, titled 'AmicusApplication', displays the command `/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java ...` and the output `objc[19485]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/ja`. Below the command, the Spring Boot logo is displayed, followed by `:: Spring Boot :: (v1.5.6.RELEASE)`. The terminal also shows a log of INFO messages from 2017-09-03 21:31:07.003 to 2017-09-03 21:31:09.205, all from PID 19485. The web browser on the right shows the address bar with `localhost:8080` and a message bar with a bookmark icon and the text '앱 북마크바에 북마크를 추가하면 더 빠르게 액세스할 수 있습니다. 지금 북마크 가져오기...'. The main content of the browser is a 'Whitelabel Error Page' with the following text: 'This application has no explicit mapping for /error, so you are seeing this as a fallback.', 'Sun Sep 03 21:32:22 KST 2017', 'There was an unexpected error (type=Not Found, status=404).', and 'No message available'.

성공

# 컨트롤러와 기본 페이지 등록

Static : html, css, js

Template : thymeleaf, jsp 등등.

별도의 view resolver 설정이 필요

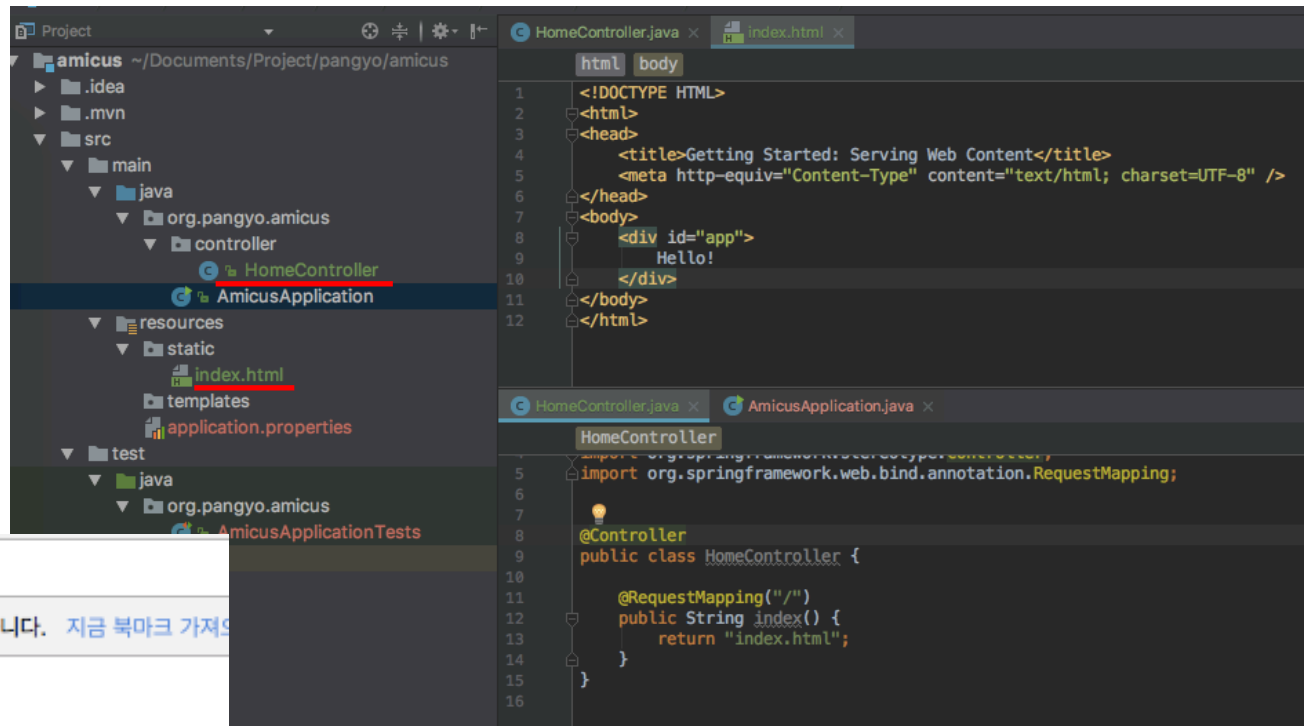
Spring static resource default path:

