How To Create Single Page Application in minutes! with AngularJs and Yii 2.0

By Hafid Mukhlasin

How To Create Single Page Application in minutes! with AngularJs and Yii 2.0

Open Your Mind

I will show You about How To Create Single Page Application (SPA) in minutes with using AngularJs and Yii Framework 2.0. This client server web application will simulate a Create Read Update Delete (CRUD) data processing. Where user interface of this application will full control by AngularJs, then data will provide by Yii Framework 2.0 as web services.

So, it will be two applications:

1. Web Client Application
2. Web Service Application

Technology Behind Scene

For web client app

I use 3 web standard language, they are HTML, JS, and CSS.

I use AngularJs as Javascript Framework and Twitter Bootstrap as CSS Framework.

For web service

I use PHP and MySQL. Yii Framework 2.0 as PHP Framework

Development tools, I use XAMPP and Notepad++ or Netbeans.

Preparing Our Gun

A. For Web Client

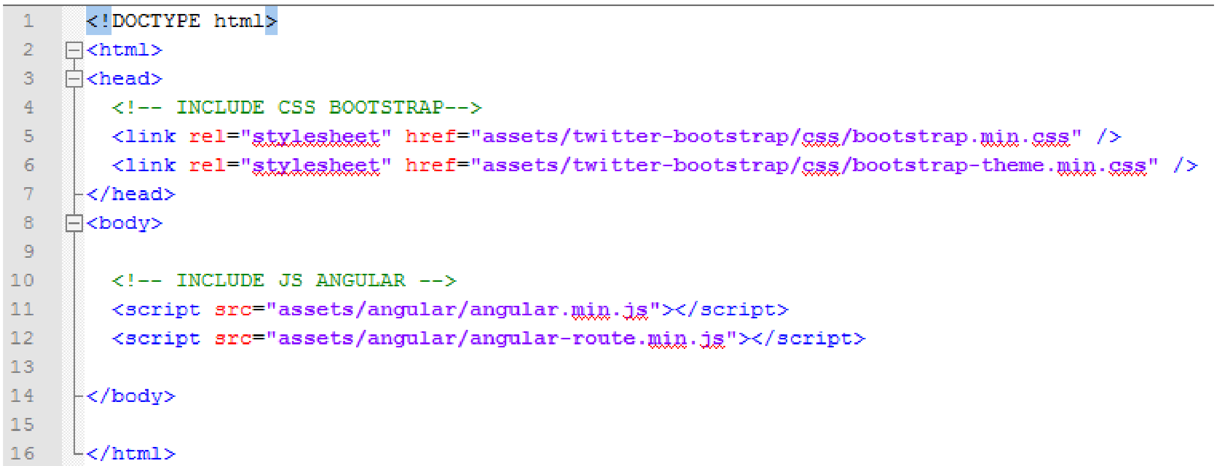
I create folder “web-client” in htdocs, then create 4 folder inside it. They are assets, controllers, models, views.

- Assets folder for core angularJs and TwitterBootstrap,
- Controllers folder for javascript controller of angularJs
- Models folders for services, connecting to Restful CRUD URL Yii 2.0
- Views folder for partial page, is like views in Yii ☺

Download http://getbootstrap.com/, and extract to folder assets

Download https://angularjs.org/, and extract to folder assets

Create file index.html in web root (web-client folder)



```
<!DOCTYPE html>
<html>
<head>
  <!-- INCLUDE CSS BOOTSTRAP-->
  <link rel="stylesheet" href="assets/twitter-bootstrap/css/bootstrap.min.css" />
  <link rel="stylesheet" href="assets/twitter-bootstrap/css/bootstrap-theme.min.css" />
</head>
<body>

  <!-- INCLUDE JS ANGULAR -->
  <script src="assets/angular/angular.min.js"></script>
  <script src="assets/angular/angular-route.min.js"></script>

</body>

</html>
```

Done

B. For Web Service

Install Yii 2.0 Basic https://github.com/yiisoft/yii2-app-basic

```
composer global require "fxp/composer-asset-plugin:1.0.0"
composer create-project --prefer-dist yiisoft/yii2-app-basic web-service
```

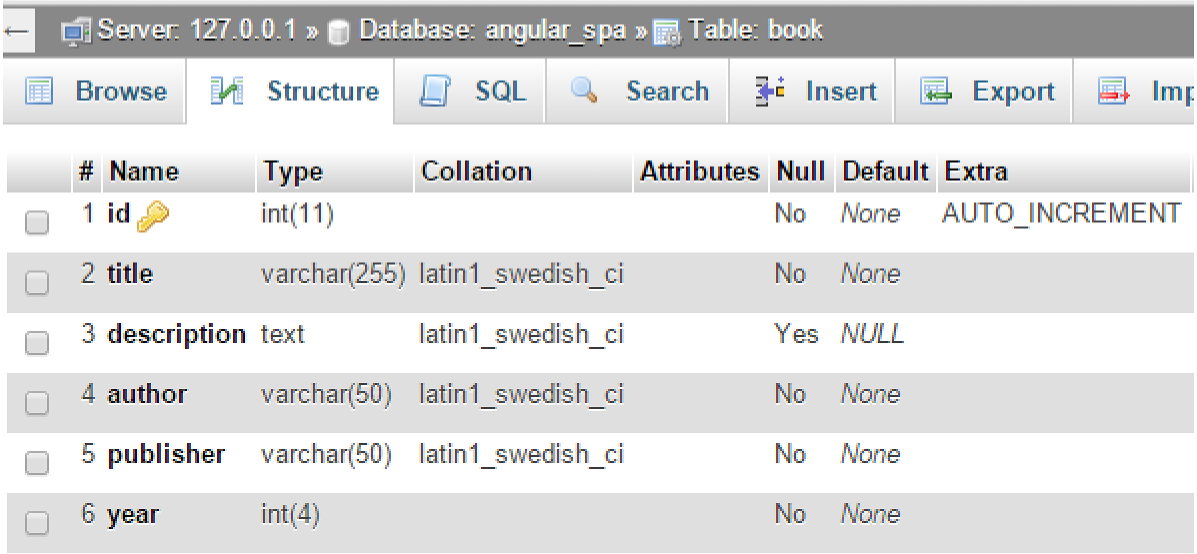
Prefered way using composer, but You can install manual checkout in http://yiiframework.com

Getting Started

A. Create Database Structure with PHPMyAdmin

Create database “angular_spa”

Create “book” table with following structure



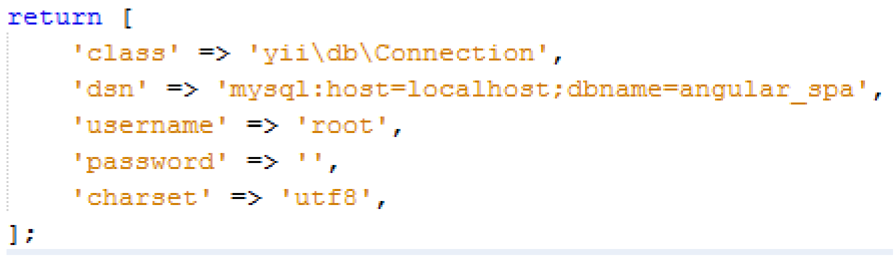
	#	Name	Type	Collation	Attributes	Null	Default	Extra
□	1	**id** 🔑	int(11)			No	*None*	AUTO_INCREMENT
□	2	**title**	varchar(255)	latin1_swedish_ci		No	*None*	
□	3	**description**	text	latin1_swedish_ci		Yes	*NULL*	
□	4	**author**	varchar(50)	latin1_swedish_ci		No	*None*	
□	5	**publisher**	varchar(50)	latin1_swedish_ci		No	*None*	
□	6	**year**	int(4)			No	*None*	

Input some datas

B. Create Web Service with Yii 2.0

1. Config Yii to connect database

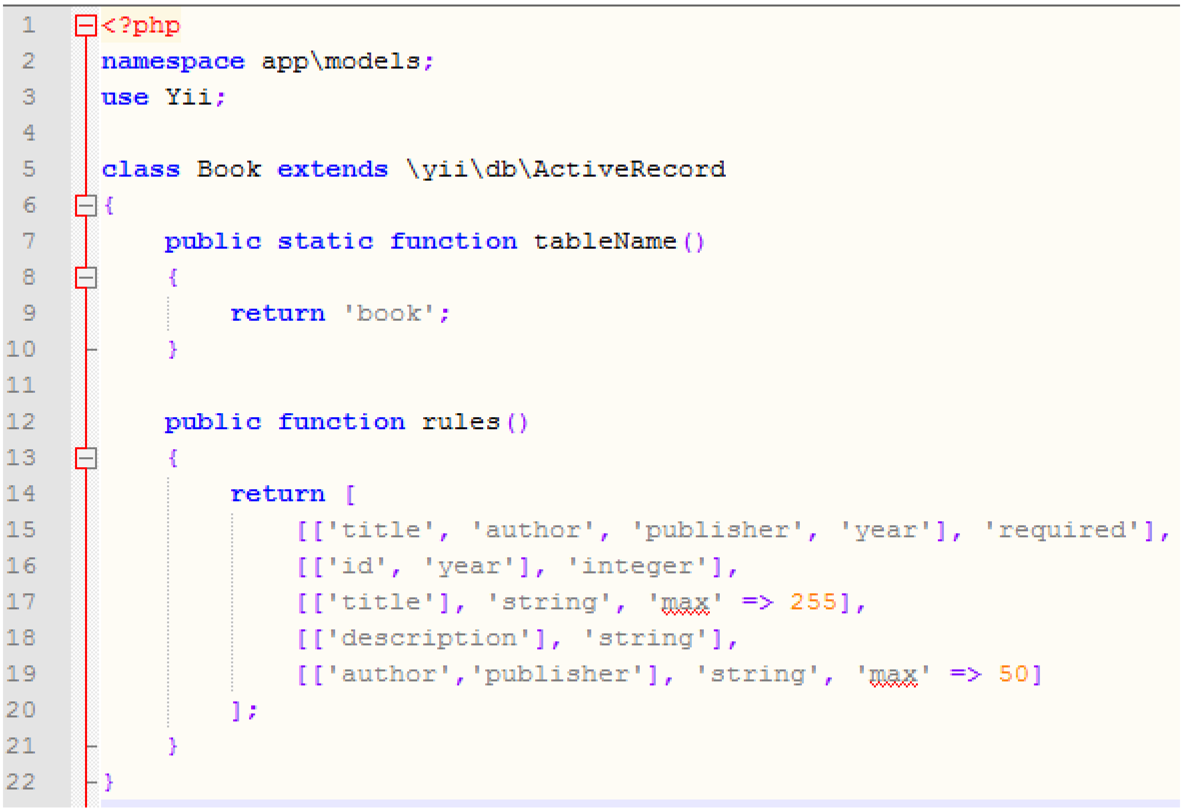
Open config/db.php in root of Yii application (web service), modify db configuration



```php
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=angular_spa',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
];
```

2. Create model of book table

Create file Book.php in folder models, You can create via Gii or manual..

