Building a static site with Node and Express

Samkeun Kim <skim@hknu.ac.kr>

http://cyber.hankyong.ac.kr

국립한경대학교 HANKYONG NATIONAL UNIV.

This chapter covers

- Prototyping an application through building a ***static*** version
- Defining routes for application URLs
- Creating views in Express using Pug and Bootstrap
- Using controllers in Express to tie routes to views
- Passing data from controllers to views

이 장에서는 애플리케이션 아키텍처를 구축할 때의 **Express** 애플리케이션에 집중한다.

두 가지 메인 스텝이 있어서 두 가지 이용가능한 소스 코드 버전이 있다.

첫 번째 버전 – 뷰에 모든 데이터를 포함한다.

두 번째 버전 – 컨트롤러에 데이터를 포함한다.

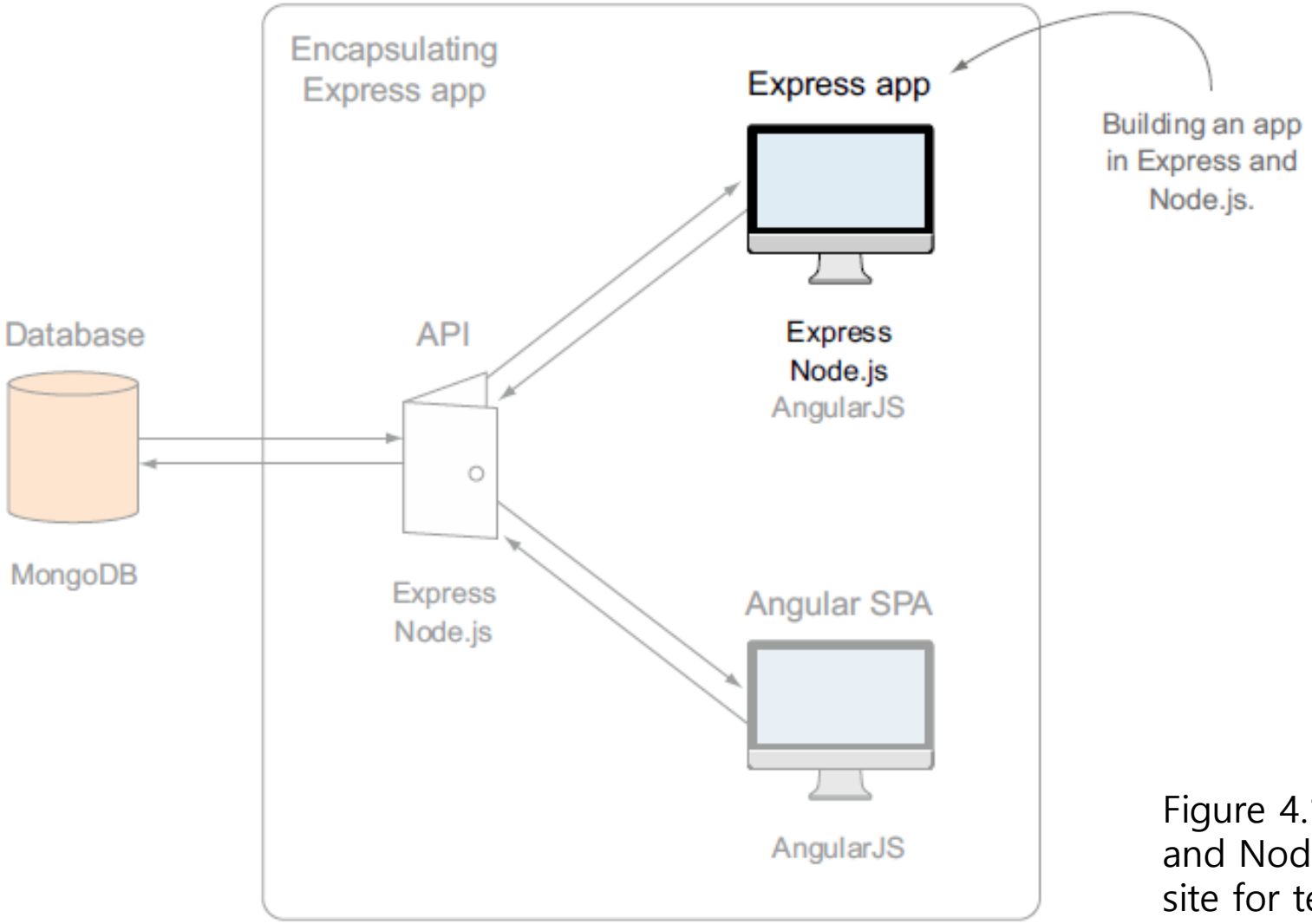


Figure 4.1 Using Express and Node to build a static site for testing views

Planning a real application

Loc8r MEAN 스택 애플리케이션을 구축해보자.

Loc8r는 사람들이 가서 뭔가 작업할 수 있는 WiFi가 있는 인근 장소를 나열해준다.

또한 각 장소의 시설, 영업 시간, 평점, 위치지도를 표시해준다.

사용자들은 로그인하여 등급과 리뷰를 제출할 수 있다.

데모 애플리케이션을 위해 Fake 데이터를 생성해서 테스트할 수 있도록 한다.

Planning the application at a high level

1. 요구되는 스크린 도출; 독립된 페이지 뷰

이 단계를 스케치하면 애플리케이션을 전체적으로 시각화하는데 도움이 된다.

2. 스크린을 Collection들로 구조화 => Good reference point 제공

- ✓ 페이지 또는 애플리케이션 로직에 첨부된 데이터가 없기 때문에
  부분들을 추가하고 제거하는 작업,
  표시되는 내용을 변경하는 작업,
  원하는 페이지 수를 변경하는 작업
  등이 정말 쉽다

PLANNING THE SCREENS

Loc8r MEAN 스택 애플리케이션을 구축해보자.

Loc8r는 사람들이 가서 뭔가 작업할 수 있는 WiFi가 있는 인근 장소를 나열해준다.

또한 각 장소의 시설, 영업 시간, 평점, 위치지도를 표시해준다.

사용자들은 로그인하여 등급과 리뷰를 제출할 수 있다.

스크린 도출:

1. 주변 장소를 나열하는 화면
2. 개별 장소에 대한 세부 정보를 보여주는 화면
3. 장소에 대한 리뷰를 추가하는 화면
4. “About us” 정보 화면

DIVIDING THE SCREENS INTO COLLECTIONS

화면 목록을 논리적으로 묶는다:

- 리스트의 처음 3개는 모두 장소를 다루고 있으므로 **Locations**로 묶음
- About 페이지는 실제로 어디에도 속하지 않으므로 **Others** 컬렉션에 포함시킴

이를 스케치하면 다음 그림과 같다.

- ✓ 이 스케치 단계는 아키텍처를 설계하기 전에 반드시 거치는 것이 좋다
- ✓ 기본 페이지들을 살펴 볼 기회 제공 / 전체 흐름에 대해 검토할 기회 제공

Locations

List page

Details page

Add Review page

Others

About page

Collections of screens we'll be building for the Loc8r application

Defining the routes in Express

일련의 스크린과 Incoming URL을 연관지어야 한다.

표 4.1: 화면과 URL의 간단한 매핑을 보여준다. 애플리케이션을 위한 라우팅의 기초가 된다.

Table 4.1 Defining a URL path, or route, for each of the screens in the prototype

Collection	Screen	URL path
Locations	List of locations (this will be the homepage)	/
Locations	Location detail	/location
Locations	Location review form	/location/review/new
Others	About Loc8r	/about

예를 들어, 누군가 홈페이지(/)를 방문하면 장소 목록을 보여주고 싶을 것이지만, 누군가 /about 페이지를 방문하면 Loc8r에 대한 정보를 보여주고 싶을 것이다.

Different controller files for different collections

앞에서 라우터와 컨트롤러를 분리시켰다.

애플리케이션은 곧 규모가 커질 것이고, 반드시 하나의 파일에 모든 컨트롤러를 집어 넣을 필요는 없다.

논리적으로 분리하기 위한 시작 포인트는 그들을 컬렉션(**Collection**)으로 나누는 것이다.

- ✓ 그림 4.3처럼 컬렉션을 검토해서 컨트롤러를 **Locations**와 **Others** 그룹으로 나눈다.
- ✓ 애플리케이션에는 라우트 파일이 포함되어 있다:
  - 라우트 파일에 여러 컨트롤러 파일을 포함시키고, 관련 컬렉션에 따라 각 파일 이름을 지정한다