/static

/public

/resources

/META-INF/resources



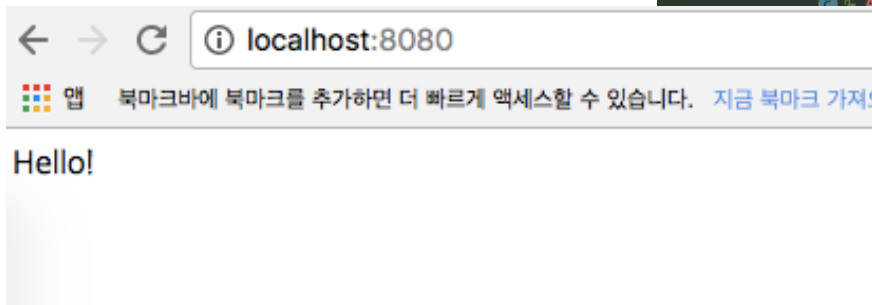
The screenshot shows an IDE with two main panels. The left panel displays the project structure for 'amicus' located at '~\Documents\Project\pangyo\amicus'. The structure includes folders for '.idea', '.mvn', 'src', and 'test'. Under 'src/main/java', there is a package 'org.pangyo.amicus' containing 'HomeController' and 'AmicusApplication'. Under 'src/main/resources', there is a 'static' folder containing 'index.html', and 'templates' and 'application.properties' files. The right panel shows the code for 'HomeController.java' and 'AmicusApplication.java'. The 'HomeController' class has a single method 'index()' that returns 'index.html'. The 'AmicusApplication' class is the main application class. The bottom panel shows the 'index.html' file, which is a simple HTML page with a title 'Getting Started: Serving Web Content' and a body containing 'Hello!'.

```
Project
├── amicus
│   ├── .idea
│   ├── .mvn
│   └── src
│       ├── main
│       │   ├── java
│       │   │   ├── org.pangyo.amicus
│       │   │   │   ├── HomeController
│       │   │   │   └── AmicusApplication
│       │   └── resources
│       │       ├── static
│       │       │   └── index.html
│       │       ├── templates
│       │       └── application.properties
│       └── test
│           ├── java
│           │   └── org.pangyo.amicus
│           │       └── AmicusApplicationTests
│           └── resources
```

```
HomeController.java
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <title>Getting Started: Serving Web Content</title>
5  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6  </head>
7  <body>
8  <div id="app">
9  Hello!
10 </div>
11 </body>
12 </html>
```

```
HomeController.java
1  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
2  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
3  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
4  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
5  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
6  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
7  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
8  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
9  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
10 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
11 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
12 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
13 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
14 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
15 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
16 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
```

```
AmicusApplication.java
1  import org.springframework.boot.SpringApplication;
2  import org.springframework.boot.autoconfigure.SpringBootApplication;
3  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
4  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
5  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
6  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
7  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
8  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
9  import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
10 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
11 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
12 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
13 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
14 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
15 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
16 import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
```



# React 설치 – npm init

```
[Gary] 21:55:55 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicus/src/main/resources/static (master *+ u= origin/master)$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (static) amicus
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/package.json:
{
  "name": "amicus",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes) yes
```

# React 설치 – react 설치

```
[Gary] 21:56:35 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicus/src/main/resources/static (master *+ u= origin/master)$ npm install --save react react-dom
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ react-dom@15.6.1
+ react@15.6.1
added 19 packages in 5.666s
```

# React 설치 – webpack 설치

```
[Gary] 22:02:51 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicus/src/main/resources/static (master *+ u= origin/master)$ npm install --save webpack

> fsevents@1.1.2 install /Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/node_modules/fsevents
> node install

[fsevents] Success: "/Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/node_modules/fsevents/lib/binding/Release/node-v57-darwin-x
Pass --update-binary to reinstall or --build-from-source to recompile

> uglifyjs-webpack-plugin@0.4.6 postinstall /Users/gwiyeong/Documents/Project/pangyo/amicus/src/main/resources/static/node_modules/uglifyjs-webpack-plugin
> node lib/post_install.js

npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ webpack@3.5.5
added 362 packages in 35.279s
```