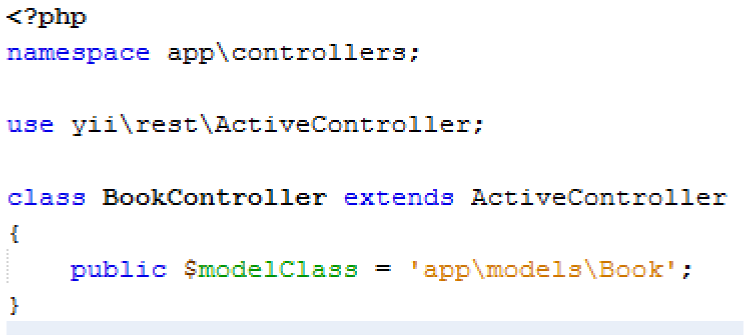


```php
<?php
namespace app\models;
use Yii;

class Book extends \yii\db\ActiveRecord
{
    public static function tableName()
    {
        return 'book';
    }

    public function rules()
    {
        return [
            [['title', 'author', 'publisher', 'year'], 'required'],
            [['id', 'year'], 'integer'],
            [['title'], 'string', 'max' => 255],
            [['description'], 'string'],
            [['author','publisher'], 'string', 'max' => 50]
        ];
    }
}
```

3. Config Yii as Restfull App

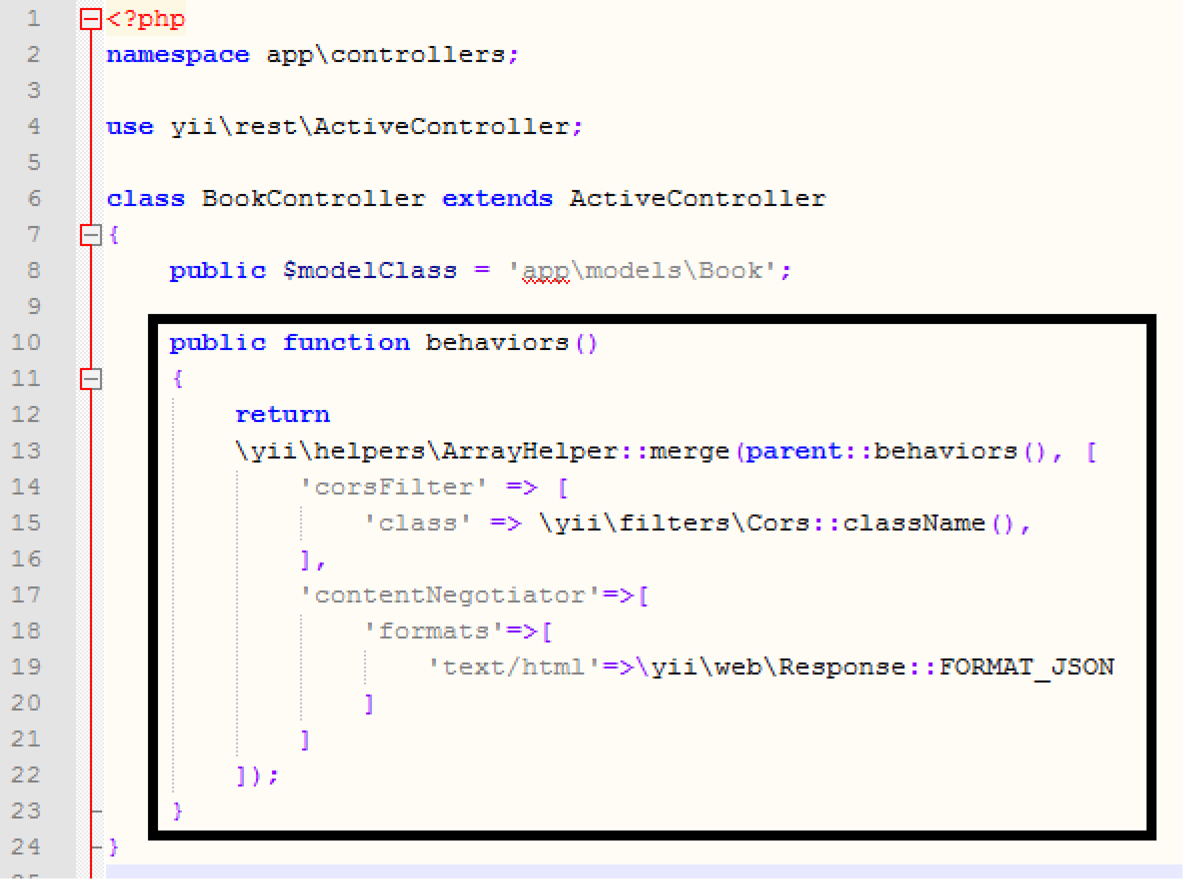
Please read from this guide https://github.com/yiisoft/yii2/blob/master/docs/guide/rest-quick-start.md, its good start, to explore restfull in Yii

Create BookController.php in folder controllers



```php
<?php
namespace app\controllers;

use yii\rest\ActiveController;

class BookController extends ActiveController
{
    public $modelClass = 'app\models\Book';
}
```

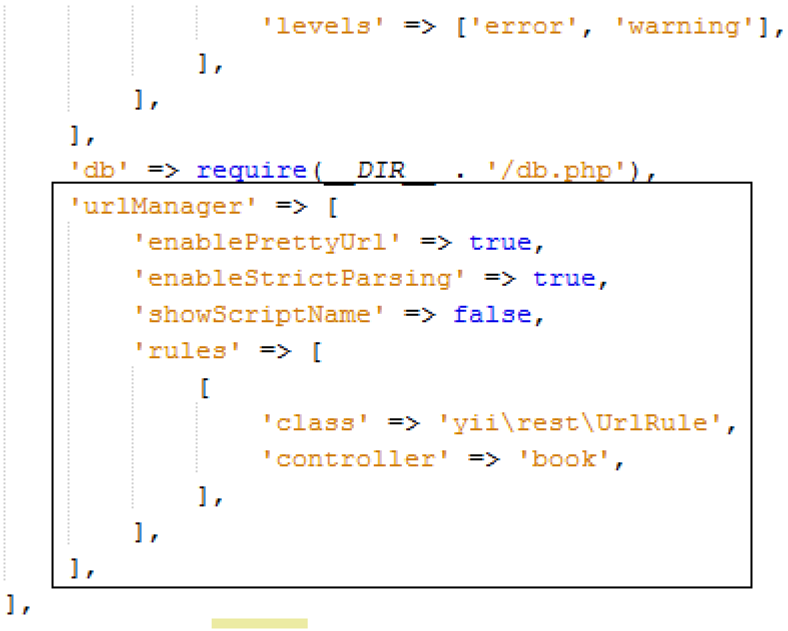
Add behavior for CORS to unblock cross origin, and set default response of REST to be JSON (optional).



```php
<?php
namespace app\controllers;

use yii\rest\ActiveController;

class BookController extends ActiveController
{
    public $modelClass = 'app\models\Book';

    public function behaviors()
    {
        return
        \yii\helpers\ArrayHelper::merge(parent::behaviors(), [
            'corsFilter' => [
                'class' => \yii\filters\Cors::className(),
            ],
            'contentNegotiator'=>[
                'formats'=>[
                    'text/html'=>\yii\web\Response::FORMAT_JSON
                ]
            ]
        ]);
    }
}
```

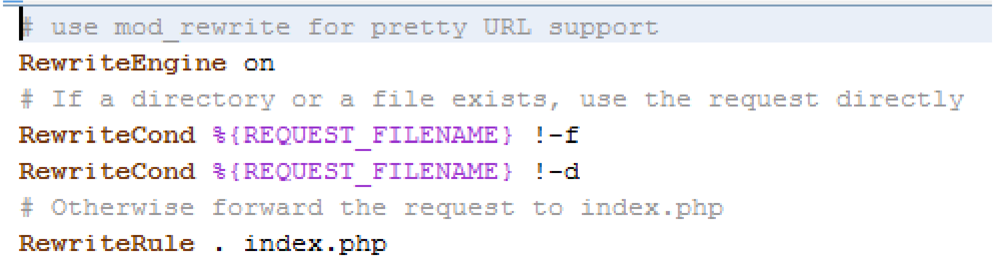
Configuring URL Rules – config\web.php

In rules You must add class yii\rest\UrlRule and define controller rest (its array), controller => [‘book’,’blabla’,’’]



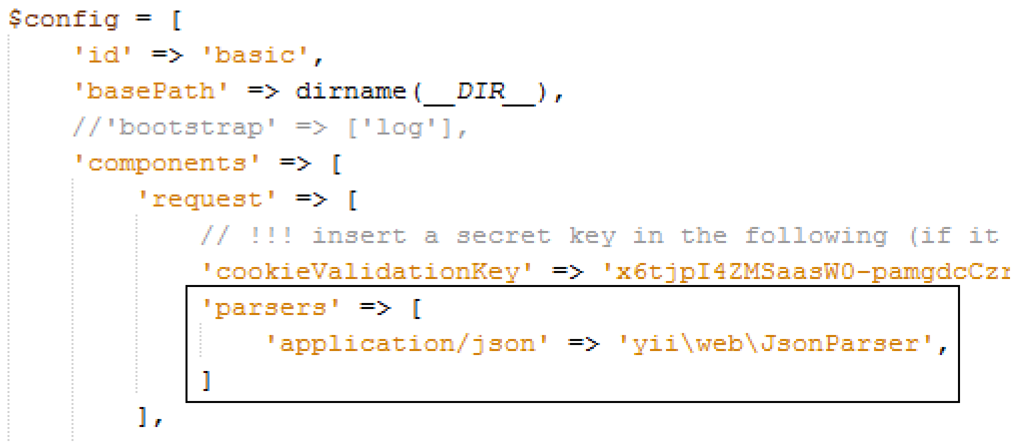
```php
                'levels' => ['error', 'warning'],
            ],
        ],
    ],
    'db' => require(__DIR__ . '/db.php'),
    'urlManager' => [
        'enablePrettyUrl' => true,
        'enableStrictParsing' => true,
        'showScriptName' => false,
        'rules' => [
            [
                'class' => 'yii\rest\UrlRule',
                'controller' => 'book',
            ],
        ],
    ],
],
```

Add .htaccess in web root (htdocs/web-service/web/), to hide index.php from url



```
# use mod_rewrite for pretty URL support
RewriteEngine on
# If a directory or a file exists, use the request directly
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
# Otherwise forward the request to index.php
RewriteRule . index.php
```