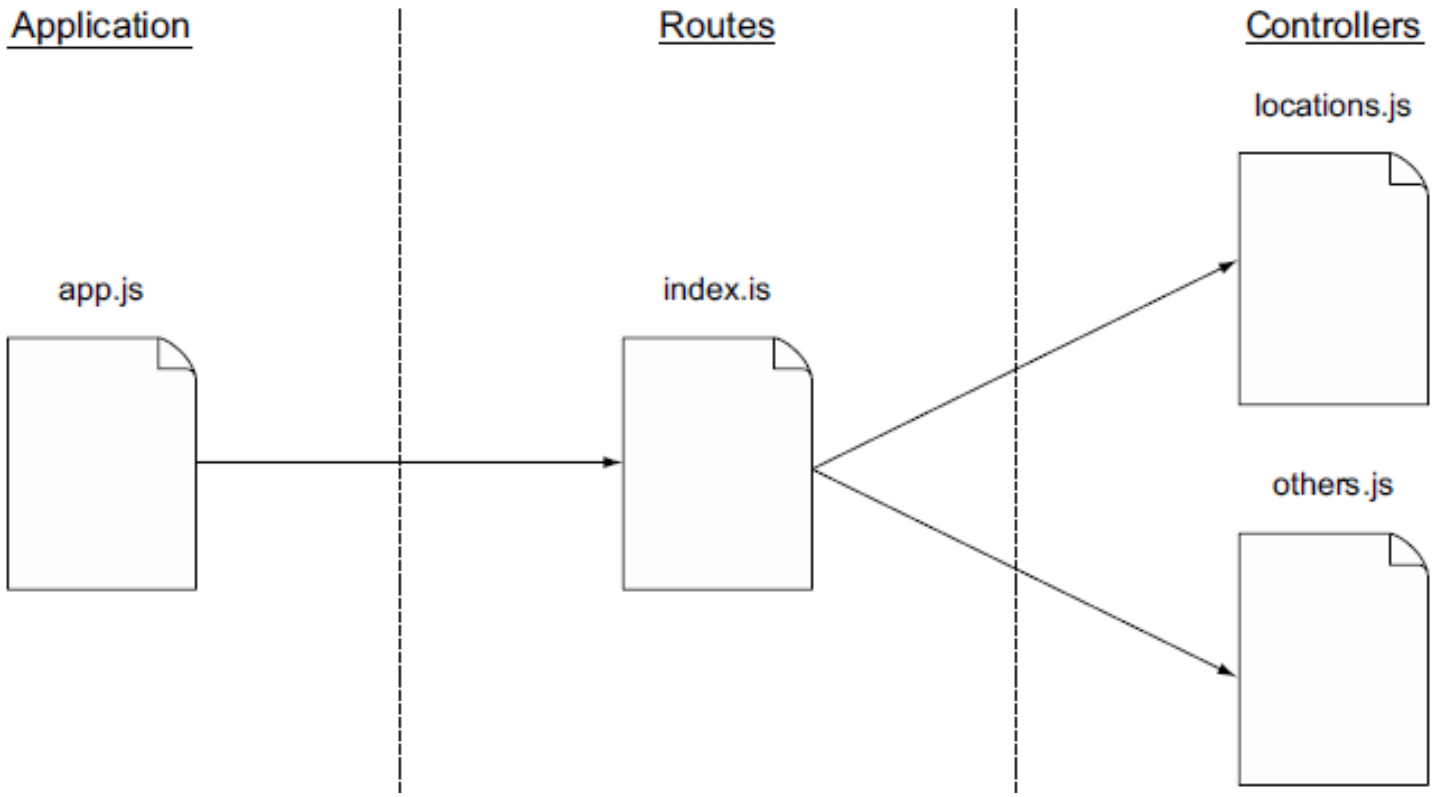


Figure 4.3 Proposed file architecture for routes and controllers in our application

1개의 라우트 파일과 각 논리적 스크린 Collection들에 대한 각각의 컨트롤러 파일이 있다.

개발환경으로 되돌아 가서 애플리케이션 오픈 => 라우트 파일 **index.js** 부터 살펴보자.

REQUIRING THE CONTROLLER FILES

라우트 파일에서 2개의 컨트롤러 파일을 레퍼런스하고 싶다: **locations.js**와 **others.js**

⇒ 이들 파일을 app_server/controllers에 저장한다

index.js에서 이 두 파일을 모두 require 시키고 관련 변수 이름에 할당한다.

Listing 4.1 Requiring the controller files in routes/index.js

```
var express = require('express');
var router = express.Router();
var ctrlLocations = require('../controllers/locations');
var ctrlOthers = require('../controllers/others');
```

Replace existing ctrlMain reference with two new requires

이제 라우트 정의에서 참조할 수 있는 2개의 변수를 얻었고, 이들은 서로 다른 **라우트 컬렉션**을 포함하게 된다.

SETTING UP THE ROUTES

index.js에 아래의 라우트를 포함시킨다:

- ✓ Locations 컬렉션에 있는 **3개의 스크린**을 위한 라우트
- ✓ Others 컬렉션에 있는 **About** 페이지를 위한 라우트

각 라우트는 컨트롤러에 대한 레퍼런스를 require한다.

- ✓ 라우트는 들어오는 요청 URL을 애플리케이션 기능의 특정 부분에 매핑시켜 주는 매핑 서비스일 뿐이다.

표 4.1에 정의된 모든 패스를 **routes/index.js** 파일에 모두 포함시킨다.

- ✓ 파일에 있어야 할 내용은 다음 리스트에 전체적으로 표시되어 있다.

Listing 4.2 Defining the routes and mapping them to controllers *routes/index.js*

```
var express = require('express');
var router = express.Router();
var ctrlLocations = require('../controllers/locations');
var ctrlOthers = require('../controllers/others');

/* Locations pages */
router.get('/', ctrlLocations.homelist);
router.get('/location', ctrlLocations.locationInfo);
router.get('/location/review/new', ctrlLocations.addReview);

/* Other pages */
router.get('/about', ctrlOthers.about);

module.exports = router;
```

Require controller files

Define location routes and map them to controller functions

Define other routes

라우팅 파일은 정의된 URL을 특정 컨트롤러에게 매핑시킨다.

이제 컨트롤러를 만들어 보자.

Building basic controllers

이 시점에서 간단한 컨트롤러를 만들어 애플리케이션을 실행하고 다른 URL과 라우팅을 테스트할 수 있다.

Setting up controllers

현재는 controllers 폴더의 홈페이지를 제어하는 단일 메소드(index)를 가지는 **main.js** 파일만 있다:

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

실제로 "**main**" 컨트롤러 파일을 더 이상 원하지 않지만, 이것을 템플릿으로 사용할 수는 있다.

이 파일을 **others.js**로 이름을 변경하자.

ADDING THE OTHERS CONTROLLERS

Listing 4.2에서 `others.js`에 있는 about이라는 컨트롤러가 호출되기를 바란다.

- ✓ 기존 index 컨트롤러 이름을 'about'으로 변경하자.
- ✓ 뷰 템플릿은 동일하게 유지하고, 'title' 프로퍼티를 관련된 내용으로 수정한다.

Listing 4.3 Others controller file

```
/* GET 'about' page */
module.exports.about = function(req, res){
  res.render('index', { title: 'About' });
};
```

Define route using same view template but changing title to About

첫 번째 작업이 완료되었지만, **Locations** 라우트에 대한 컨트롤러들이 아직 없기 때문에 애플리케이션이 동작하지는 않는다.

ADDING THE LOCATIONS CONTROLLERS

Locations 라우트에 대한 컨트롤러 추가 => 앞과 동일한 프로세스!

- ✓ 라우트 파일에 3개의 컨트롤러 함수 이름 지정
- ✓ controllers 폴더에서 **locations.js**라는 파일을 만들고, homelist, locationInfo, addReview의 3 가지 기본 컨트롤러 기능을 만든다.

Listing 4.4 Locations controller file

```
/* GET 'home' page */
module.exports.homelist = function(req, res){
  res.render('index', { title: 'Home' });
};

/* GET 'Location info' page */
module.exports.locationInfo = function(req, res){
  res.render('index', { title: 'Location info' });
};

/* GET 'Add review' page */
module.exports.addReview = function(req, res){
  res.render('index', { title: 'Add review' });
};
```

Testing the controllers and routes

```
$ nodemon
```

Troubleshooting

If you're having problems restarting the application at this point, the main thing to check is that all of the files, functions, and references are named correctly. Look at the error messages you're getting in the terminal window and see if they give you any clues. Sometimes they're more helpful than others! Take a look at the following possible error, and we'll pick out the parts that are interesting to us:

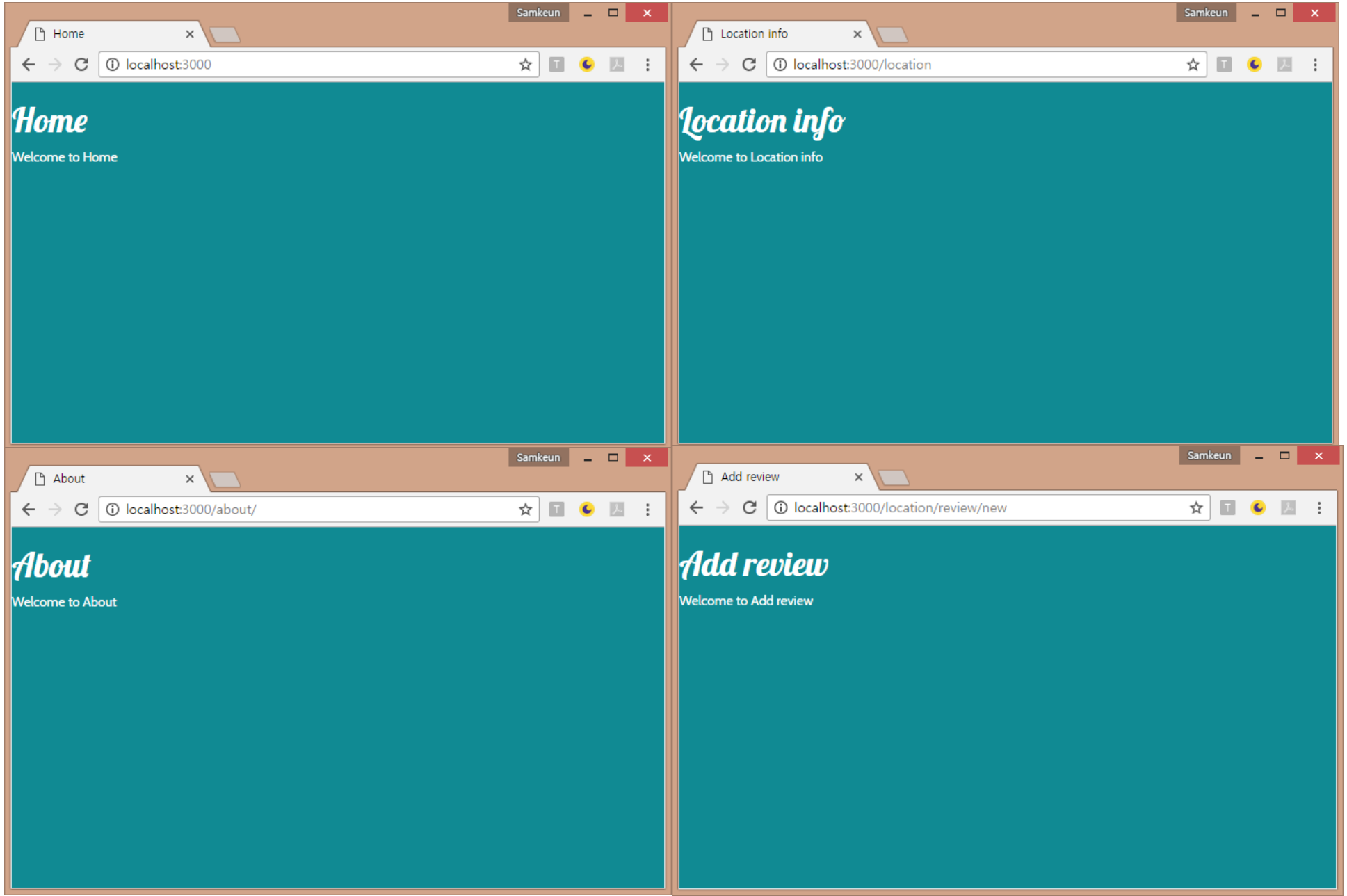
```
module.js:340
    throw err;
          ^

Error: Cannot find module '../controllers/other'
    at Function.Module._resolveFilename (module.js:338:15)
    at Function.Module._load (module.js:280:25)
    at Module.require (module.js:364:17)
    at require (module.js:380:17)
    at module.exports (/Users/sholmes/Dropbox/Manning/Getting-MEAN/Code/
          Loc8r/BookCode/routes/index.js:2:3)
    at Object.<anonymous> (/Users/sholmes/Dropbox/Manning/Getting-MEAN/
          Code/Loc8r/BookCode/app.js:26:20)
    at Module._compile (module.js:456:26)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
```