# React 설치 – babel 설치

npm install --save babel-loader babel-core babel-preset-es2015 babel-preset-react babel-plugin-transform-class-properties

```
[Gary] 22:03:42 gwiyeong@AL01011625 ~/Documents/Project/pangyo/amicu
act
npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ babel-preset-es2015@6.24.1
+ babel-preset-react@6.24.1
+ babel-loader@7.1.2
+ babel-core@6.26.0
added 92 packages in 16.369s
```

npm install --save extract-text-webpack-plugin style-loader css-loader #text plugin for css

```
[Gary] 22:25:14 gwiyeong@AL01011625 ~/Documents/Proj
npm WARN amicus@1.0.0 No description
npm WARN amicus@1.0.0 No repository field.

+ extract-text-webpack-plugin@3.0.0
added 2 packages in 7.19s
```

# React 설치 – webpack.config.js 설정

entry : bundle의 시작점  
output: bundle의 결과물

module : bundle에 사용되는 모듈

Webpack으로 build를 할 경우  
./src/index.js파일을 읽어  
./dist/index.js파일을 생성한다.

View에서는 ./dist/index.js를 import하도록 설정.

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Getting Started: Serving Web Content</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  <div id="app">
    Hello!
  </div>
</body>
<script type="text/javascript" src="/dist/index.js"></script>
</html>
```

```
var path = require('path');
var ExtractTextPlugin = require("extract-text-webpack-plugin");

module.exports = {
  entry: {
    'index': './src/index.jsx'
  },
  output: {
    path: path.resolve('dist'),
    filename: '[name].js'
  },
  module: {
    loaders: [{
      test: /\.js|jsx$/,
      loader: 'babel-loader',
      exclude: /node_modules/,
      query: {
        presets: ['es2015', 'react'],
        plugins: ['transform-class-properties'],
      }
    },
    {
      test: /\.css$/,
      loader: ExtractTextPlugin.extract({
        fallback: "style-loader",
        use: "css-loader"
      })
    }
  ],
  plugins: [
    new ExtractTextPlugin("[name].css"),
  ],
}
```

# React 설치 – package.json 설정

scripts.build 추가.

npm run build로 실행

```
{
  "name": "amicus",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "build": "./node_modules/.bin/webpack --config webpack.config.js --progress --colors"
  },
  "babel": {
    "presets": [
      "es2015"
    ]
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-plugin-transform-class-properties": "^6.24.1",
    "babel-preset-es2015": "^6.24.1",
    "babel-preset-react": "^6.24.1",
    "css-loader": "^0.28.7",
    "extract-text-webpack-plugin": "^3.0.0",
    "react": "^15.6.1",
    "react-dom": "^15.6.1",
    "style-loader": "^0.18.2",
    "webpack": "^3.5.5"
  }
}
```

# Hello React

```
시작 index.jsx x ... index.jsp x
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './css/index.css'
4
5 class Hello extends React.Component {
6   render() {
7     return (
8       <div>
9         Hello React!!!
10      </div>
11    );
12  }
13
14
15 ReactDOM.render(
16   <Hello />,
17   document.getElementById('app')
18 );