Enabling JSON Input



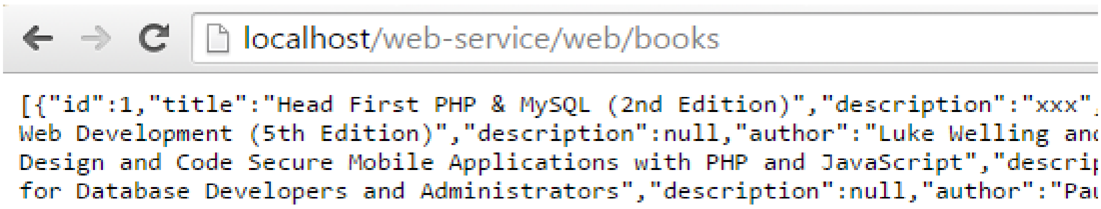
```
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    //'bootstrap' => ['log'],
    'components' => [
        'request' => [
            // !!! insert a secret key in the following (if it
            'cookieValidationKey' => 'x6tjpI4ZMSaasW0-pamgdcCzr
            'parsers' => [
                'application/json' => 'yii\web\JsonParser',
            ]
        ],
```

Trying it Out

With the above minimal amount of effort, you have already finished your task of creating the RESTful APIs for accessing the user data. The APIs you have created include:

- GET /books: list all books page by page;
- HEAD /books: show the overview information of book listing;
- POST /books: create a new book;
- GET /books/123: return the details of the user 123;
- HEAD /books/123: show the overview information of book 123;
- PATCH /books/123 and PUT /books/123: update the book 123;
- DELETE /books/123: delete the book 123;
- OPTIONS /books: show the supported verbs regarding endpoint /books;
- OPTIONS /books/123: show the supported verbs regarding endpoint /books/123.

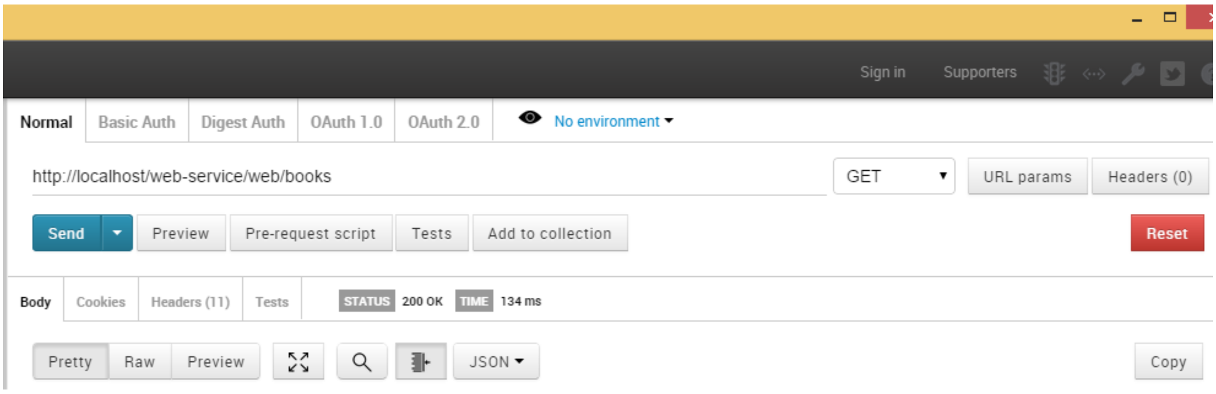
Info: Yii will automatically pluralize controller names for use in endpoints. You can configure this using the [[yii\rest\UrlRule::$pluralize]]-property.



localhost/web-service/web/books

```
[{"id":1,"title":"Head First PHP & MySQL (2nd Edition)","description":"xxx",
Web Development (5th Edition)","description":null,"author":"Luke Welling an
Design and Code Secure Mobile Applications with PHP and JavaScript","descri
for Database Developers and Administrators","description":null,"author":"Pa
```

You can testing with Postman (Chrome Extensions)



http://localhost/web-service/web/books

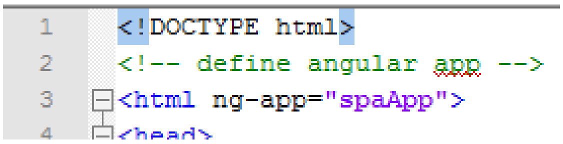
GET

STATUS 200 OK TIME 134 ms

C. Create Web Client with AngularJs + Twitter Bootstrap

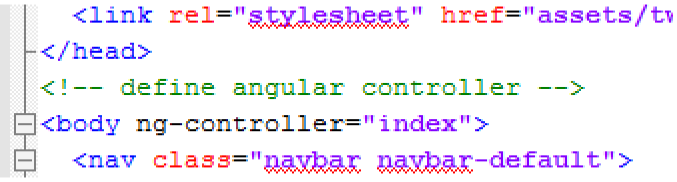
1. Modify index.html atau main layout

Attaching ng-app attribute to html tag to define angular app,



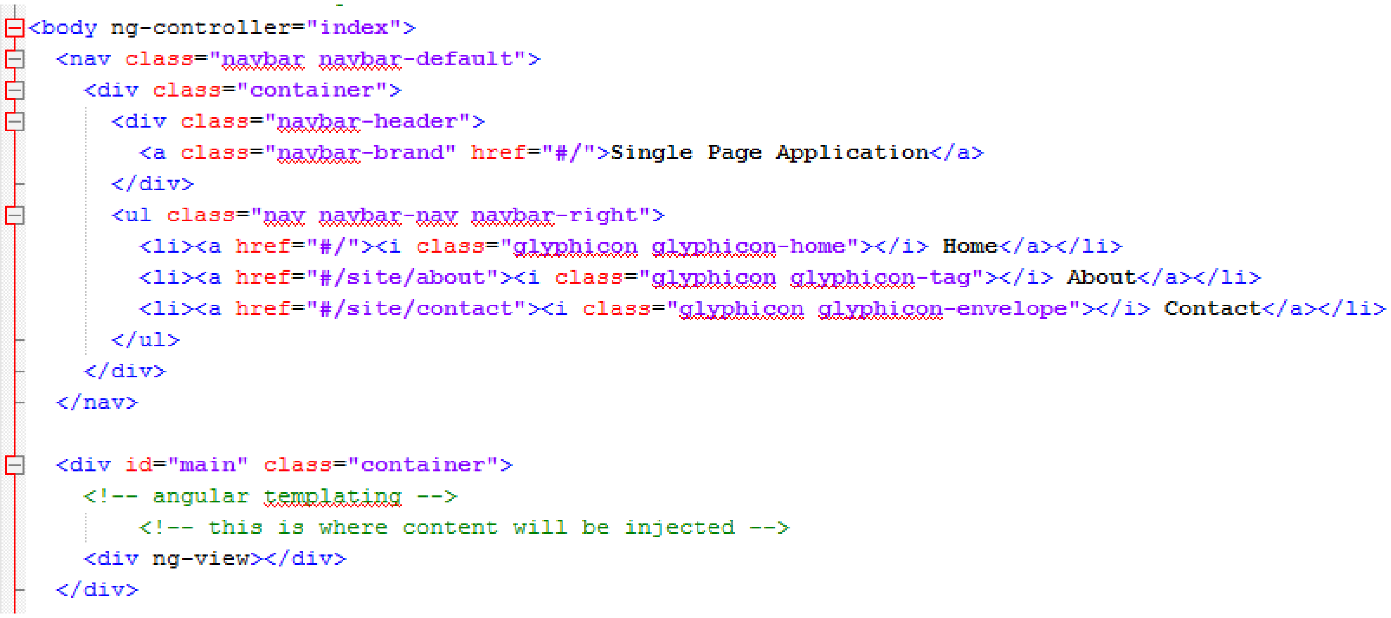
```html
<!DOCTYPE html>
<!-- define angular app -->
<html ng-app="spaApp">
<head>
```

Attaching ng-controller attribute to body tag to define default angular controller,



```html
    <link rel="stylesheet" href="assets/tw
</head>
<!-- define angular controller -->
<body ng-controller="index">
    <nav class="navbar navbar-default">
```

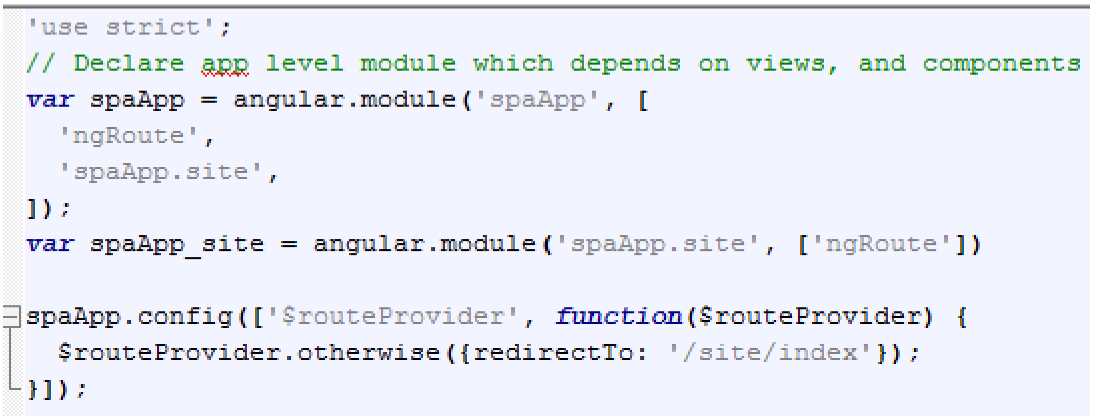
Add this content inside body tag, its are Navbar for menu, main for page container, and footer



```html
<body ng-controller="index">
    <nav class="navbar navbar-default">
        <div class="container">
            <div class="navbar-header">
                <a class="navbar-brand" href="#/">Single Page Application</a>
            </div>
            <ul class="nav navbar-nav navbar-right">
                <li><a href="#/"><i class="glyphicon glyphicon-home"></i> Home</a></li>
                <li><a href="#/site/about"><i class="glyphicon glyphicon-tag"></i> About</a></li>
                <li><a href="#/site/contact"><i class="glyphicon glyphicon-envelope"></i> Contact</a></li>
            </ul>
        </div>
    </nav>

    <div id="main" class="container">
        <!-- angular templating -->
            <!-- this is where content will be injected -->
        <div ng-view></div>
    </div>
```

Important thing is main container.. <div ng-view></div>, in this div will placed the dynamic content from other file or page view.