❶ **Clue one: a module can't be found**

❷ **Clue two: file throwing error**

First, you can see that a module called `other` can't be found ❶. Further down the stack trace you can see the file where the error originated ❷. So you'd then open the routes/index.js file and discover that you'd written `require('../controllers/other')` when the file you want to require is others.js. So to fix the problem you'd simply need to correct the reference by changing it to `require('../controllers/others')`.

실습과제 24-1 Screenshots of the four routes



Creating some views

Empty 페이지, 패스, 라우트를 가졌다면, 이제 애플리케이션에 **콘텐트와 레이아웃**을 가져올 때이다.

아이디어를 현실로 만들어 보자.

필요한 기술: Pug와 Bootstrap

Pug는 Express의 디폴트 템플릿 엔진이다.

Bootstrap은 데스크탑과 모바일 장치에서 다르게 보이는 반응형 웹 사이트를 쉽게 구성할 수 있게 해주는 프론트엔드 레이아웃 프레임워크이다.

A look at Bootstrap

시작하기 전에 **Bootstrap**을 간단히 살펴보자.

Bootstrap이 할 수 있는 모든 것에 대한 자세한 내용은 다루지 않겠지만, 템플릿 파일에 끼워 넣기 전에 핵심 개념을 다시 한번 정리해 보는 것이 유용하다.

BOOTSTRAP RESPONSIVE GRID SYSTEM

Bootstrap은 **12컬럼 그리드**를 사용한다. 사용중인 디스플레이의 크기에 관계없이 항상 이 12개의 컬럼이 있다.

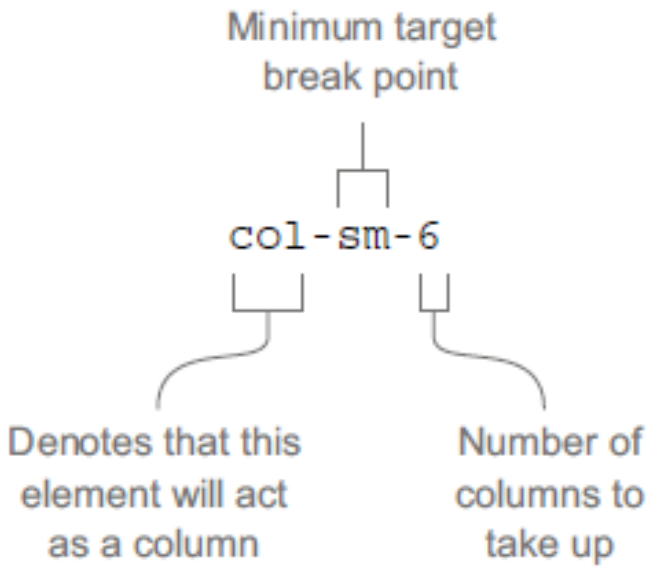
- 폰에서는 각 컬럼은 좁아지고, 커다란 외부 모니터에서는 각 컬럼이 넓어진다.
- Bootstrap의 기본 개념은 요소가 사용할 컬럼의 수를 정의할 수 있으며, 이는 다른 화면 크기에 따라 다른 개수가 된다.
- Bootstrap에는 레이아웃에 대해 최대 네 개의 서로 다른 **pixelwidth** 브레이크 포인트를 타겟으로 할 수 있는 다양한 **CSS 레퍼런스**가 있다.
- 브레이크 포인트는 각 크기에서 타겟으로 삼을 장치 예와 함께 표 4.2에 나와 있다.

Table 4.2 Breakpoints that Bootstrap targets for different types of devices

Breakpoint name	CSS reference	Example device	Width in pixels
Extra-small devices	`xs`	Phones	Less than 768
Small devices	`sm`	Tablets	768 or more
Medium devices	`md`	Laptops	992 or more
Large devices	`lg`	External monitors	1,200 or more

요소의 너비를 정의하려면 표 4.2의 **CSS 레퍼런스**와 범위를 확장하려는 **컬럼 수**를 결합하면 된다.

컬럼을 나타내는 클래스는 아래처럼 구성한다:



이 클래스의 **col-sm-6**은 적용된 요소를 크기가 **sm** 이상인 화면에서 6개의 컬럼을 차지하도록 만든다.

따라서 태블릿, 랩톱, 모니터에서 이 컬럼은 사용 가능한 너비의 절반을 차지한다.

반응형으로 만들기 위해 단일 요소에 여러 클래스를 적용할 수 있다.

폰에서는 전체화면으로, 태블릿에서는 절반의 폭으로 span하는 div 태그를 원한다면:

```
<div class="col-xs-12 col-sm-6"></div>
```

col-xs-12 클래스는 폰(초소형 장치)에서 12개의 컬럼을 사용하도록 레이아웃에 지시하고,

col-sm-6 클래스는 태블릿(소형 장치) 이상에서 6 컬럼을 사용하도록 레이아웃에 지시한다.

```
<div class="col-xs-12 col-sm-6">DIV ONE</div>
<div class="col-xs-12 col-sm-6">DIV TWO</div>
```

그림 4.5는 서로 다른 장치에 미치는 영향을 보여준다:

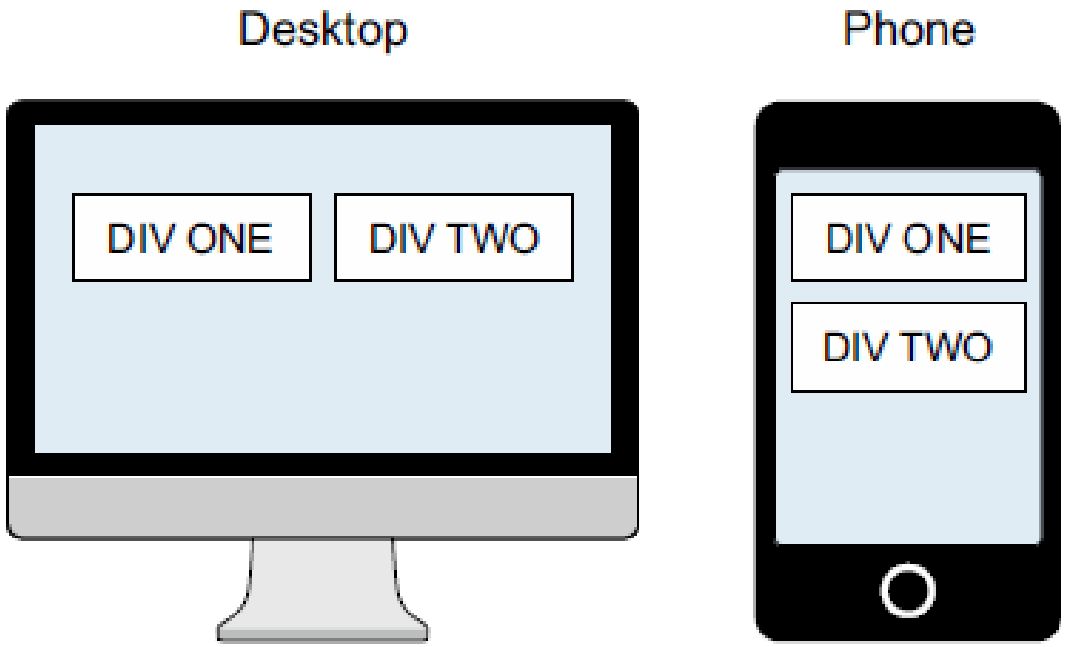


Figure 4.5 Bootstrap's responsive column system on a desktop and mobile device. CSS classes are used to determine the number of columns out of 12 that each element should take up at different screen resolutions.

Setting up the HTML framework with Pug templates and Bootstrap

모든 페이지에 공통적인 요소가 있다.

페이지의 위쪽에는 **Navigation bar**와 **Logo**가 있고,

페이지의 아래 쪽에는 **Footer**로서 **Copyright notice**가

들어가고, 중앙에는 **Content**가 있게 된다.

여기서 우리가 목표로 하는 것은 그림 4.6과 같다.

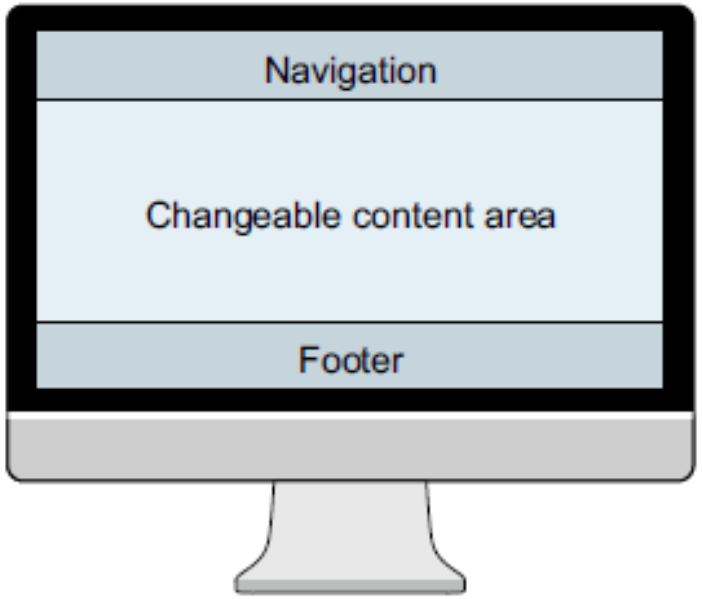


Figure 4.6 Basic structure of the reusable layout, comprising a standard navigation bar and footer with an extendable and changeable content area in between

LOOKING AT THE LAYOUT

공통 프레임워크 구축을 위해 `layout.pug` 파일(in app_server/views)에서 작업:

```
doctype html
html
  head
    meta(name='viewport', content='width=device-width, initial-scale=1.0')
    title= title
    link(rel='stylesheet', href='/bootstrap/css/amelia.bootstrap.css')
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    block content

    script(src='/javascripts/jquery-1.11.1.min.js')
    script(src='/bootstrap/js/bootstrap.min.js')
```

Body 영역에는 HTML 내용이 전혀 없으며, **content**라고 하는 블록과 스크립트 참조가 2개 있다.