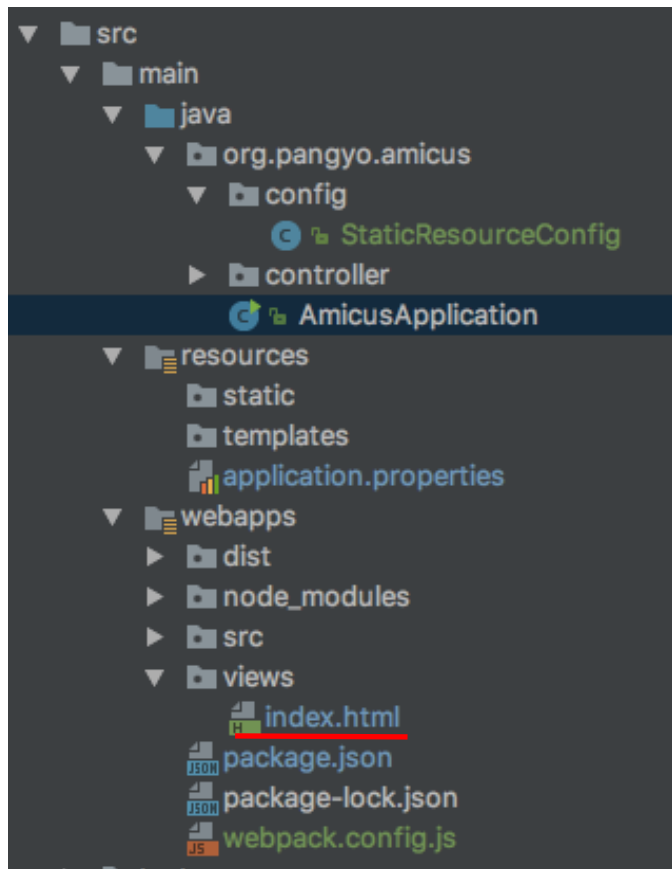
1 <%@ page language="java" contentType="text/html; charset=UTF-8" %>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <!DOCTYPE html>
4 <html lang="ko">
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" %>
7 <title>Hello</title>
8 </head>
9 <body>
10   <div id="app"></div>
11   <script type="text/javascript" src="../dist/index.js"></script>
12 </body>
13 </html>
```

# React 경로와 Spring 경로 설정.

Spring이 정적 resources를 serving하는 경로와 webpack이 resources를 떨구는 경로를 맞춰야 한다!!

먼저 spring의 정적 resources 경로를 설정한후 해당 경로 밑에 webpack이 컴파일 후 떨구도록 해야 한다.

# React 경로와 Spring 경로 설정.



thymeleaf template의 root 경로를 설정.

```
spring.thymeleaf.templates.root=src/main/webapps/views/
```

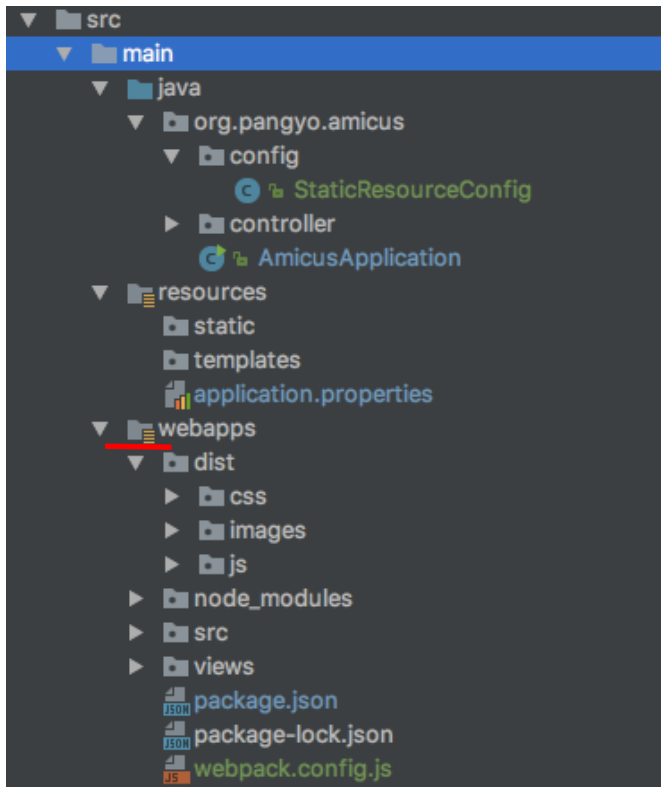
```
@Value("${spring.thymeleaf.templates.root}")
private String templatesRoot;

@Bean
public ITemplateResolver defaultTemplateResolver() {
    FileTemplateResolver resolver = new FileTemplateResolver();
    resolver.setSuffix(properties.getSuffix());
    resolver.setPrefix(templatesRoot);
    resolver.setTemplateMode(properties.getMode());
    resolver.setCacheable(properties.isCache());
    return resolver;
}
```

```
@RequestMapping("/")
public String home() { return "index"; }
```

Controller에서 string return시 template를 찾아서 rendering.

# React 경로와 Spring 경로 설정.