2. Create app.js in web root



```javascript
'use strict';
// Declare app level module which depends on views, and components
var spaApp = angular.module('spaApp', [
  'ngRoute',
  'spaApp.site',
]);
var spaApp_site = angular.module('spaApp.site', ['ngRoute'])

spaApp.config(['$routeProvider', function($routeProvider) {
  $routeProvider.otherwise({redirectTo: '/site/index'});
}]);
```

Default route is /site/index, this route will handled by sub module “spaApp.site”

3. Create file site.js in folder controller

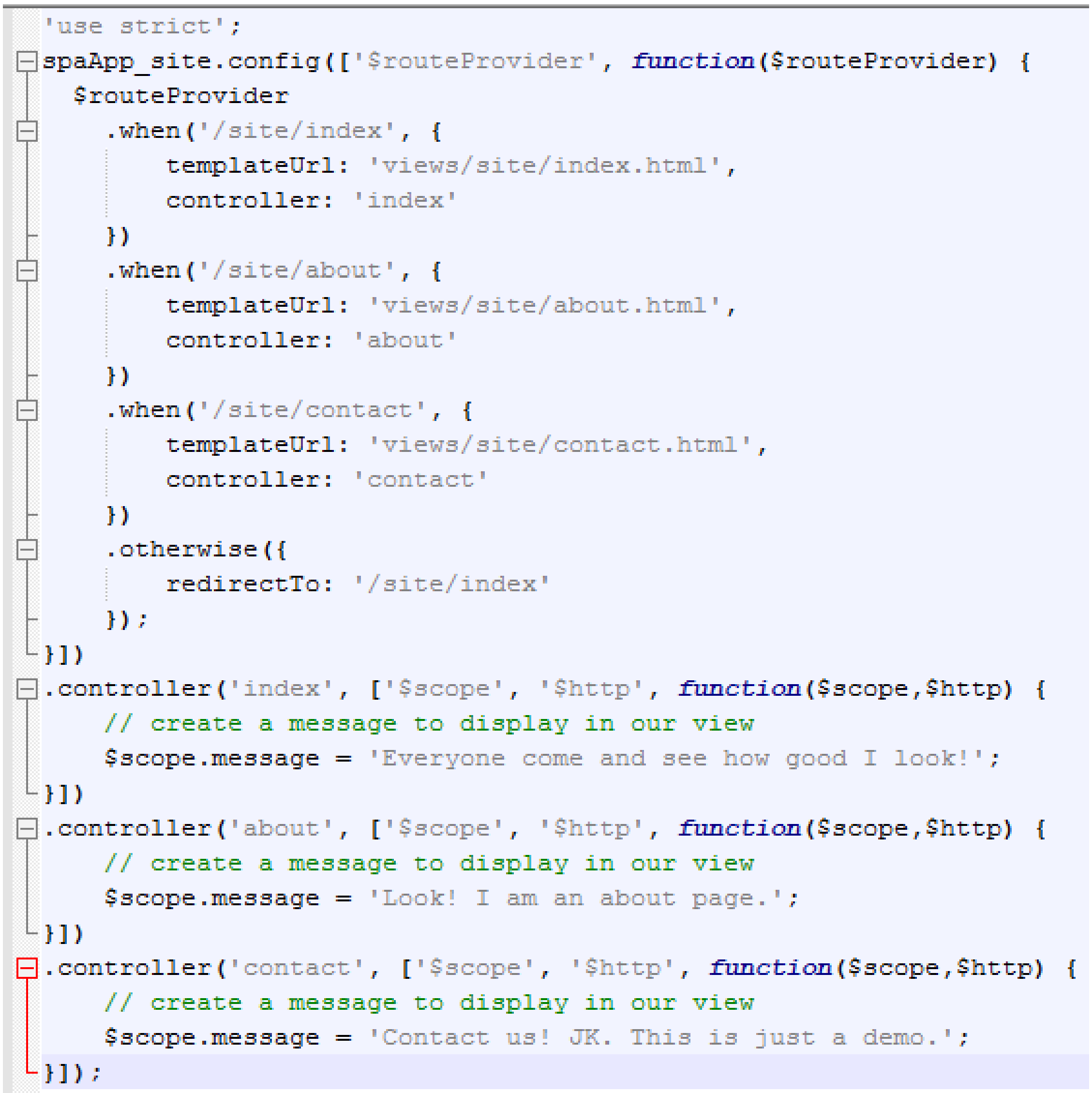
This file is controller to handle site views, maybe in Yii is like as SiteController.php

This file handle routing of angular app for sub module site. Every route have may templateUrl and controller.

TemplateUrl is other file html as partial content

Controller is name of controller that do some process, for example set variable in template.

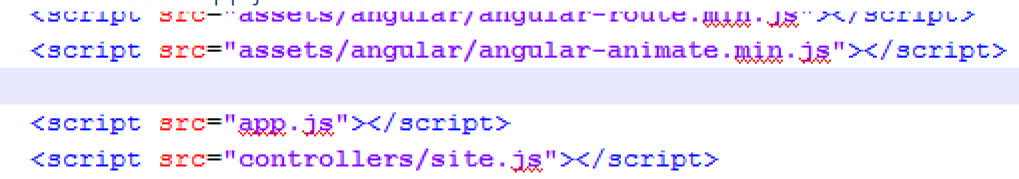
$scope.message, message is variabel in file templateUrl, let say views/site/index.html, point to {{message}}



```
'use strict';
spaApp_site.config(['$routeProvider', function($routeProvider) {
  $routeProvider
    .when('/site/index', {
        templateUrl: 'views/site/index.html',
        controller: 'index'
    })
    .when('/site/about', {
        templateUrl: 'views/site/about.html',
        controller: 'about'
    })
    .when('/site/contact', {
        templateUrl: 'views/site/contact.html',
        controller: 'contact'
    })
    .otherwise({
        redirectTo: '/site/index'
    });
}])
.controller('index', ['$scope', '$http', function($scope,$http) {
    // create a message to display in our view
    $scope.message = 'Everyone come and see how good I look!';
}])
.controller('about', ['$scope', '$http', function($scope,$http) {
    // create a message to display in our view
    $scope.message = 'Look! I am an about page.';
}])
.controller('contact', ['$scope', '$http', function($scope,$http) {
    // create a message to display in our view
    $scope.message = 'Contact us! JK. This is just a demo.';
}]);
```

If You Yiiers, Sub module in angular maybe is like as Controller in Yii, and .controller in angular is like as function of controllers in Yii.

4. Include app.js in index.html

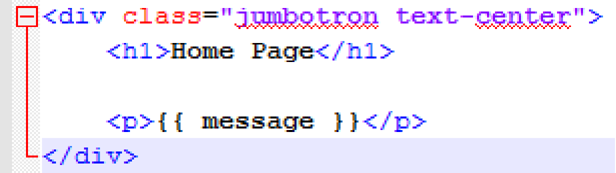


```
<script src="assets/angular/angular-route.min.js"></script>
<script src="assets/angular/angular-animate.min.js"></script>

<script src="app.js"></script>
<script src="controllers/site.js"></script>
```

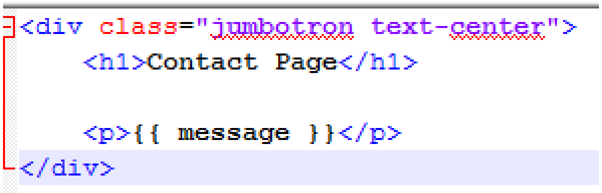
Okey, after it then create views

5. Create views/site/index.html, views/site/contact.html, views/site/about.html

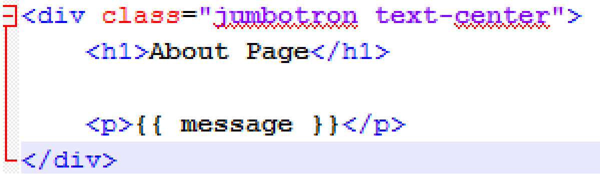
This is only example



```
<div class="jumbotron text-center">
    <h1>Home Page</h1>

    <p>{{ message }}</p>
</div>
```

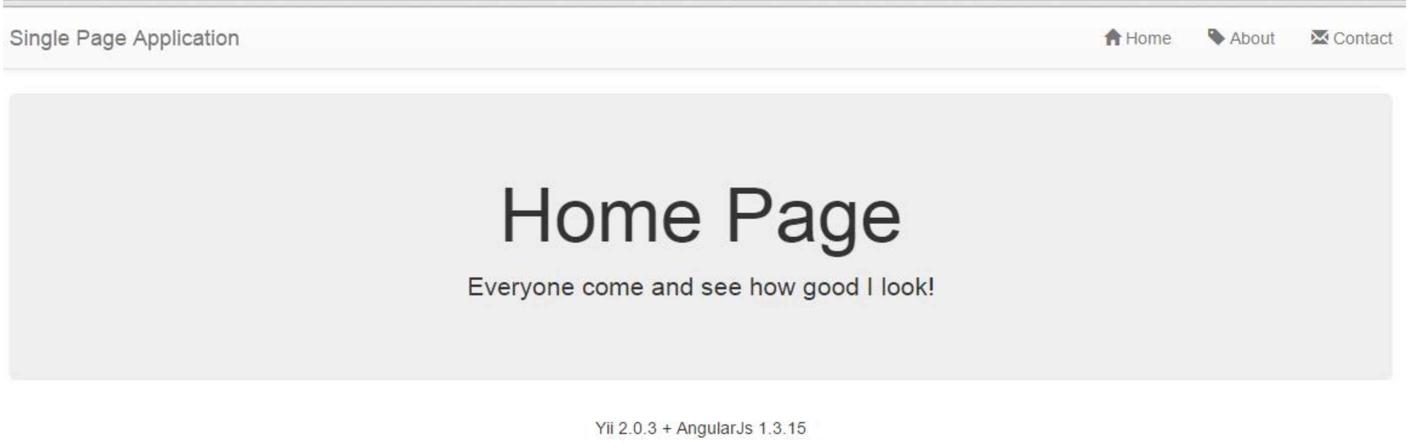


```
<div class="jumbotron text-center">
    <h1>Contact Page</h1>

    <p>{{ message }}</p>
</div>
```



```
<div class="jumbotron text-center">
    <h1>About Page</h1>

    <p>{{ message }}</p>
</div>
```

6. Testing Your Angular App



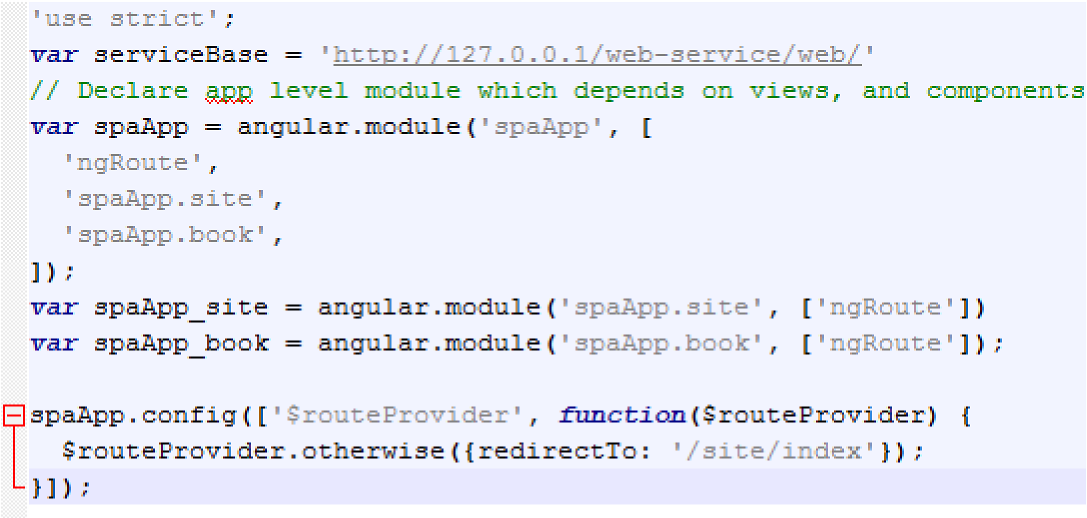
Home Page

Everyone come and see how good I look!

