

EJS 템플릿 엔진

개요

- 템플릿 엔진이란? (아래링크 참조)
 - http://inlab.cu.ac.kr/~wlgns12www/backend/Template_Engine.pdf
- EJS 템플릿 엔진이란?
 - Embedded JavaScript의 약자이다. 즉, 자바스크립트가 내장되어 있는 HTML 파일이다.
 - 다시 말하면, HTML 코드에 JavaScript를 삽입할 수 있는 것이다.
 - 예시) EJS 문법

```
<html>
  <head>
    <title><%= title %></title>
  </head>
  <body>
    <h1><%= title %></h1>
    <% for(var i=0 ; i<5 ; i++) { %>
      <p> Welcome to <%= name %></p>
    <% } %>
  </body>
</html>
```

※ HTML 코드 사이에 보이는 "<% %>"태그 내부에 JavaScript 코드가 들어가는 부분이다.
"<% %>"와 "<%= %>"의 기능에는 차이가 있으며 이 설명은 아래에서 하도록 한다.

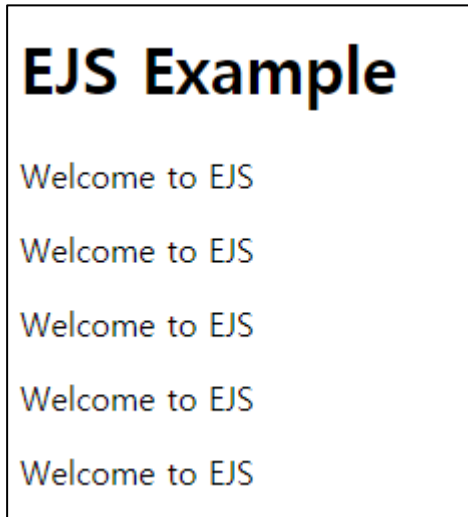
- ✓ Express에서는 EJS 말고도 Jade(PUG)라는 템플릿 엔진도 존재한다.
Jade는 HTML문법이 아닌 다른 문법으로 진행해야 함으로 여기서는 비교적 쉽게 배울 수 있는(HTML문법을 따르므로..) EJS를 해보도록 한다.

예시1) JavaScript 코드 사용 태그

- "<% %>" 태그 : JavaScript의 코드를 사용할 수 있는 태그이다.
 - 예시)

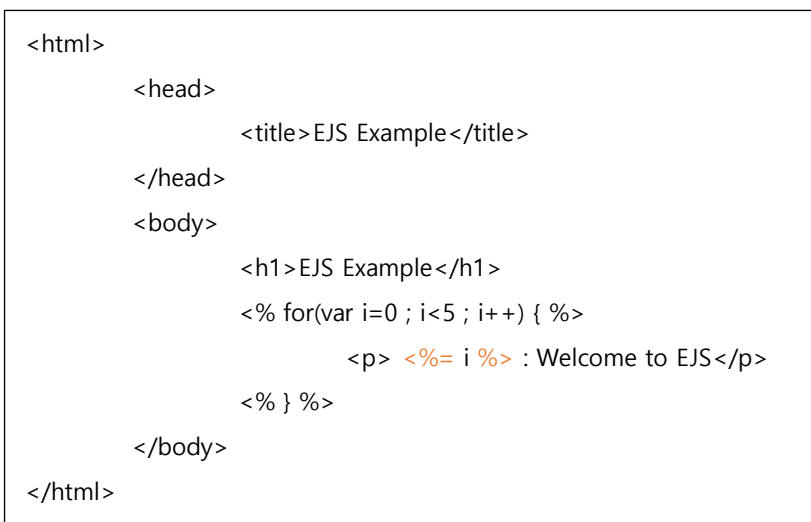
```
<html>
  <head> <title>EJS Example</title> </head>
  <body>
    <h1>EJS Example</h1>
    <% for(var i=0 ; i<5 ; i++) { %>
      <p> Welcome to EJS</p>
    <% } %>
  </body>
</html>
```

- 예시의 결과)

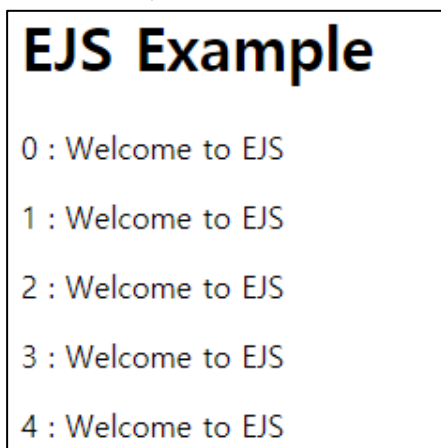


예시2) JavaScript 변수 사용 태그

- "<%= %>" 태그 : JavaScript의 변수를 사용할 수 있는 태그이다.
 - 예시)



- 예시의 결과)



실습1) Node.js : Express + EJS

- ※ 본 실습은 HTML의 문법 및 태그 등을 알고 있다는 전제하에 진행한다.
- ※ 본 실습에서는 Node.js의 Express 프레임워크에 EJS 템플릿 엔진을 달아보도록 한다.
Express generator로 초기화시엔 기본적으로 Jade(요즘은 pug라고 불림) 템플릿 엔진이 내장되어 있어서 옵션을 주어서 EJS 템플릿 엔진으로 초기화할 수 있도록 실행해야 한다.