이 모든 것을 유지하면서 **content** 블록 위에 **navigation** 섹션을 추가하고 그 아래에는 **footer**를 추가해야 한다.

BUILDING THE NAVIGATION

Bootstrap은 스크린 상단에 고정 Navigation bar를 위한 엘리먼트와 클래스들을 지원한다.

- 모바일 장치에서는 옵션을 드롭다운 메뉴로 Collapse 시켜 준다.

네비게이션에서 아래 두 가지를 수행하도록 한다:

- ✓ 홈페이지에 링크된 Loc8r 로고
- ✓ /about URL 페이지를 가리키는 About 링크

이 모든 작업을 수행하는 코드를 `layout.pug` 파일의 **block content 라인 위**에 놓는다:

```
.navbar.navbar-default.navbar-fixed-top
  .container
    .navbar-header
      a.navbar-brand(href='/') Loc8r
      button.navbar-toggle(type='button', data-toggle='collapse', data-
 target='#navbar-main')
        span.icon-bar
        span.icon-bar
        span.icon-bar
    #navbar-main.navbar-collapse.collapse
      ul.nav.navbar-nav
        li
          a(href='/about/') About
```

Set up a Bootstrap navigation bar fixed to top of window

Add a brand-styled link to homepage

Set up collapsing navigation for smaller screen resolutions

Add About link to left side of bar

TIP **Pug**에는 HTML 태그가 포함되어 있지 않으며, 올바른 **들여 쓰기**가 중요하다는 점을 기억하자.

WRAPPING THE CONTENT

`layout.pug` 파일의 **content block** 라인

- ✓ 이 라인에서 할 일은 거의 없다 => 다른 Pug 파일(`index.pug`)에서 내용 결정!

content block은 장치의 전체 폭에 표시된다.

⇒ 단순히 content block을 container **`div`**로 감싸기만 하면 된다.

```
.container
  block content
```

⇒ `.container` 클래스를 갖는 **`div`**는 윈도우의 **가운데에 배치**된다.

⇒ Container `div`의 **content**는 정상적으로 **왼쪽 정렬**된 채로 유지된다.

ADDING THE FOOTER

페이지 끝 부분에 표준 footer를 추가해 보자.

- ✓ 여기에 이용 약관, 개인 정보 취급 방침 또는 링크를 추가할 수 있다.
- ✓ 레이아웃 파일에 들어가기 때문에 나중에 모든 페이지에서 이를 쉽게 업데이트할 수 있다.

간단한 **footer** 예:

```
footer
  .row
     .col-xs-12
        small &copy; Samkeun Kim
```

content block을 저장하고 있는 container div **내부**에 놓여지는 것이 가장 적합하다.

추가할 때 footer 행이 block content 행과 **동일한 들여 쓰기 레벨**에 있는지 확인해야 한다.

ALL TOGETHER NOW

이제 `navigation bar, content area, footer` 모두를 포함한 layout이 완성되었다.

`layout.pug`의 전체 코드는 다음 리스트에 표시된다.

주의) Pug 엔진은 탭(Tab)과 빈칸(blank)을 엄격하게 구분!

한 가지 방식으로만 작성되어야 함. **탭 방식 권장!!**

Listing 4.5 Final code for the layout framework in app_server/views/layout.pug

```
doctype 5
html
  head
    meta(name='viewport', content='width=device-width, initial-scale=1.0')
    title= title
    link(rel='stylesheet', href='/bootstrap/css/amelia.bootstrap.css')
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    .navbar.navbar-default.navbar-fixed-top
      .container
        .navbar-header
          a.navbar-brand(href='/') Loc8r
          button.navbar-toggle(type='button', data-toggle='collapse',
    ➥ data-target='#navbar-main')
            span.icon-bar
            span.icon-bar
            span.icon-bar
        #navbar-main.navbar-collapse.collapse
          ul.nav.navbar-nav
            li
              a(href='/about/') About

    .container
      block content

      footer
        .row
          .col-xs-12
            small &copy; Simon Holmes 2014
    script(src='/javascripts/jquery-1.11.1.min.js')
    script(src='/bootstrap/js/bootstrap.min.js')
```

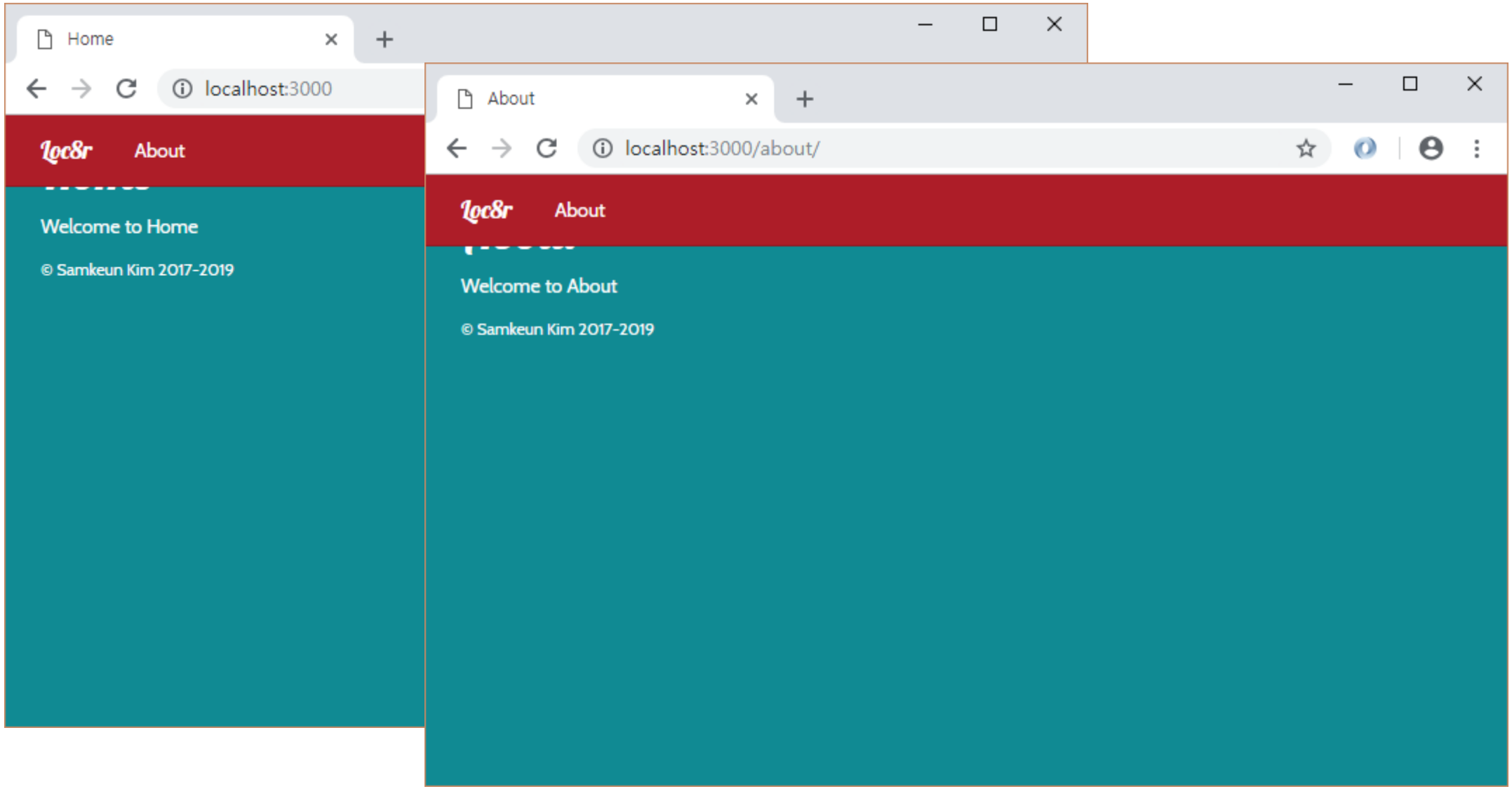
Starting layout with fixed navigation bar

한 줄이다!

Extendable content block now wrapped in a container div

Simple copyright footer in same container as content block

실습과제 24-2


Home
localhost:3000
Loc8r
About
Welcome to Home
© Samkeun Kim 2017-2019
About
localhost:3000/about/
Loc8r
About
Welcome to About
© Samkeun Kim 2017-2019

Building a template

템플릿을 만들 때 가장 적절한 템플릿부터 시작하자.

- **Loc8r**를 시작하기에 좋은 곳은 **홈페이지**이다; 이것부터 자세하게 살펴보자.

DEFINING A LAYOUT

Homepage의 주요 목적은 location 리스트를 보여주는 것이다.

각 location에는 이름, 주소, 거리, 사용자 등급과 시설 목록이 있어야 한다.

또한 페이지에 헤더를 추가하여, 사용자가 처음 방문했을 때 그들이 보고 있는 것이 무엇인지를 알 수 있도록 약간의 텍스트를 두자.

레이아웃을 스케치하는 것이 유용하다.