Yii 2.0.3 + AngularJs 1.3.15

7. Modify app.js

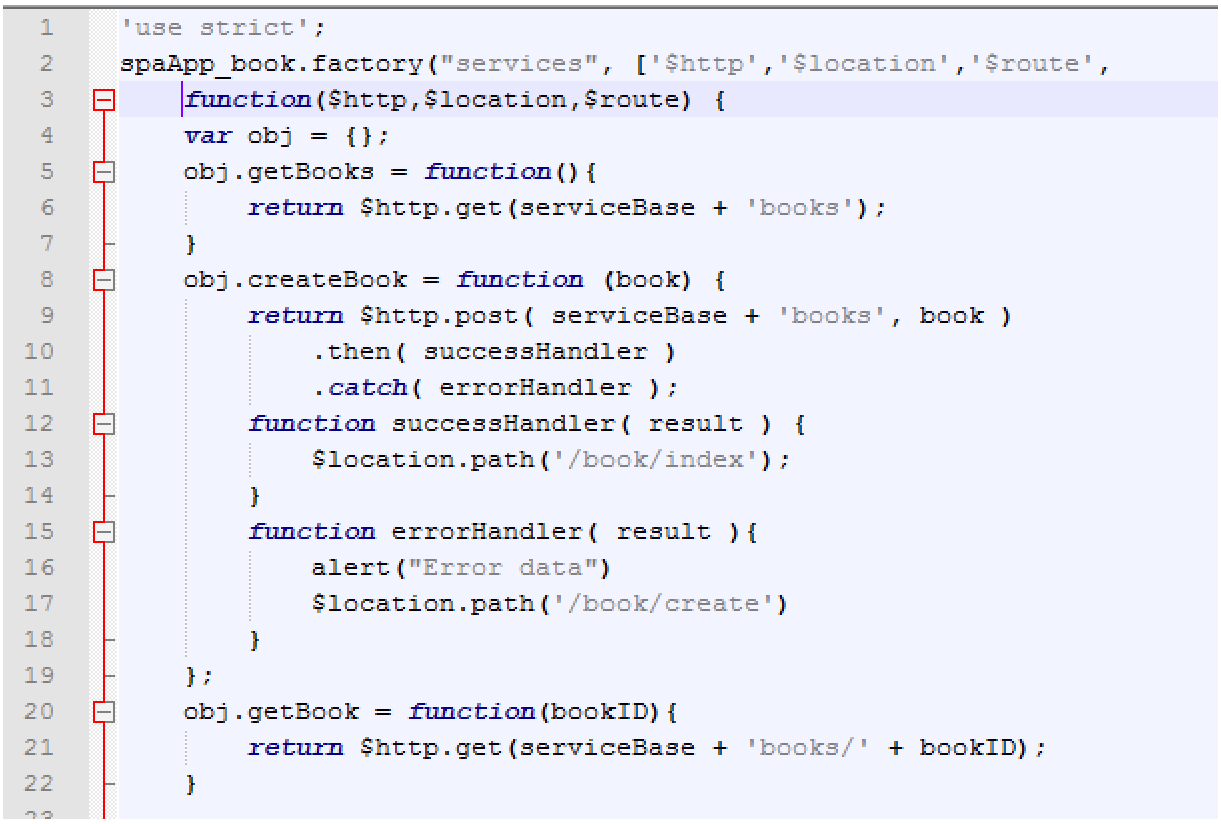
Add js global varabel serviceBase that refer to Your Yii 2 web service, then add sub module spaApp.book



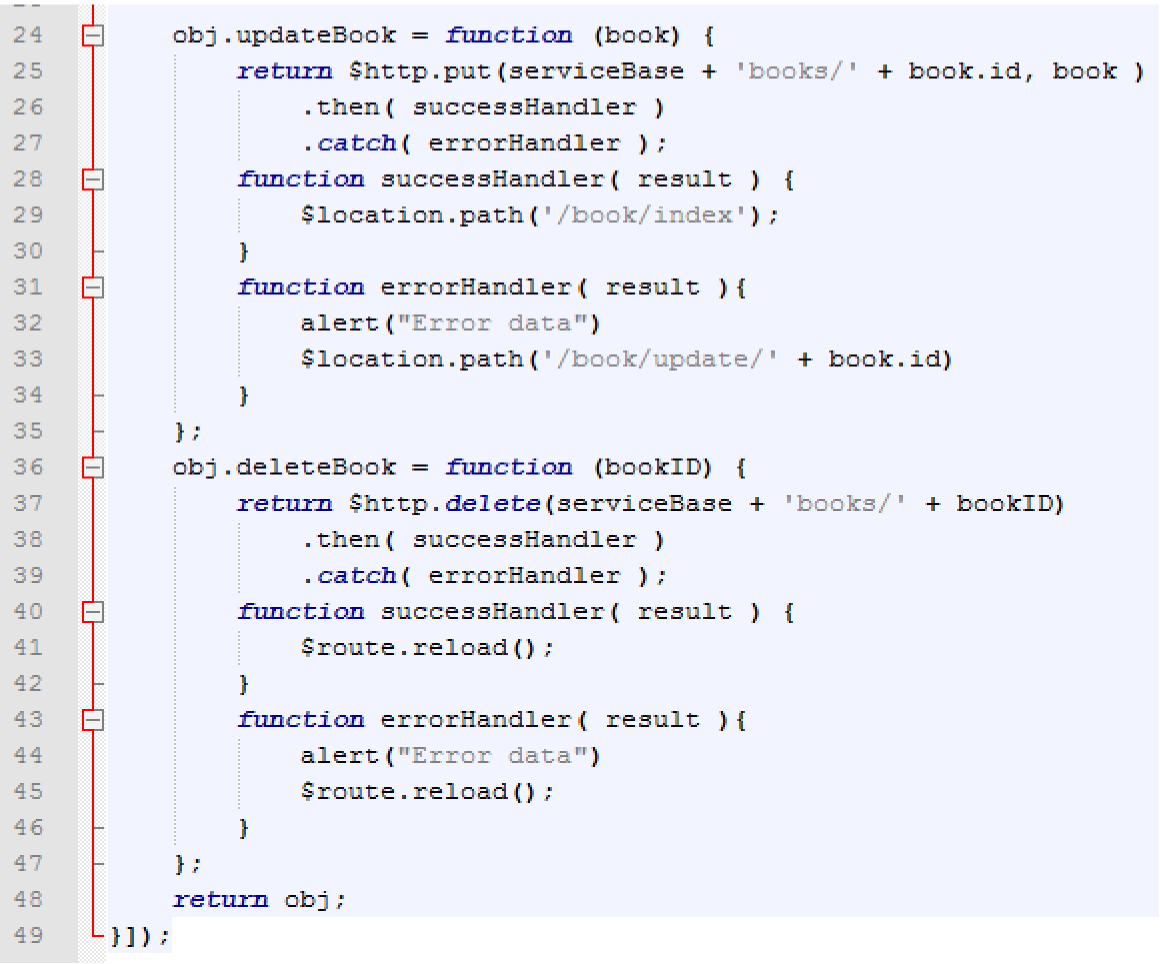
```
'use strict';
var serviceBase = 'http://127.0.0.1/web-service/web/'
// Declare app level module which depends on views, and components
var spaApp = angular.module('spaApp', [
  'ngRoute',
  'spaApp.site',
  'spaApp.book',
]);
var spaApp_site = angular.module('spaApp.site', ['ngRoute'])
var spaApp_book = angular.module('spaApp.book', ['ngRoute']);

spaApp.config(['$routeProvider', function($routeProvider) {
  $routeProvider.otherwise({redirectTo: '/site/index'});
}]);
```

8. Create file book.js in folder models

book.js will handle CRUD data from Rest Yii 2. I force You understand it as models in Yii 2 ☺



```
'use strict';
spaApp_book.factory("services", ['$http','$location','$route',
    function($http,$location,$route) {
    var obj = {};
    obj.getBooks = function(){
        return $http.get(serviceBase + 'books');
    }
    obj.createBook = function (book) {
        return $http.post( serviceBase + 'books', book )
            .then( successHandler )
            .catch( errorHandler );
        function successHandler( result ) {
            $location.path('/book/index');
        }
        function errorHandler( result ){
            alert("Error data")
            $location.path('/book/create')
        }
    };
    obj.getBook = function(bookID){
        return $http.get(serviceBase + 'books/' + bookID);
    }
```



```
    obj.updateBook = function (book) {
        return $http.put(serviceBase + 'books/' + book.id, book )
            .then( successHandler )
            .catch( errorHandler );
        function successHandler( result ) {
            $location.path('/book/index');
        }
        function errorHandler( result ){
            alert("Error data")
            $location.path('/book/update/' + book.id)
        }
    };
    obj.deleteBook = function (bookID) {
        return $http.delete(serviceBase + 'books/' + bookID)
            .then( successHandler )
            .catch( errorHandler );
        function successHandler( result ) {
            $route.reload();
        }
        function errorHandler( result ){
            alert("Error data")
            $route.reload();
        }
    };
    return obj;
}]);
```

To understand it, so simple.. focus to function of objects: obj,getBooks, obj.createBook, etc. And in every function, there are process to shoot to Rest Yii.. for example

```
obj.getBooks = function(){
    return $http.get(serviceBase + 'books');
```