```
static.resource.location=classpath:/dist/
```

```
@Configuration
public class StaticResourceConfig extends WebMvcConfigurerAdapter {

    @Value("${static.resource.location}")
    private String staticResourceLocation;

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**").addResourceLocations(staticResourceLocation);
    }
}
```

1. webapps를 resources root로 설정 (build시 target.class폴더 밑에 webapps의 모든 파일/폴더 들이 복사된다.)
2. static.resource.location을 classpath:/dist/로 설정. (target.class폴더가 classpath로 잡혀있음)
3. /resources/경로에 classpath:/dist/경로를 mapping.

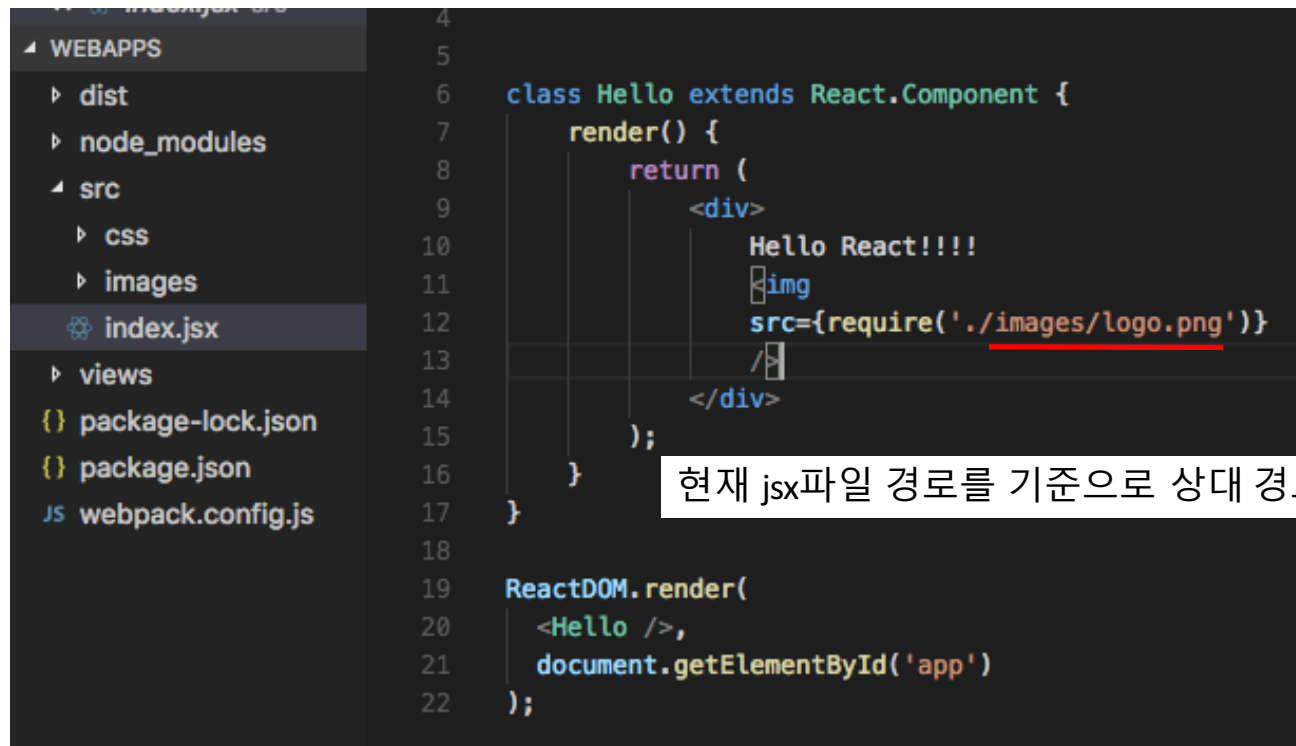
# React 경로와 Spring 경로 설정.