- ✓ 그림 4.8은 **Loc8r**의 홈페이지에 대해 스케치한 것을 보여준다.
- ✓ 데스크탑과 폰의 두 가지 레이아웃이 있다:

Bootstrap이 할 수 있는 일과 마음 속에서 어떻게 작동하는지에 대한 이해를 바탕으로 이 시점에서 둘의 차이를 구분하는 것은 충분한 가치가 있다.

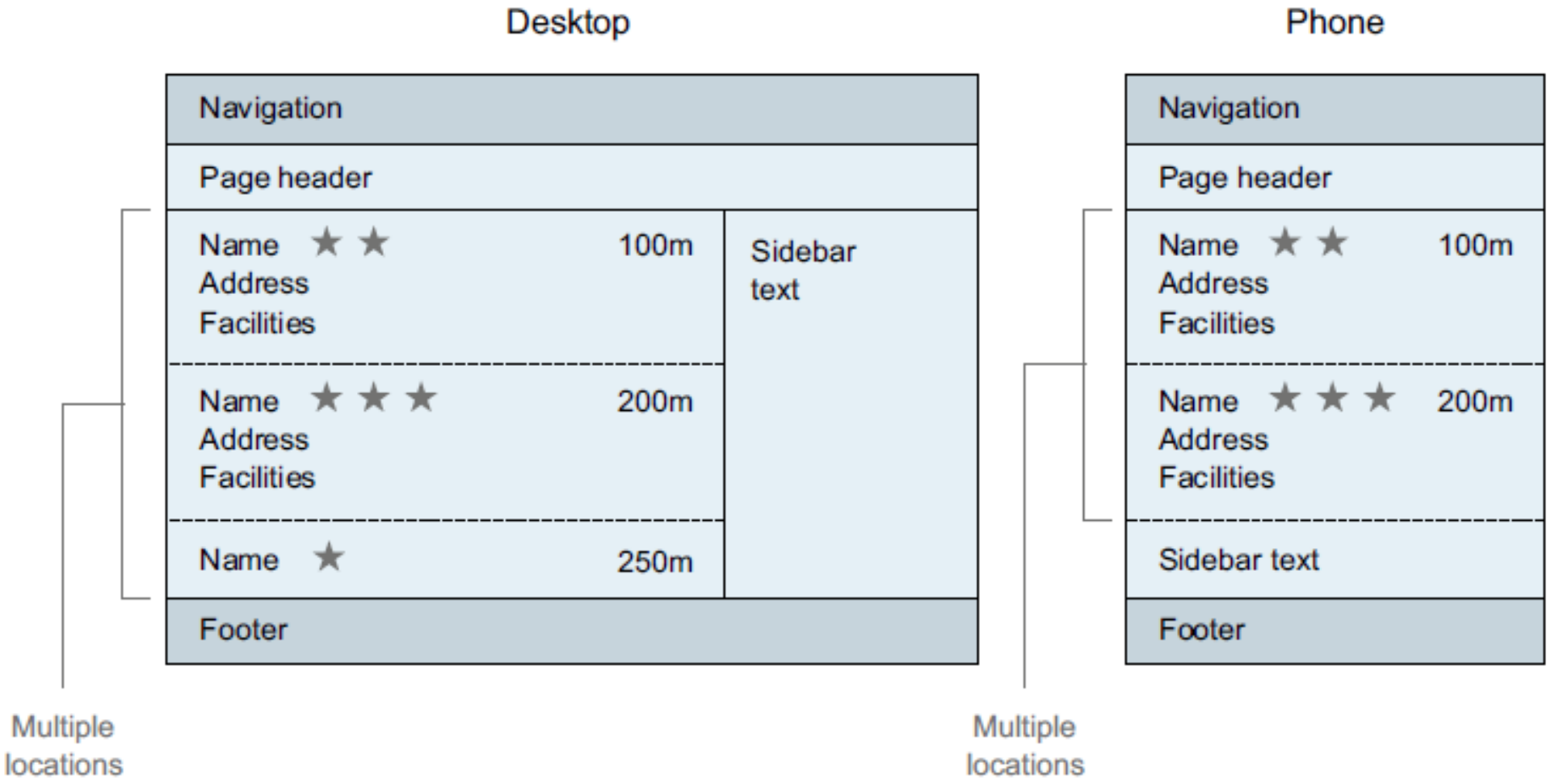


Figure 4.8 Desktop and mobile layout sketches for the homepage. Sketching out the layouts for a page can give you a quick idea of what you're going to build, without getting distracted by the intricacies of Photoshop or technicalities of code.

SETTING UP THE VIEW AND THE CONTROLLER

Step 1: 새로운 view 파일을 생성하여 controller에 연결한다.

app_server/views 폴더에서 `index.pug` 뷰의 내용을 **복사**해서, 같은 폴더에 `locations-list.pug` 파일명으로 저장한다.

Step 2: homepage의 컨트롤러에게 새로운 뷰를 사용하고 싶다고 말해 준다.

홈페이지의 컨트롤러 => app_server/controllers의 `locations.js` 파일

homelist 컨트롤러에 의해 호출되는 뷰를 변경하려면 아래처럼 수정한다:

```
module.exports.homelist = function(req, res){
  res.render('locations-list', { title: 'Home' });
};
```

이제 뷰 템플릿을 구축해보자.

CODING THE TEMPLATE: PAGE LAYOUT

레이아웃 파일 확장:

navigation bar, footer => 이미 확장함.

page header, **main area for the list**, **sidebar** => 새로 추가 필요.

locations-list.pug

```
#banner.page-header
  .row
    .col-lg-6
      h1 Loc8r
       small  Find places to work with wifi near you!
.row
  .col-xs-12.col-sm-8
    p List area.
  .col-xs-12.col-sm-4
    p.lead Loc8r helps you find places to work when out and about.
```

Page header that fills entire width, containing a column that limits text width to 6 columns on large screens for readability

Container for list of locations, spanning all 12 columns on extra-small devices and 8 columns on small devices and larger

Container for secondary or sidebar information, spanning all 12 columns on extra-small devices and 4 columns on small devices and larger

CODING THE TEMPLATE: LOCATIONS LIST

Homepage의 컨테이너들을 정의했다면 이제 **Main** 영역 차례다.

Page layout 스케치로부터 아이디어를 얻어 보자.

- ✓ 각 장소는 이름, 주소, 등급, 얼마나 멀리 있는지, 주요 시설을 보여줘야 한다.
- ✓ 클릭 가능한 프로토타입을 만들고 있기 때문에 현재는 모든 데이터가 템플릿에 실제로 존재해야 한다 (**hard-coded**).

다시 레이아웃을 살펴보자: **Pug**와 **Bootstrap**을 이용하면 작업이 훨씬 더 쉬워진다.

아래 코드 조각은 single location에 대한 정보를 보여준다:

```
.row.list-group
  .col-xs-12.list-group-item
    h4
      a(href="/location") Starcups
      small  
        span.glyphicon.glyphicon-star
        span.glyphicon.glyphicon-star
        span.glyphicon.glyphicon-star
        span.glyphicon.glyphicon-star-empty
        span.glyphicon.glyphicon-star-empty
      span.badge.pull-right.badge-default 100m
    p.address 125 High Street, Reading, RG6 1PS
    p
      span.label.label-warning Hot drinks
       
      span.label.label-warning Food
       
      span.label.label-warning Premium wifi
       
```

Set up a Bootstrap list group and create a single item spanning full 12 columns

Name of listing and a link to location

Use Bootstrap's badge helper class to hold distance away

Use Bootstrap's glyphicons to output a star rating

Address of location

Facilities of location, output using Bootstrap's label classes

Pug와 **Bootstrap**의 결합 덕택에 상대적으로 적은 노력과 코드로 아래와 같은 Single location 페이지를 얻었다:

Figure 4.9 Onscreen rendering of a single location on the List page

CODING THE TEMPLATE: PUTTING IT TOGETHER

이제 페이지 요소들의 레이아웃, 리스트 영역의 구조, 직접 입력된 약간의 데이터를 만들었다.

모두 함께 모아 놓으면 어떤 모양인지 보자.

> **List** 페이지를 복사하여 수정해서 여러 개의 location들이 표시되도록 하면 브라우저의 레이아웃을 더 잘 이해할 수 있다.

Single location을 포함한 코드는 다음 리스트에 나와 있다:

Listing 4.6 Complete template for app_server/views/locations-list.pug

```
extends layout

block content
  #banner.page-header
    .row
      .col-lg-6
        h1 Loc8r
          small  Find places to work with wifi near you!
  .row
    .col-xs-12.col-sm-8
      .row.list-group
        .col-xs-12.list-group-item
          h4
            a(href="/location") Starcups
            small  
              span.glyphicon.glyphicon-star
              span.glyphicon.glyphicon-star
              span.glyphicon.glyphicon-star
              span.glyphicon.glyphicon-star-empty
              span.glyphicon.glyphicon-star-empty
            span.badge.pull-right.badge-default 100m
          p.address 125 High Street, Reading, RG6 1PS
          p
            span.label.label-warning Hot drinks
            |  
            span.label.label-warning Food
            |  
            span.label.label-warning Premium wifi
            |  
    .col-xs-12.col-sm-4
      p.lead Looking for wifi and a seat? Loc8r helps you find places to work
      when out and about. Perhaps with coffee, cake or a pint? Let Loc8r help
      you find the place you're looking for.
```