}

For get list of the book use method GET, see in https://github.com/yiisoft/yii2/blob/master/docs/guide/rest-quick-start.md,

obj.createBook = function (book) {

return $http.post(serviceBase + 'books', book)

create book use method POST

obj.updateBook = function (book) {

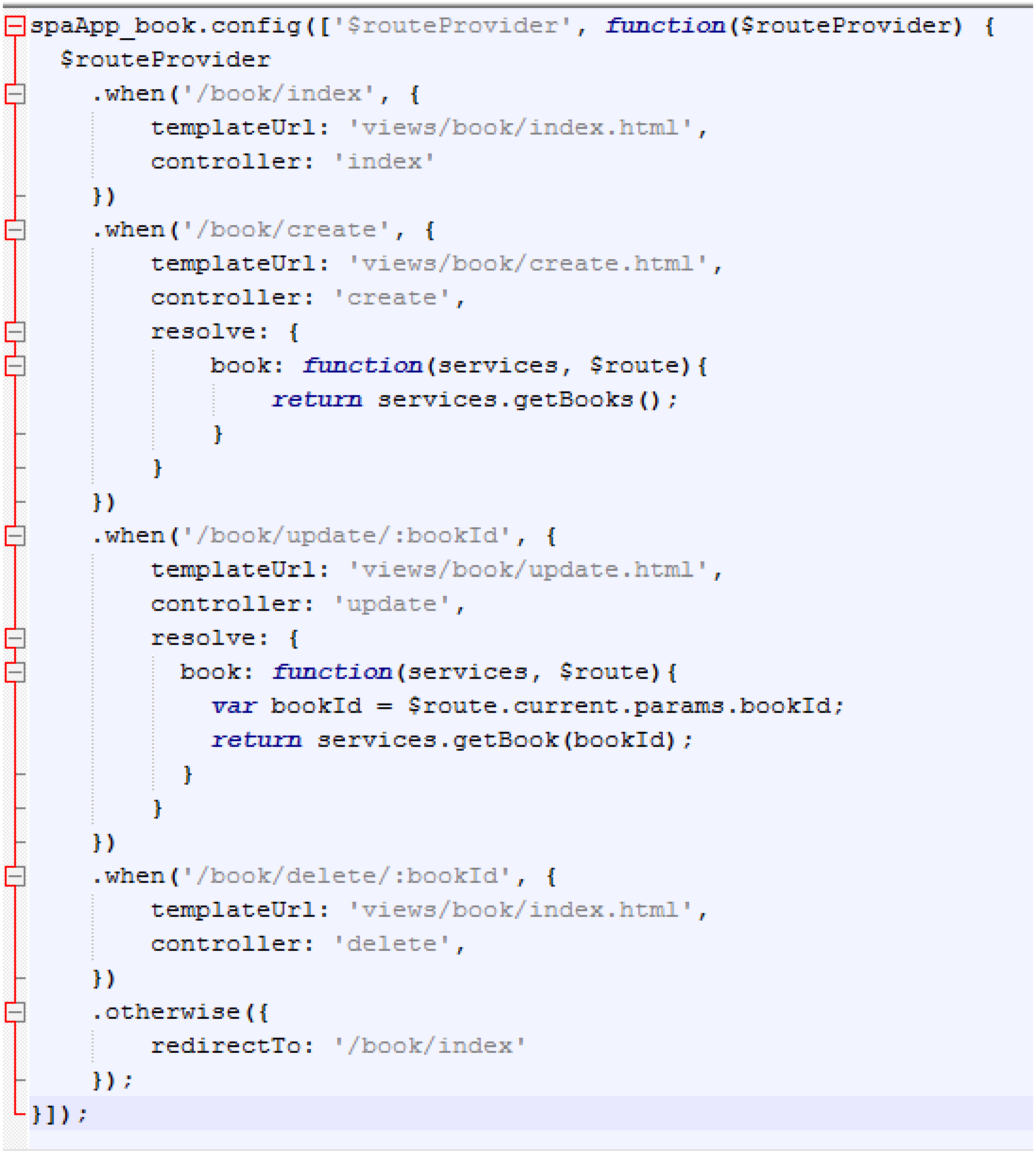
return $http.put(serviceBase + 'books/' + book.id, book)

update book use PUT method..

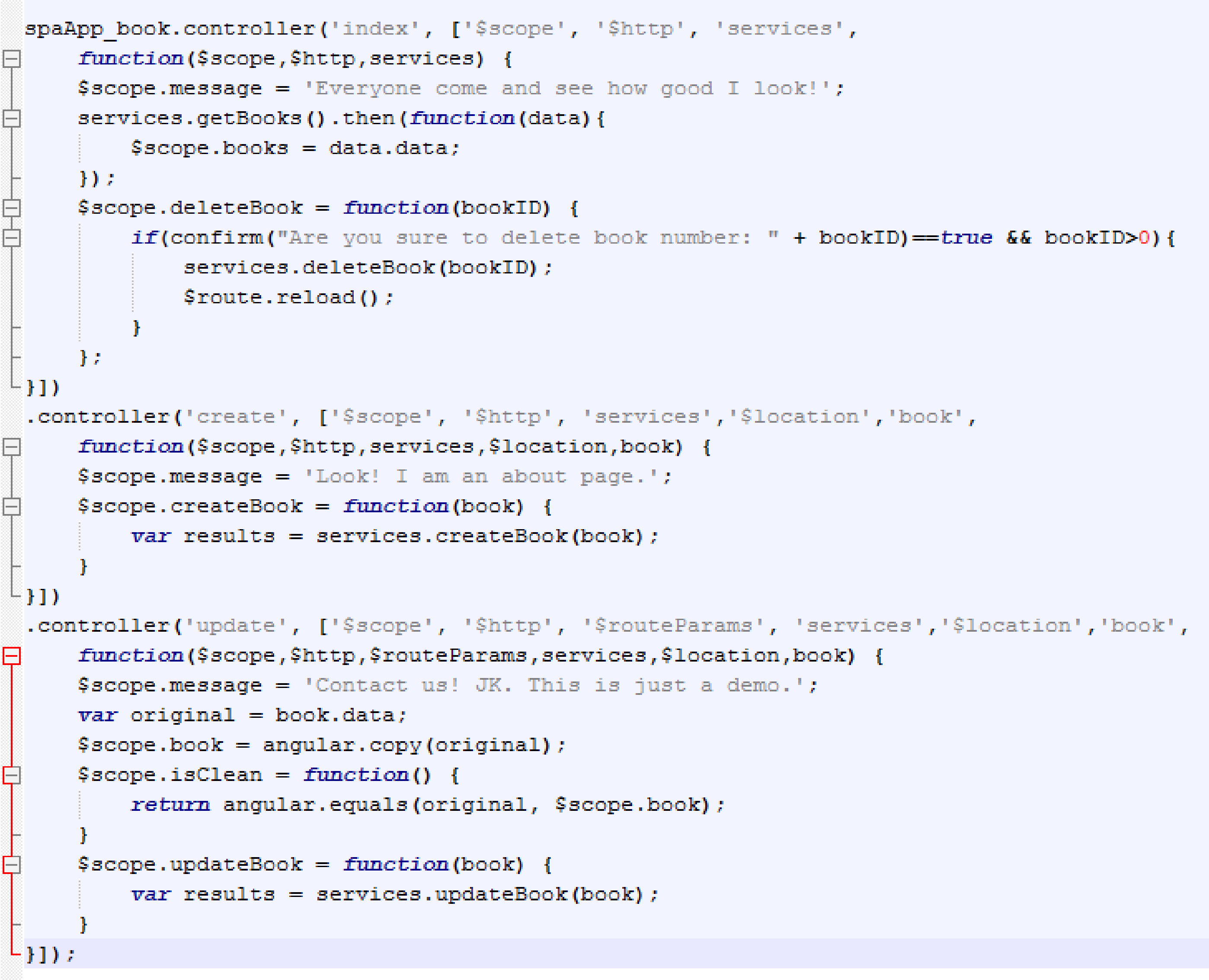
Okey, sos simple, You understand?

9. Create file book.js in folder controller

This file is controller to handle book views, maybe in Yii is like as BookController.php

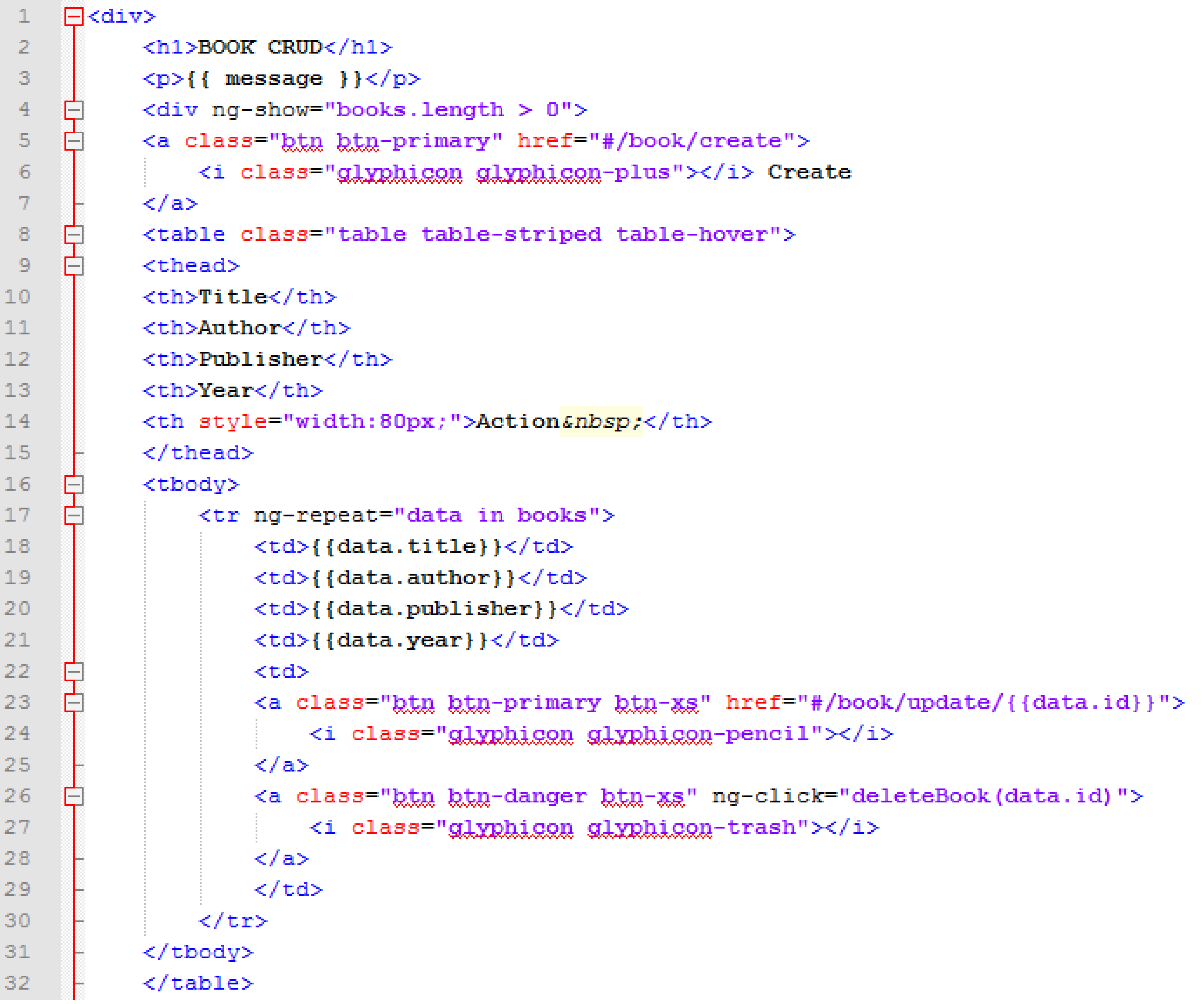


```
spaApp_book.config(['$routeProvider', function($routeProvider) {
  $routeProvider
    .when('/book/index', {
        templateUrl: 'views/book/index.html',
        controller: 'index'
    })
    .when('/book/create', {
        templateUrl: 'views/book/create.html',
        controller: 'create',
        resolve: {
            book: function(services, $route){
                return services.getBooks();
            }
        }
    })
    .when('/book/update/:bookId', {
        templateUrl: 'views/book/update.html',
        controller: 'update',
        resolve: {
          book: function(services, $route){
            var bookId = $route.current.params.bookId;
            return services.getBook(bookId);
          }
        }
    })
    .when('/book/delete/:bookId', {
        templateUrl: 'views/book/index.html',
        controller: 'delete',
    })
    .otherwise({
        redirectTo: '/book/index'
    });
}]);
```