```
module.exports = {
  entry: {
    'index': './src/index.jsx'
  },
  output: {
    path: path.resolve('dist'), //compile된 파일들이 dist에 떨어짐.
    filename: 'js/[name].js',
    publicPath: '/resources/' //import code들에 prefix로 resources를 붙임. spring 설정에서 dist폴더가 resources경로에 mapping되어 있음.
  },
  module: {
    loaders: [{
      test: /\.jsx$/,
      loader: 'babel-loader',
      exclude: /node_modules/,
      query: {
        presets: ['es2015', 'react'],
        plugins: ['transform-class-properties'],
      }
    },
    {
      test: /\.css$/,
      exclude: /node_modules/,
      loader: ExtractTextPlugin.extract({
        fallback: "style-loader",
        use: "css-loader"
      })
    },
    {
      test: /\.?(gif|png|jpe?g|svg)$/i,
      exclude: /node_modules/,
      loaders: [
        'file-loader?hash=sha512&digest=hex&name=images/[hash].[ext]',
        'image-webpack-loader?{optimizationLevel: 7, interlaced: false, pngquant:{quality: "65-90", speed: 4}, mozjpeg: {quality: 65}}'
      ]
    }
  ]
}
```

1. Path를 path.resolve('dist')로 설정. Webpack.config가 있는 폴더를 기준으로 해당 폴더 아래 dist폴더에 output이 복사됨.
2. publicPath 를 /resources/로 설정. 관련 리소스를 가져오는 부분에서 prefix로 resources를 붙여줌.(spring에서 resources/\*\*에 대해서 Static resource를 mapping해 놔기 때문에 이처럼 설정 됨.)
3. Images나 font, css같은 경우 폴더를 나눠서 load하고 싶으면 loader에 prefix로 images등등을 붙인다. (그러면 webpack이 알아서 폴더 나눠서 dist아래에 넣어줌)



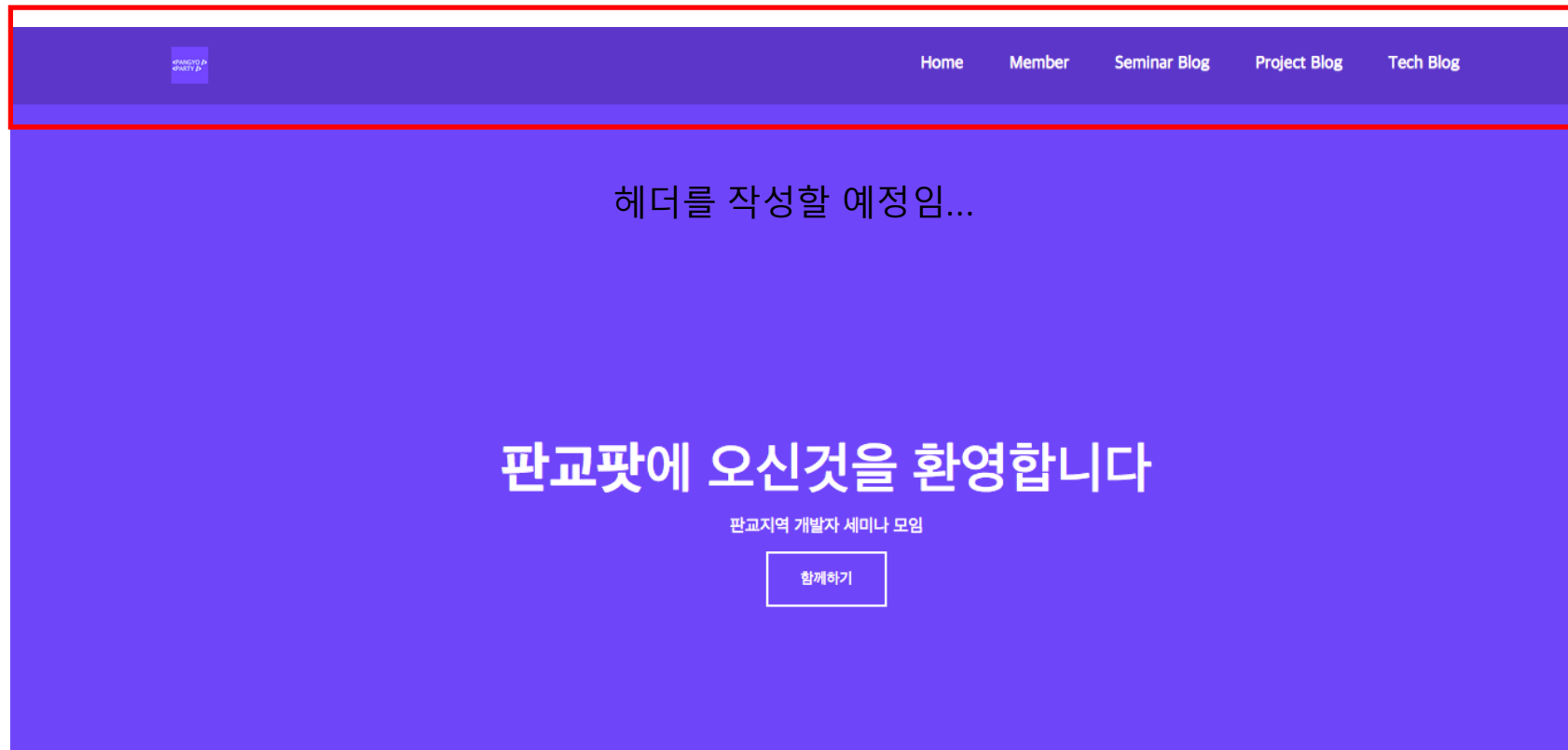
# React 경로와 Spring 경로 설정.