1. Express + EJS 셋팅

\$ express --ejs [Application Name]

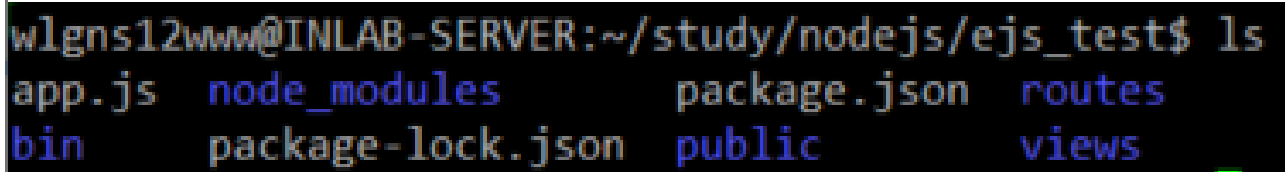
Ex) express --ejs ejs_test

2. Express + EJS 설치

\$ cd [Application Name] && npm install

Ex) cd ejs_test && npm install

- ✓ 2번까지 진행하였다면 초기화된 디렉터리는 아래와 같을 것이다.



```
wlgn12www@INLAB-SERVER:~/study/nodejs/ejs_test$ ls
app.js  node_modules  package.json  routes
bin     package-lock.json  public        views
```

- app.js : Node.js Application 파일
- routes : 라우터 디렉터리 (처음엔 index.js와 user.js 파일이 들어있음)
- bin : 실행 디렉터리 (내부에 www라는 파일로 Node.js를 실행시킨다.)
- public : 에셋(js, image, css)이 들어있는 디렉터리
- **views** : ejs파일이 들어있는 디렉터리

3. views/index.ejs 파일내용 확인

```
<html>
  <head>
    <title><%= title %></title>
    <link rel="stylesheet" href="/stylesheets/style.css" />
  </head>
  <body>
    <h1><%= title %></h1>
    <p>Welcome to <%= title %></p>
  </body>
</html>
```

- "<%= %>" 태그로 title이라는 변수를 HTML 태그에 담는 것을 볼 수 있다.
- title이라는 변수의 값은 routes/index.js 라우터에서 넘겨진다.

4. routes/index.js 파일내용 확인

```
var express = require("express");
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res) {
    res.render( 'index', { title : 'Express' } );
});
```

- **res.render('index', { title : 'Express' })**
: 템플릿 엔진에서 index.ejs파일과 title변수가 결합 및 렌더링 되어 HTML코드가 발생된다.
- res.render(file_path, json_data)
: res.render() >> 템플릿(file_path)을 json_data와 결합 및 렌더링하여 HTML코드를 클라이언트로 전달.
: file_path >> 렌더링할 템플릿 파일의 경로
: json_data >> 템플릿 파일로 입력될 JSON형식 Data

5. app.js 파일내용 확인

```
... (중략)

app.set( 'views', path.join( __dirname, 'views' ));
app.set( 'view engine', 'ejs' );

... (중략)
```

- **app.set('views', path.join(__dirname, 'views'))**
: view들의 시작 경로를 지정
: 해당 함수로 인해 4번 index.js의 res.render()에서 index.ejs의 경로를 다 적어주지 않은 것이다.
 - **app.set('view engine', 'ejs')**
: view engine(템플릿 엔진)을 ejs로 설정
- ✓ 5번까지 진행하였다면 EJS가 어떻게 동작되는지 대충은 감을 잡았을 것이다.
이제 Application을 실행시켜보도록 하자.
(express --ejs [Application Name] 명령으로 인해 기본적인 셋팅은 완료된 상태 => App실행가능)

6. Application 실행

\$./bin/www

>> www 실행파일은 nodejs명령어로 실행되는 파일임으로 node 명령을 앞에 붙이지 않아도 된다.
(붙여도 무관)

7. 테스트 진행

7.1. 웹 브라우저를 통해 테스트



- res.render() 함수를 통해 views/index.ejs 파일과 title 변수가 결합 및 렌더링 되어 웹 브라우저에서는 위와 같은 결과를 출력한다.
- 빨간 박스 부분이 템플릿 파일에 데이터가 입력된 부분이다.

7.2. cUrl을 통해 테스트

\$ curl -X GET http://localhost:3000/

```
<!DOCTYPE html>
<html>
  <head>
    <title>Express</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Express</h1>
    <p>Welcome to Express</p>
  </body>
</html>
```

- 결과 >>
- 빨간 박스 부분이 템플릿 파일에 데이터가 입력된 부분이다.

참고문헌

- 주인장 야호호코코 (2019). <https://yahohococo.tistory.com/43>
- 박진형 (2017). <https://jinbroing.tistory.com/107>
- <https://expressjs.com/ko/api.html#res.render>