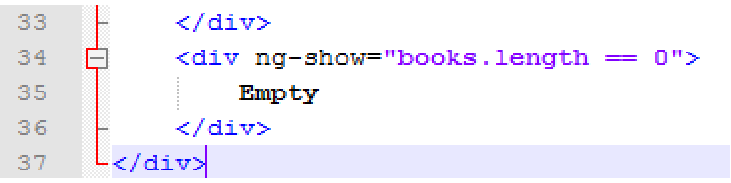


```
spaApp_book.controller('index', ['$scope', '$http', 'services',
    function($scope,$http,services) {
    $scope.message = 'Everyone come and see how good I look!';
    services.getBooks().then(function(data){
        $scope.books = data.data;
    });
    $scope.deleteBook = function(bookID) {
        if(confirm("Are you sure to delete book number: " + bookID)==true && bookID>0){
            services.deleteBook(bookID);
            $route.reload();
        }
    };
}])
.controller('create', ['$scope', '$http', 'services','$location','book',
    function($scope,$http,services,$location,book) {
    $scope.message = 'Look! I am an about page.';
    $scope.createBook = function(book) {
        var results = services.createBook(book);
    }
}])
.controller('update', ['$scope', '$http', '$routeParams', 'services','$location','book',
    function($scope,$http,$routeParams,services,$location,book) {
    $scope.message = 'Contact us! JK. This is just a demo.';
    var original = book.data;
    $scope.book = angular.copy(original);
    $scope.isClean = function() {
        return angular.equals(original, $scope.book);
    }
    $scope.updateBook = function(book) {
        var results = services.updateBook(book);
    }
}]);
```

10. Create index.html in folder views/book/

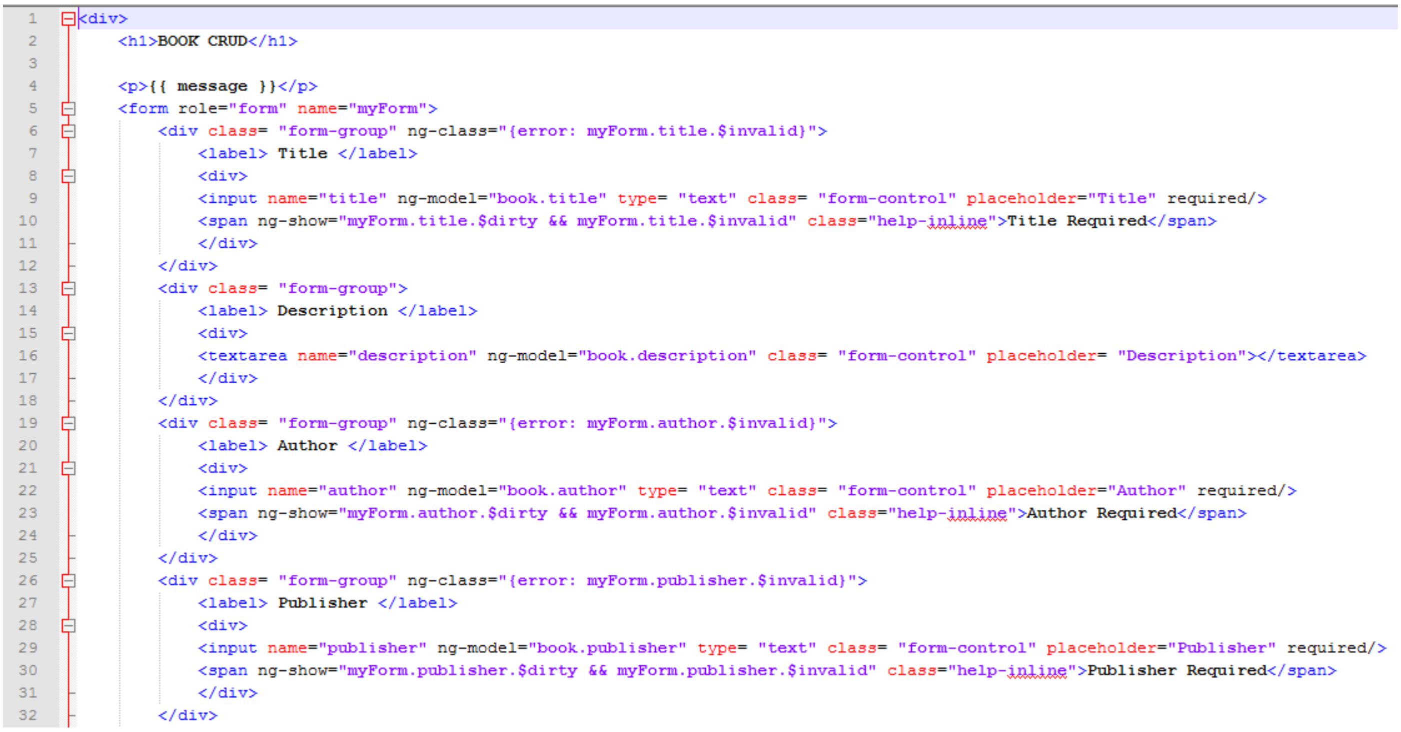


```
<div>
    <h1>BOOK CRUD</h1>
    <p>{{ message }}</p>
    <div ng-show="books.length > 0">
    <a class="btn btn-primary" href="#/book/create">
        <i class="glyphicon glyphicon-plus"></i> Create
    </a>
    <table class="table table-striped table-hover">
    <thead>
    <th>Title</th>
    <th>Author</th>
    <th>Publisher</th>
    <th>Year</th>
    <th style="width:80px;">Action&nbsp;</th>
    </thead>
    <tbody>
        <tr ng-repeat="data in books">
            <td>{{data.title}}</td>
            <td>{{data.author}}</td>
            <td>{{data.publisher}}</td>
            <td>{{data.year}}</td>
            <td>
            <a class="btn btn-primary btn-xs" href="#/book/update/{{data.id}}">
                <i class="glyphicon glyphicon-pencil"></i>
            </a>
            <a class="btn btn-danger btn-xs" ng-click="deleteBook(data.id)">
                <i class="glyphicon glyphicon-trash"></i>
            </a>
            </td>
        </tr>
    </tbody>
    </table>
```

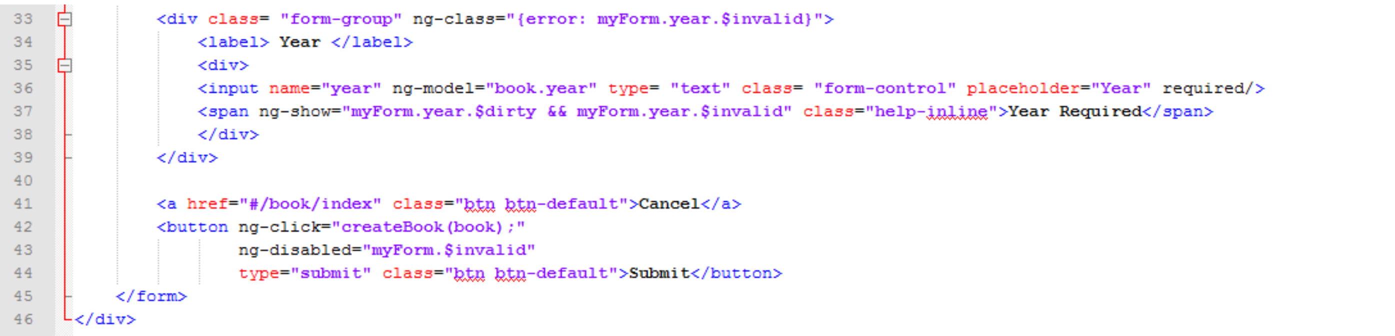


```
    </div>
    <div ng-show="books.length == 0">
        Empty
    </div>
</div>
```