```
4
5
6 class Hello extends React.Component {
7   render() {
8     return (
9       <div>
10         Hello React!!!!
11         <img
12           src={require('./images/logo.png')}
13         />
14       </div>
15     );
16   }
17 }
18
19 ReactDOM.render(
20   <Hello />,
21   document.getElementById('app')
22 );
```

현재 jsx파일 경로를 기준으로 상대 경로 image file을 위와 같이 입력하면 load됨

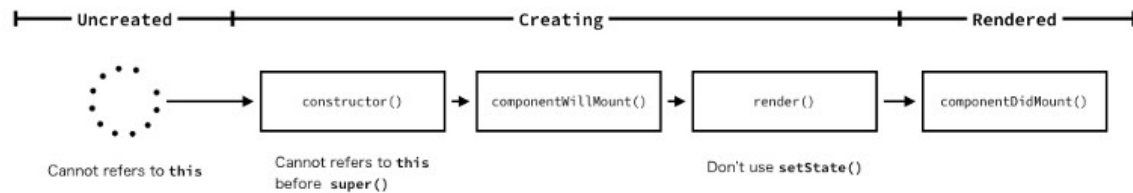
# React Component



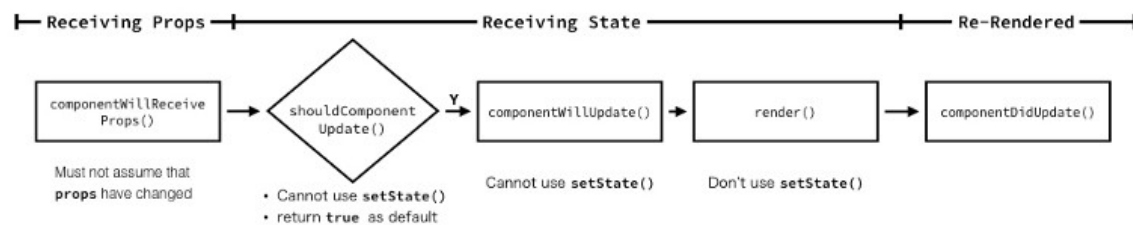
# React Component

## Component Life Cycle

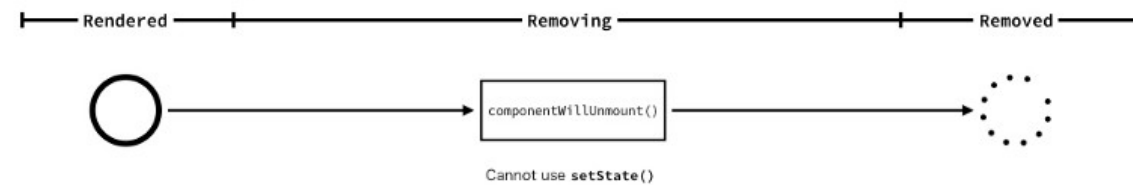
### Mounting



### Updating



### Unmounting

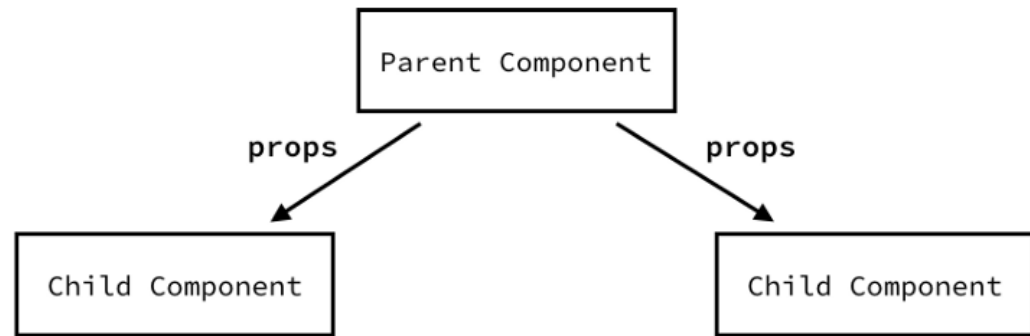


# React Component

Props, State

Props : 자식 컴포넌트 전달 하는 데이터  
단방향 전달(unidirection).

State : 현재 컴포넌트가 가진 데이터.  
setState로 state를 update한다.  
setState - 비동기적으로 동작하고,  
새로운 오브젝트를 생성하여 state를  
변형시키기 때문에 reference값이 변경되어  
참조하는 모든 컴포넌트가 변경을 인지할수 있고, rerender 된다.  
callback은 setState(nextState, callback)으로 작업 해야함.



# React Component

Presentation Component : render에  
필요한 데이터만 입력하면 view를  
그려주는 component

Container Component : 마크업 보다  
비즈니스 로직에 집중된, ajax,  
HOC(High Order Component)등을  
이용해 데이터를 fetching하는  
component.