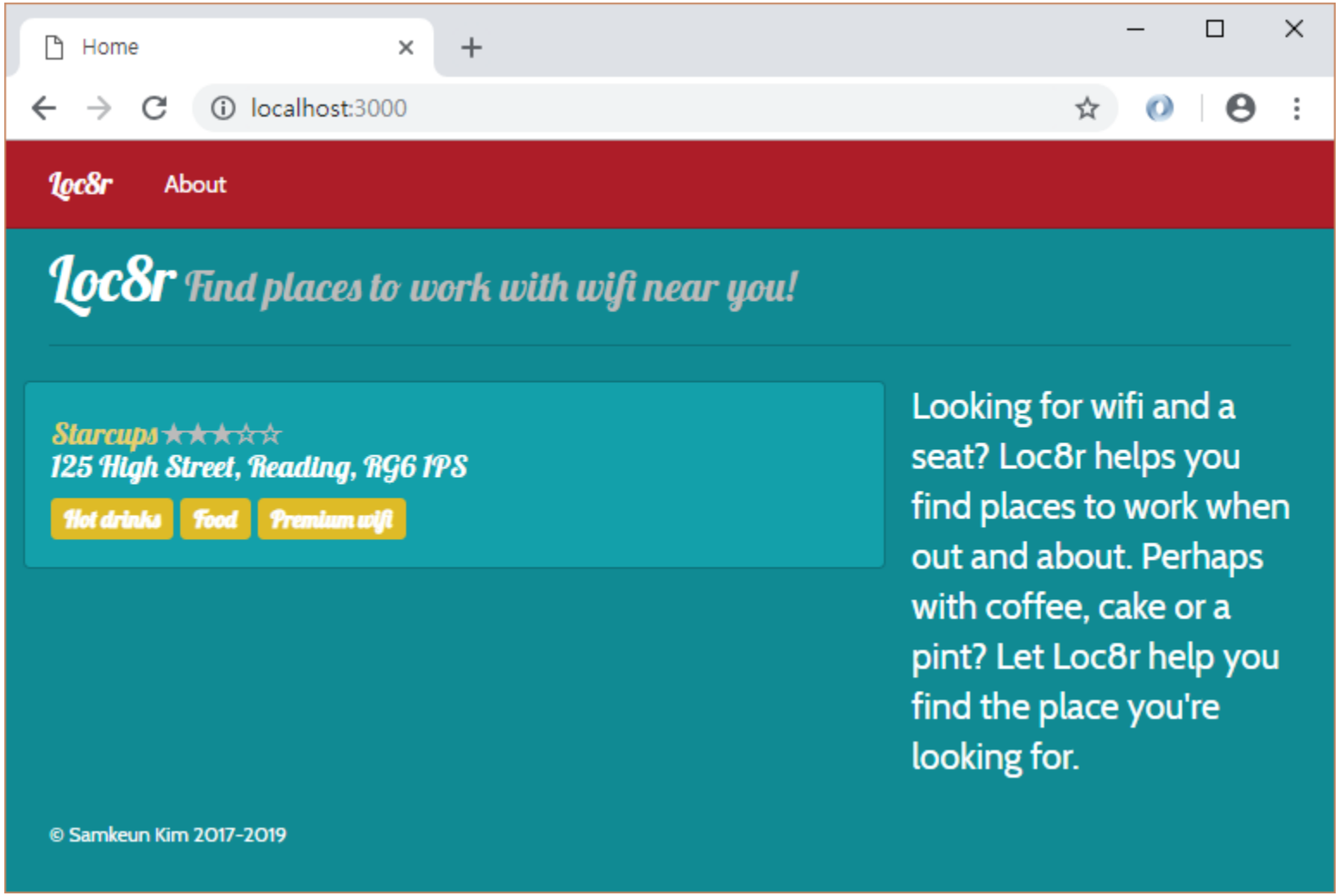
Start header area

Start responsive main listing column section

An individual listing; duplicate this section to create list of multiple items

Set up sidebar area and populate with some content

실습과제 24-3


Home
localhost:3000
Loc8r
About
Loc8r Find places to work with wifi near you!
Starcups
125 High Street, Reading, RG6 1PS
Hot drinks
Food
Premium wifi
Looking for wifi and a seat? Loc8r helps you find places to work when out and about. Perhaps with coffee, cake or a pint? Let Loc8r help you find the place you're looking for.
© Samkeun Kim 2017-2019

Adding the rest of the views

Locations 컬렉션의 **List** 페이지가 완료되었으므로, 이제 다른 페이지들을 생성하여 사용자가 클릭할 수 있는 사이트를 제공해보자.

이 섹션에서는 아래 페이지들을 추가하는 방법에 대해 설명한다:

1 — **Details**

2 — **Add Review**

3 — **About**

모든 과정에 대해 자세하게 설명하지는 않겠지만 약간의 설명, 코드 및 산출물을 제공하기로 한다.

Details page

단언컨대, 가장 중요한 페이지는 각 location에 대한 **Details** 페이지다.

이 페이지는 아래와 같은 위치 정보를 모두 표시해야 한다:

- Name
- Address
- Rating
- Opening hours
- Facilities
- Location map
- Reviews, each with
  - Rating
  - Reviewer name
  - Review date
  - Review text
- Button to add a new review
- Text to set the context of the page

꽤 많은 정보가 있다!

이 페이지는 만들고 있는 **Loc8r** 애플리케이션이 가지게 되는 가장 복잡한 단일 템플릿이다.

PREPARATION

첫 번째 단계는 이 페이지에 대한 컨트롤러가 다른 뷰를 이용하도록 수정하는 것이다.

- ✓ app_server/controllers의 `locations.js` 파일에서 **locationInfo** 컨트롤러를 찾는다.
- ✓ 아래 코드 스니펫처럼 뷰의 이름을 location-info로 변경한다:

```
module.exports.locationInfo = function(req, res){
  res.render('location-info', { title: 'Location info' });
};
```

Express가 뷰 템플릿(`location-info.pug`)을 찾을 수 없으므로, 이 시점에서는 애플리케이션이 실행되지 않는다.

이제 만들어보자.

THE VIEW

app_server/views에 새 파일을 만들고, `location-info.pug`로 저장한다.

클릭 가능한 페이지를 생성하고 데이터를 직접 하드 코딩했다.

Listing 4.7 View for the Details page, app_server/views/location-info.pug

```
extends layout

block content
  .row.page-header
    .col-lg-12
      h1 Starcups
  .row
    .col-xs-12.col-md-9
      .row
        .col-xs-12.col-sm-6
          p.rating
            span.glyphicon.glyphicon-star
            span.glyphicon.glyphicon-star
            span.glyphicon.glyphicon-star
            span.glyphicon.glyphicon-star-empty
            span.glyphicon.glyphicon-star-empty
```

Start with page header

Set up nested responsive columns needed for template

```
  p 125 High Street, Reading, RG6 1PS
  .panel.panel-primary
    .panel-heading
      h2.panel-title Opening hours
    .panel-body
      p Monday - Friday : 7:00am - 7:00pm
      p Saturday : 8:00am - 5:00pm
      p Sunday : closed
  .panel.panel-primary
    .panel-heading
      h2.panel-title Facilities
    .panel-body
      span.label.label-warning
        span.glyphicon.glyphicon-ok
        |  Hot drinks
      |  
      span.label.label-warning
        span.glyphicon.glyphicon-ok
        |  Food
      |  
      span.label.label-warning
        span.glyphicon.glyphicon-ok
        |  Premium wifi
      |  
.col-xs-12.col-sm-6.location-map
  .panel.panel-primary
    .panel-heading
      h2.panel-title Location map
```

One of several Bootstrap panel components used to define information areas, in this case opening hours

```
        .panel-body
          img.img-responsive.img-rounded(src='http://maps.googleapis.com/
maps/api/staticmap?center=51.455041,-
0.9690884&zoom=17&size=400x350&sensor=false&markers=51.455041,-
0.9690884&scale=2')
  .row
    .col-xs-12
      .panel.panel-primary.review-panel
        .panel-heading
          a.btn.btn-default.pull-right(href='/location/review/new') Add
review
          h2.panel-title Customer reviews
        .panel-body.review-container
          .row
            .review
              .well.well-sm.review-header
                span.rating
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star
                span.reviewAuthor Simon Holmes
                small.reviewTimestamp 16 July 2013
              .col-xs-12
```

Use static Google Maps image, including coordinates in the query string 51.455041,-0.9690884

Create link to Add Review page using Bootstrap's button helper class

```
img.img-responsive.img-
rounded(src='https://maps.googleapis.co
m/maps/api/staticmap?key=YOUR_API_KEY&c
enter=37.485727,126.99134&zoom=17&size=
400x350&sensor=false&markers=37.485727,
126.99134&scale=1&zoom=17&size=400x350')
```

```
                p What a great place. I can't say enough good things about
it.
          .row
            .review
              .well.well-sm.review-header
                span.rating
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star
                  span.glyphicon.glyphicon-star-empty
                  span.glyphicon.glyphicon-star-empty
                span.reviewAuthor Charlie Chaplin
                small.reviewTimestamp 16 June 2013
              .col-xs-12
                p It was okay. Coffee wasn't great, but the wifi was fast.
.col-xs-12.col-md-3
 p.lead Simon's cafe is on Loc8r because it has accessible wifi and space
to sit down with your laptop and get some work done.
 p If you've been and you like it - or if you don't - please leave a
review to help other people just like you.
```

Final responsive column for sidebar contextual information

꽤 긴 템플릿이다! 우리는 어떻게 이 부분을 축약시킬 수 있는지를 살펴볼 것이다.

현재 페이지 자체는 매우 복잡하며 많은 정보와 중첩된 응답형 컬럼들을 포함하고 있다.

이것이 전체 HTML로 작성되었다면 얼마나 길어질지 상상해보라!

ADDING A BIT OF STYLE

템플릿 OK!! 이제 CSS를 추가하여 스타일을 갖춰보자.

프로젝트를 설정할 때, 디폴트 Express 스타일 시트를 남겨 두었다.
이 파일은 `style.css`라고 하며, public/stylesheets 폴더에 있다.