11. Create create.html in folder views/book/

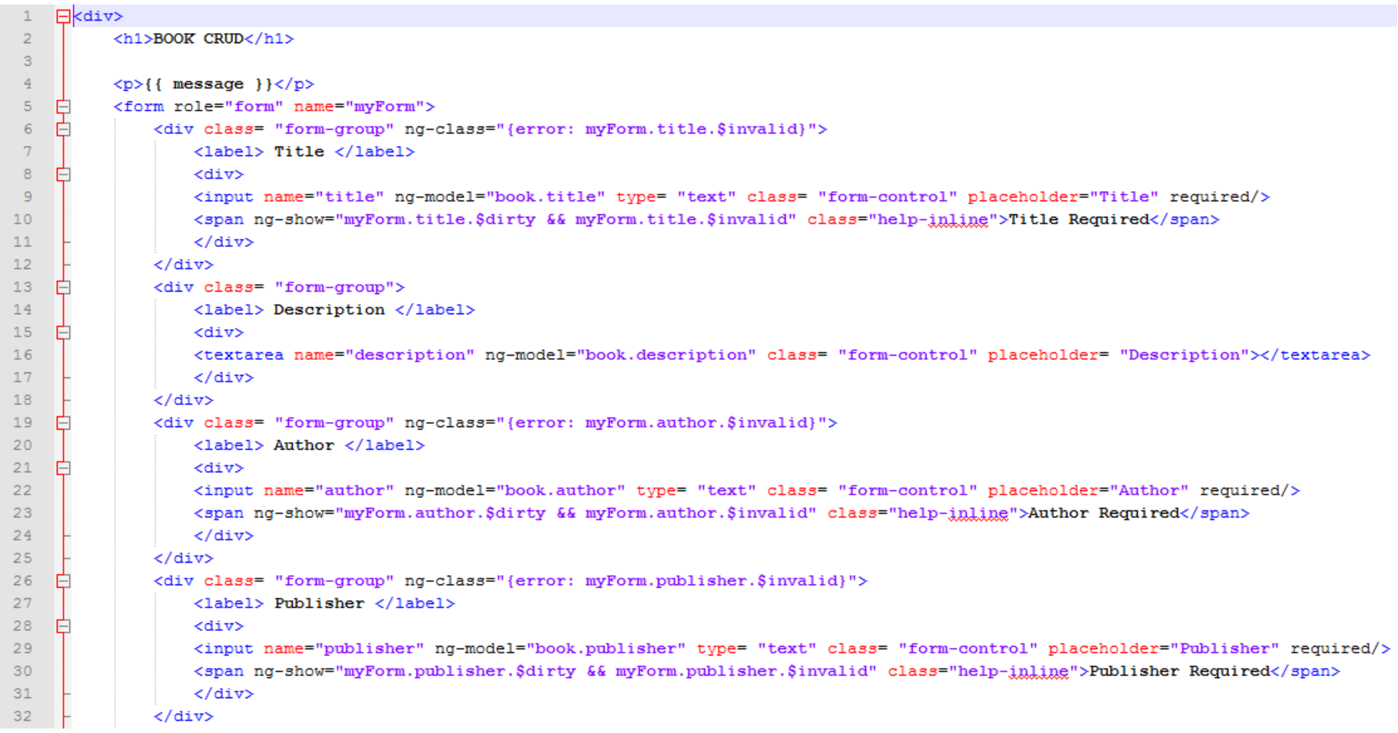


```html
<div>
    <h1>BOOK CRUD</h1>

    <p>{{ message }}</p>
    <form role="form" name="myForm">
        <div class= "form-group" ng-class="{error: myForm.title.$invalid}">
            <label> Title </label>
            <div>
            <input name="title" ng-model="book.title" type= "text" class= "form-control" placeholder="Title" required/>
            <span ng-show="myForm.title.$dirty && myForm.title.$invalid" class="help-inline">Title Required</span>
            </div>
        </div>
        <div class= "form-group">
            <label> Description </label>
            <div>
            <textarea name="description" ng-model="book.description" class= "form-control" placeholder= "Description"></textarea>
            </div>
        </div>
        <div class= "form-group" ng-class="{error: myForm.author.$invalid}">
            <label> Author </label>
            <div>
            <input name="author" ng-model="book.author" type= "text" class= "form-control" placeholder="Author" required/>
            <span ng-show="myForm.author.$dirty && myForm.author.$invalid" class="help-inline">Author Required</span>
            </div>
        </div>
        <div class= "form-group" ng-class="{error: myForm.publisher.$invalid}">
            <label> Publisher </label>
            <div>
            <input name="publisher" ng-model="book.publisher" type= "text" class= "form-control" placeholder="Publisher" required/>
            <span ng-show="myForm.publisher.$dirty && myForm.publisher.$invalid" class="help-inline">Publisher Required</span>
            </div>
        </div>
```

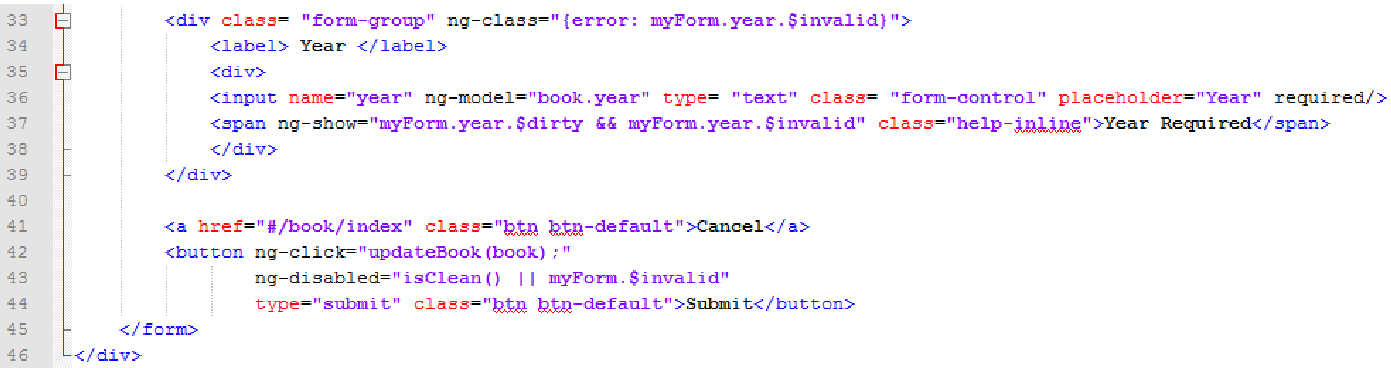


```html
        <div class= "form-group" ng-class="{error: myForm.year.$invalid}">
            <label> Year </label>
            <div>
            <input name="year" ng-model="book.year" type= "text" class= "form-control" placeholder="Year" required/>
            <span ng-show="myForm.year.$dirty && myForm.year.$invalid" class="help-inline">Year Required</span>
            </div>
        </div>

        <a href="#/book/index" class="btn btn-default">Cancel</a>
        <button ng-click="createBook(book);"
                ng-disabled="myForm.$invalid"
                type="submit" class="btn btn-default">Submit</button>
    </form>
</div>
```

12. Create update.html in folder views/book/

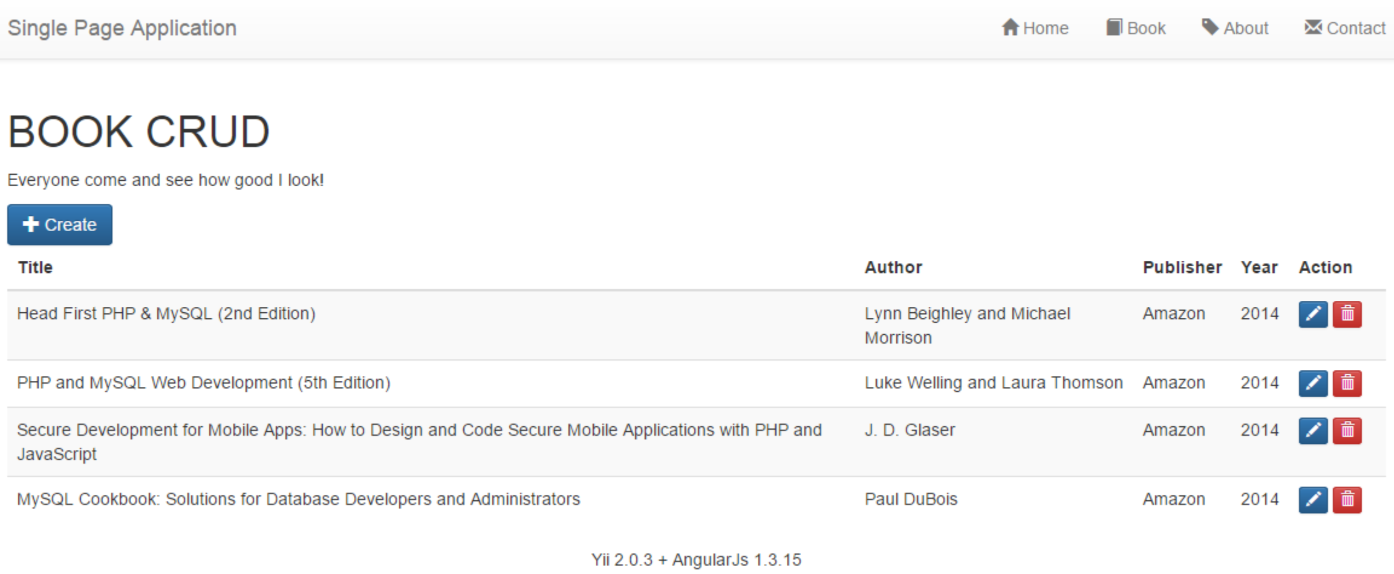


```html
<div>
    <h1>BOOK CRUD</h1>

    <p>{{ message }}</p>
    <form role="form" name="myForm">
        <div class= "form-group" ng-class="{error: myForm.title.$invalid}">
            <label> Title </label>
            <div>
            <input name="title" ng-model="book.title" type= "text" class= "form-control" placeholder="Title" required/>
            <span ng-show="myForm.title.$dirty && myForm.title.$invalid" class="help-inline">Title Required</span>
            </div>
        </div>
        <div class= "form-group">
            <label> Description </label>
            <div>
            <textarea name="description" ng-model="book.description" class= "form-control" placeholder= "Description"></textarea>
            </div>
        </div>
        <div class= "form-group" ng-class="{error: myForm.author.$invalid}">
            <label> Author </label>
            <div>
            <input name="author" ng-model="book.author" type= "text" class= "form-control" placeholder="Author" required/>
            <span ng-show="myForm.author.$dirty && myForm.author.$invalid" class="help-inline">Author Required</span>
            </div>
        </div>
        <div class= "form-group" ng-class="{error: myForm.publisher.$invalid}">
            <label> Publisher </label>
            <div>
            <input name="publisher" ng-model="book.publisher" type= "text" class= "form-control" placeholder="Publisher" required/>
            <span ng-show="myForm.publisher.$dirty && myForm.publisher.$invalid" class="help-inline">Publisher Required</span>
            </div>
        </div>
```



```
        <div class= "form-group" ng-class="{error: myForm.year.$invalid}">
            <label> Year </label>
            <div>
            <input name="year" ng-model="book.year" type= "text" class= "form-control" placeholder="Year" required/>
            <span ng-show="myForm.year.$dirty && myForm.year.$invalid" class="help-inline">Year Required</span>
            </div>
        </div>

        <a href="#/book/index" class="btn btn-default">Cancel</a>
        <button ng-click="updateBook(book);"
                ng-disabled="isClean() || myForm.$invalid"
                type="submit" class="btn btn-default">Submit</button>
    </form>
</div>
```

13. Testing now



Single Page Application

BOOK CRUD

Everyone come and see how good I look!

Title	Author	Publisher	Year	Action
Head First PHP & MySQL (2nd Edition)	Lynn Beighley and Michael Morrison	Amazon	2014	✎ 🗑
PHP and MySQL Web Development (5th Edition)	Luke Welling and Laura Thomson	Amazon	2014	✎ 🗑
Secure Development for Mobile Apps: How to Design and Code Secure Mobile Applications with PHP and JavaScript	J. D. Glaser	Amazon	2014	✎ 🗑
MySQL Cookbook: Solutions for Database Developers and Administrators	Paul DuBois	Amazon	2014	✎ 🗑

Yii 2.0.3 + AngularJs 1.3.15

How about You?

Can You create CRUD for some tables?

Contact Us

+6281559915720

hafidmukhlasin@gmail.com

http://www.hafidmukhlasin.com