```
// CommentList.js
import React from "react";

const CommentList = comments => (
  <ul>
    {comments.map(({ body, author }) =>
      <li>{body}-{author}</li>
    )}
  </ul>
)
```

Presentation

```
// CommentListContainer.js
import React from "react";
import CommentList from "../CommentList";

class CommentListContainer extends React.Component {
  constructor() {
    super();
    this.state = { comments: [] }
  }

  componentDidMount() {
    fetch("/my-comments.json")
      .then(res => res.json())
      .then(comments => this.setState({ comments }));
  }

  render() {
    return <CommentList comments={this.state.comments} />;
  }
}
```

container

# React Component

Header

Presentation component  
로 구현

Img경로를 require js로  
입력

```
index.jsx  header.jsx x  JS webpack.config.js
1  import React from 'react';
2
3  const Header = props =>
4  (<header id="masthead" className="masthead navbar navbar-default navbar-fixed-top" xmlns="http://www.w3.org/1999/html">
5    <div className="container">
6      <div className="navbar-header">
7        <button type="button" className="navbar-toggle collapsed" data-toggle="collapse" data-target="#main-menu">
8          <i className="fa fa-bars"></i>
9        </button>
10       <a className="navbar-brand" href="."><img src={require('../../images/logo.png')} alt="Site Logo"/></a>
11     </div>
12     <nav id="main-menu" className="collapse navbar-collapse pull-right">
13       <ul className="nav navbar-nav">
14         <li className="active"><a href="/">Home</a></li>
15         <li><a href="/">Member</a></li>
16         <li><a href="/">Seminar Blog</a></li>
17         <li><a href="#">Project Blog</a></li>
18         <li><a href="/">Tech Blog</a></li>
19       </ul>
20     </nav>
21   </div>
22 </header>)
23
24 export default Header;
```

# React Component

구현된 header를 import하고  
Header 태그로 입력.

```
import React from 'react';
import ReactDOM from 'react-dom';
import Header from './jsx/snippet/header.jsx';
import Slider from './jsx/snippet/slider.jsx';
import IndexSlider from './jsx/index-slider.jsx';
import Footer from './jsx/snippet/footer.jsx';

import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/css/bootstrap-theme.min.css';
import 'font-awesome/css/font-awesome.css';
import './css/index.css';

class Hello extends React.Component {
  render() {
    return (
      <div>
        <Header />
        <Slider />
        <IndexSlider />
        <Footer />
      </div>
    );
  }
}
```