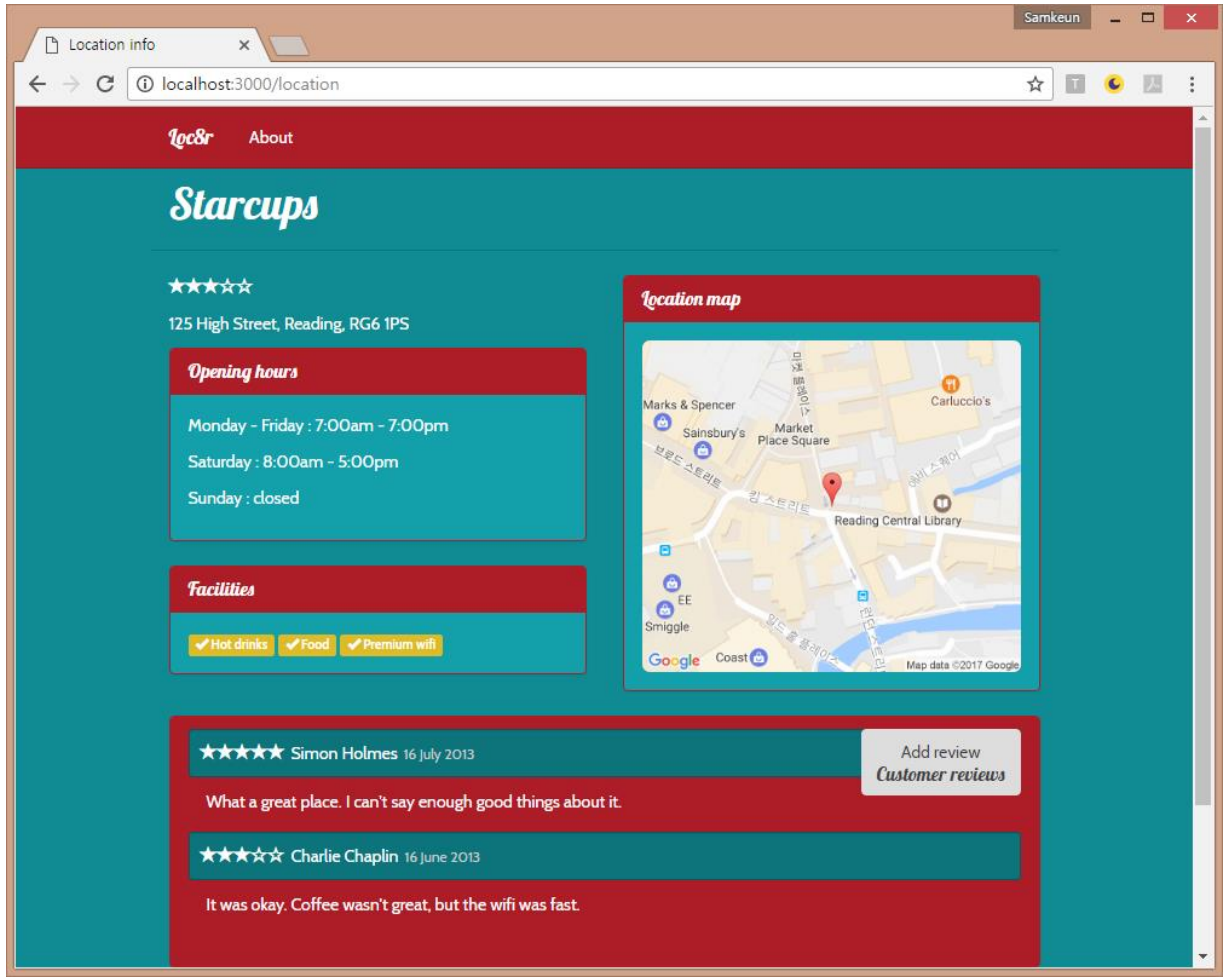
파일에 아래 코드 스니펫을 입력하고 저장하자:

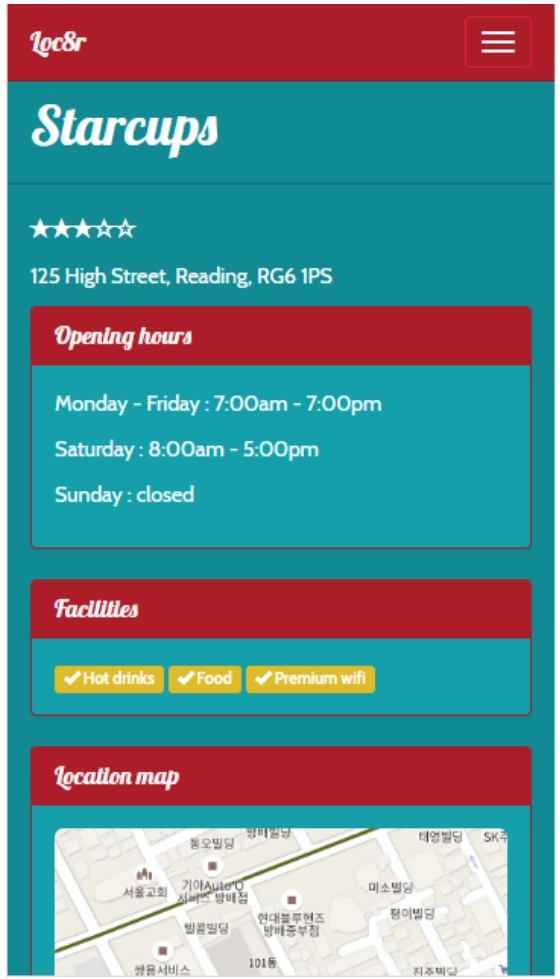
```
.review {padding-bottom: 5px;}
.panel-body.review-container {padding-top: 0;}
.review-header {margin-bottom: 10px;}
.reviewAuthor {margin: 0 5px;}
.reviewTimestamp {color: #ccc;}
```

이 정보를 저장하면 Details 페이지 레이아웃을 완성한 것이다.

localhost:3000/location으로 가서 체크해보자.

실습과제 24-4


Samkeun
Location info
localhost:3000/location
Loc8r
About
Starcups
★★★★☆
125 High Street, Reading, RG6 1PS
Opening hours
Monday - Friday : 7:00am - 7:00pm
Saturday : 8:00am - 5:00pm
Sunday : closed
Facilities
Hot drinks
Food
Premium wifi
Location map
Marks & Spencer
Sainsbury's
Market Place Square
Carluccio's
Reading Central Library
EE
Smiggle
Coast
Google
Map data ©2017 Google
★★★★★ Simon Holmes 16 July 2013
What a great place. I can't say enough good things about it.
★★★★☆ Charlie Chaplin 16 June 2013
It was okay. Coffee wasn't great, but the wifi was fast.
Add review
Customer reviews


Loc8r
Starcups
★★★★☆
125 High Street, Reading, RG6 1PS
Opening hours
Monday - Friday : 7:00am - 7:00pm
Saturday : 8:00am - 5:00pm
Sunday : closed
Facilities
Hot drinks
Food
Premium wifi
Location map

Adding Review page

이것은 아주 간단한 페이지이다.

사용자 이름과 등급 및 리뷰를 위한 몇 개의 입력 필드를 포함하는 양식을 가져야 한다.

첫 번째 단계는 새 뷰를 참조하도록 컨트롤러를 업데이트하는 것이다.

app_server/controllers/locations.js에서 아래 코드 스니펫처럼 addReview 컨트롤러가 새 뷰인 location-review-form을 사용하도록 변경한다:

```
module.exports.addReview = function(req, res){
  res.render('location-review-form', { title: 'Add review' });
};
```

두 번째 단계는 뷰 자체를 만드는 것이다.

app_server/views 폴더에서 `location-review-form.pug`라는 새 파일을 만든다.

이것은 클릭할 수 있는 프로토타입으로 설계되기 때문에 form 데이터를 포스팅하는 일은 없다.

단순히 리뷰 데이터를 보여주는 Details 페이지로 연결해주는 액션만 수행하면 된다.

그러면 폼에서 액션을 /location과 get 메소드로 설정한다.

리뷰 폼 페이지의 전체 코드는 다음 목록에 나와 있다.

Listing 4.8 View for the Add Review page, app_server/views/location-review-form.pug

```
extends layout

block content
  .row.page-header
    .col-lg-12
      h1 Review Starcups
  .row
    .col-xs-12.col-md-6
      form.form-horizontal(action="/location", method="get", role="form")
        .form-group
          label.col-xs-10.col-sm-2.control-label(for="name") Name
          .col-xs-12.col-sm-10
            input#name.form-control(name="name")
        .form-group
          label.col-xs-10.col-sm-2.control-label(for="rating") Rating
          .col-xs-12.col-sm-2
            select#rating.form-control.input-sm(name="rating")
              option 5
              option 4
              option 3
              option 2
              option 1
        .form-group
          label.col-sm-2.control-label(for="review") Review
          .col-sm-10
            textarea#review.form-control(name="review", rows="5")
        button.btn.btn-default.pull-right Add my review
    .col-xs-12.col-md-4
```

Set form action to /location and method to get

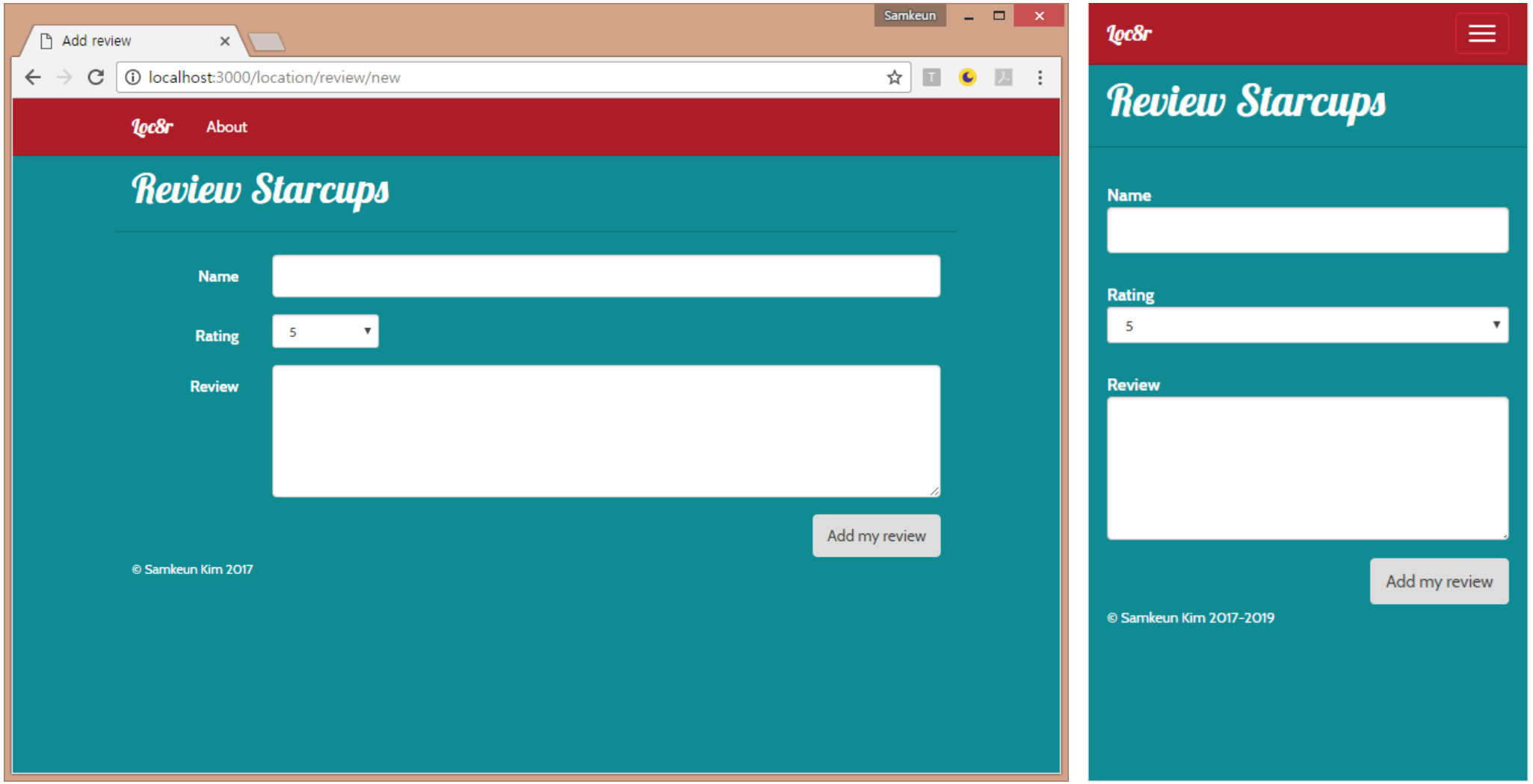
Input field for reviewer to leave name

Dropdown select box for rating 1 to 5

Submit button for form

Text area for text content of review

실습과제 24-5


Samkeun
Add review
localhost:3000/location/review/new
Loc8r
About
Review Starcups
Name
Rating
5
Review
Add my review
© Samkeun Kim 2017
Loc8r
Review Starcups
Name
Rating
5
Review
Add my review
© Samkeun Kim 2017-2019

The About page

Static 프로토타입의 최종 페이지는 About 페이지이다: 헤더와 일부 내용이 있는 페이지

레이아웃은 개인 정보 취급 방침이나 이용 약관과 같이 좀 더 진행해 가면서 다른 페이지에 유용하게 쓰일 수 있으므로, 일반적이면서 재사용 가능한 뷰로 만드는 것이 좋다.

About 페이지의 컨트롤러는 app_server/controllers의 `others.js` 파일에 있다.

about이라는 컨트롤러를 기대하면서 아래 코드 스니펫처럼 뷰 이름을 'generic-text'로 변경한다:

```
module.exports.about = function(req, res){
  res.render('generic-text', { title: 'About' });
};
```

그런 다음 app_server/views에서 `generic-text.pug` 뷰를 만든다.

이것은 매우 작은 템플릿이며 다음 리스트와 비슷하다.

Listing 4.9 View for text only pages, app_server/views/generic-text.pug

```
extends layout

block content
  #banner.page-header
    .row
      .col-md-6.col-sm-12
        h1= title
  .row
    .col-md-6.col-sm-12
      p
        | Loc8r was created to help people find places to sit down and get a
      bit of work done.
        | <br /><br />
        | Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc sed
      lorem ac nisi dignissim accumsan.
```

Use | to create lines of plain text within a <p> tag

Listing 4.9는 매우 간단한 레이아웃이다.

이 시점에서 페이지별 콘텐트를 일반 뷰에 포함하는 것에 대해 걱정할 필요가 없다; 곧 이것을 사용해서 페이지를 재사용할 수 있도록 만들 것이다.

지금은 클릭 가능한 정적 프로토타입을 완성하는 데는 충분하다.

이제 이것을 **Heroku**로 푸시하고, 사람들이 URL을 방문하여 주변을 클릭하도록 할 수 있다.

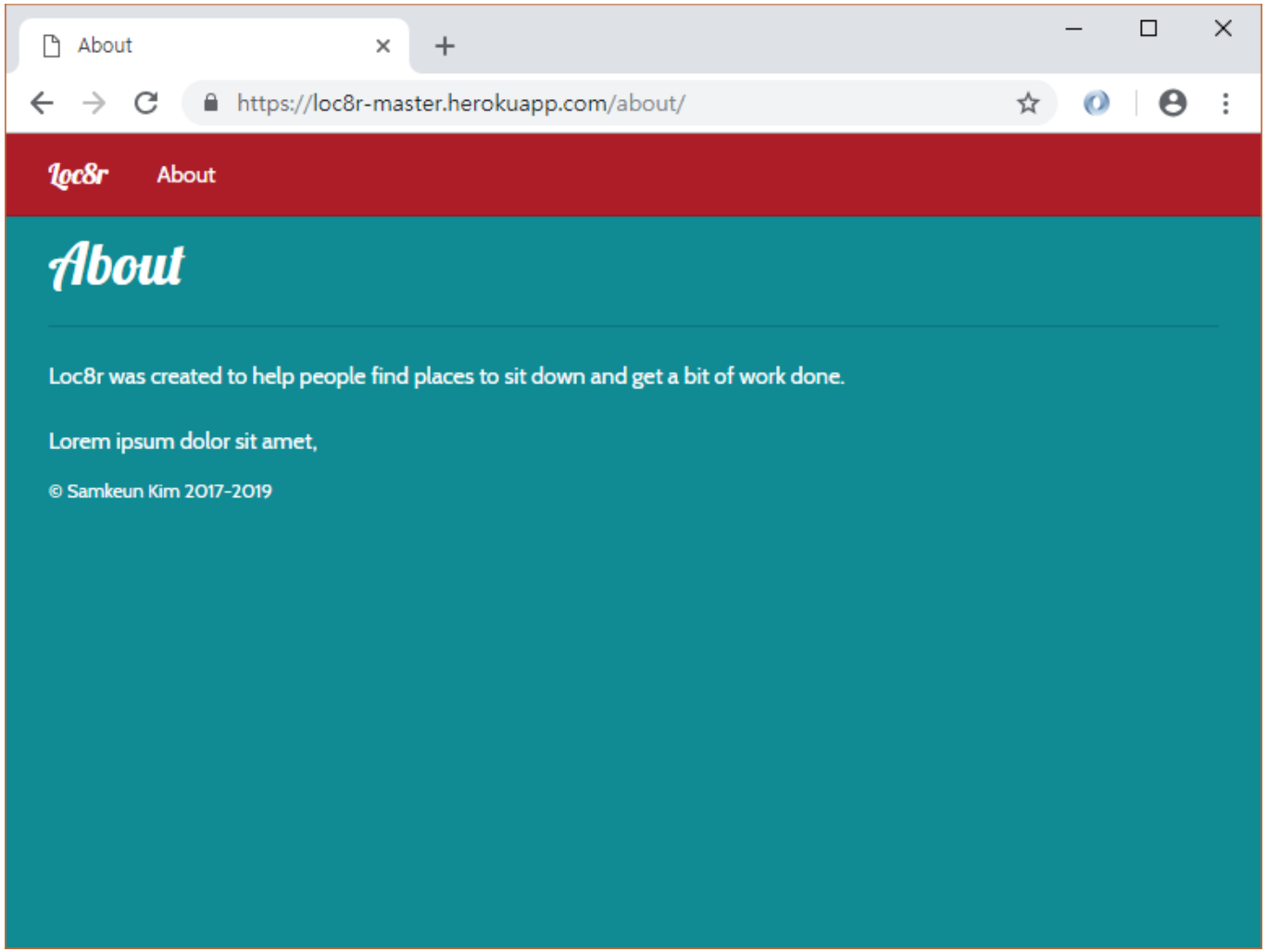
아래 코드 스니펫은 이미 **Heroku**를 설정했다고 가정하고 필요한 터미널 명령을 보여준다.

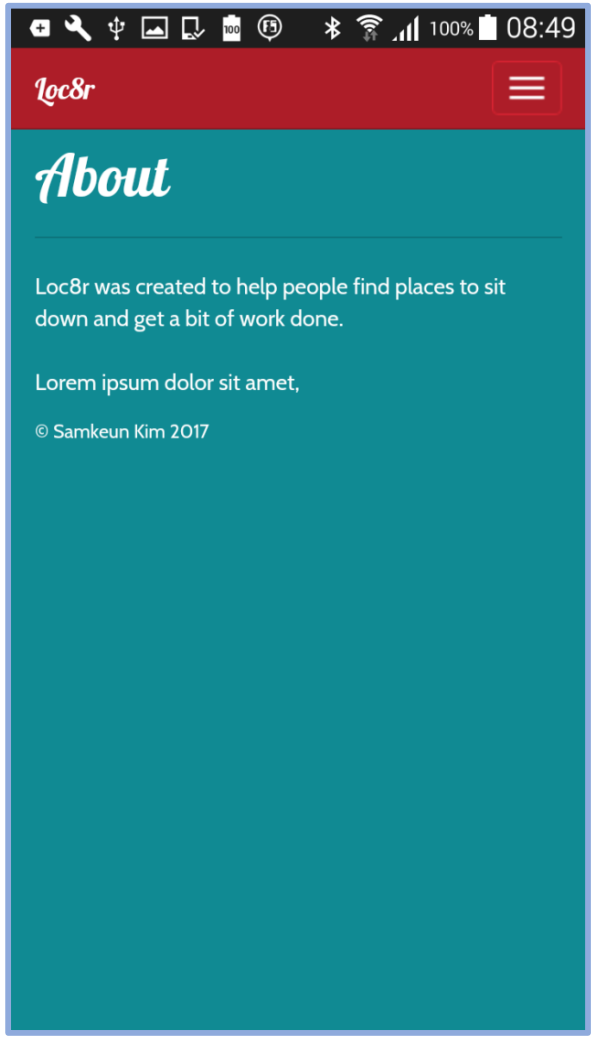
Visual Studio Code > View > Terminal:

Terminal에서 애플리케이션의 루트 폴더에 있어야 한다.

```
~~$ git init~~
$ git add .
$ git commit –m "Adding the view templates"
$ git push heroku master
$ heroku open
```

실습과제 24-6





https://loc8r-master.herokuapp.com/