**CS/BS 全栈开发 Modern web application development**

Note: could not delete file “ because the path is too long?” (because npm nests dependencies)

**template> npm install –g rimraf**

**template> rimraf node\_modules**

1. **Deployd**

生成RESTful API based on database

<http://deployd.com/>

1. 创建web app框架

>dpd create sportstore & cd sportstore\public

2) 创建angular框架,写代码，构建

public>yo angular

public>grunt serve:dist

1. 启动web app

sportstore> dpd –p 5500 app.apd

localhost:5500\dist\index.html

1. **Jhipster 全栈开发 (Angularjs + Sprint boot )**

pre-condition: node, git

**set sys env: npm; git\bin**

// install tools for frontend

>npm install –g yo bower grunt-cli

若无法下载，则设置proxy

**> npm config set proxy** [**http://165.225.96.34:10015**](http://165.225.96.34:10015)

**> npm config set https-proxy** [**http://165.225.96.34:10015**](http://165.225.96.34:10015)

// install jhipster

Template > **node --version && npm --version && git –version**

Template > **yo --version && bower --version && grunt --version**

Template > npm install –g generator-jhipster

Template > yo jhipster

What is the base name of your application? sample

What is your default Java package name” com.philips.sample

Do you want to use Java 8? Yes

Which type of authentication would you like to use? HTTP Session Authentication (stateful,…)

Which type of database would you like to use? SQL

Which production database would you like to use? PostgreSQL

Which development database would like to use? PostgreSQL

Do you want to use Hibernate 2nd level cache? Yes, with encache (local cache, for a single node)

Do you want to use clustered HTTP sessions? No

Do you want to use WebSockets? No

Would you like to use Maven or Gradle for building the backend? Gradle

Would you like to use Grunt or Gulp.js for building the frontend? Grunt

Would you like to use the Compass CSS Authoring Framework? Yes

pdAmin3 for PostgreSQL

if fails, >npm install & bower install

// run live-reload

Template > grunt

Features:

angular, bootstrap, bower, karma, grunt 都可以正常使用

Note:

1. bower代理：

若bower\_package无法找到，需要翻墙下载，设置bower代理，然后>bower install

set env for设置bower代理

**HTTP\_PROXY =** [**http://165.225.96.34:10015**](http://165.225.96.34:10015)

**HTTPS\_PROXY =** [**http://165.225.96.34:10015**](http://165.225.96.34:10015)

1. 若sass无法编译，需要安装sass.js, 先安装ruby,然后>gem install sass
2. 若git 别人的sample,>grunt报错unable to find local grunt, 则执行

>npm install grunt –save-dev

>npm install

// run server

Template > gradlew

access to web exploer: <http://localhost:8080>

Features:

1. Spring boot (jpa, mongodb, cassandra, Spring security, spring mvc REST, Thymeleaf(server-side templates), monitoring), liquibase, Caching(Encache, HazelCast, Hibernate 2nd level cache or Spring Caching abstraction), cloud deployment
2. Template > yo jhipster:entity foo

Template > yo jhipster:service user

1. Spring Security includes Ajax endpoints, secured remember-me, audits…

Note:

若无法连上数据库服务器，启动数据库服务器

若出现下载mongodb失败，是因为测试使用内置mondodb服务器，而下载需要设置代理（目前代码写死，或者可以配置我还没找到），只需在build.gradle注释de.flapdoodle.embed.mongo,这样spring boot就不会自动加载和注入内置mongodb服务器）

>yo jhipster:entity author

自动生成如下

//配置文件

.jhipster/Author.json

//后台

src/main/java/com.philips.bookstore/domain/Author

src/main/java/com.philips.bookstore/repository/AuthorRepository

src/main/java/com.philips.bookstore/web/AuthorResource

//数据库

src/main/resources/config/liquibase/master.xml

<include file="classpath:config/liquibase/changelog/20160101093953\_added\_entity\_Author.xml" relativeToChangelogFile="false"/>

src/main/resources/config/liquibase/changelog/20160101093953\_added\_entity\_Author.xml

pgAdmin III -> PostgreSQL 9.5 (localhost:5432) -> bookstore -> author

//前端

src/main/webapp/scripts/app/entities/author/…

src/main/webapp/scripts/components/entities/author/…

src/main/webapp/i18n/en/author.json

src/main/webapp/i18n/fr/author.json

//测试

src/test/java/com.philips.bookstore/web/rest/AthorResourceIntTest.java

src/test/javascript/spec/app/entities/author/author-detail.controller.spec.js

src/test/gatling/simlations/AuthorGatlingTest.scala

1. Basic knowledge
2. 如何实现异步函数？

使用$q.defer().resolve(data), $q.defer().promise().then(function(data){})

$q: A service that helps you run functions asynchronously, and use their return values (or exceptions) when they are done processing.

服务层：

angular.module('21pointsApp')

.factory('Language', function ($q, $http, $translate, LANGUAGES) {

return {

getCurrent: function () {

var deferred = $q.defer();

var language = $translate.storage().get('NG\_TRANSLATE\_LANG\_KEY');

if (angular.isUndefined(language)) { language = 'en'; }

deferred.resolve(language);

return deferred.promise;

}

};

})

客户层：

Language.getCurrent().then(function (language) {

$translate.use(language);

});

1. AngularJS Resource：与 RESTful API 交互

Refer: <https://docs.angularjs.org/api/ngResource/service/$resource>

var app = angular.module('helloApp, ['ngResource']);

app.factory('BloodPressure', function ($resource, DateUtils) {

return $resource('api/bloodPressures/:id', {}, {

'byMonth': { method: 'GET', isArray: false, url: 'api/bp-by-month/:month'},

'last30Days': { method: 'GET', isArray: false, url: 'api/bp-by-days/30'},

'get': { method: 'GET',

transformResponse: function (data) {

data = angular.fromJson(data);

data.timestamp = DateUtils.convertDateTimeFromServer(data.timestamp);

return data;

}

},

'update': { method:'PUT' }

});

});

$resource提供默认5种actions:

{ 'get': { method: 'GET' },

'save': { method: 'POST' },

'query': { method: 'GET', isArray:true },

'remove': { method: 'DELETE' },

'delete': { method: 'DELETE' }

};

It is important to realize that invoking a $resource object method **immediately returns an empty reference (object or array depending on isArray)**. Once the data is returned from the server the existing reference is populated with the actual data. This is a useful trick since **usually the resource is assigned to a model** which is then rendered by the view. Having an empty object results in no rendering, once the data arrives from the server then the object is populated with the data and the view automatically re-renders itself showing the new data. This means that **in most cases one never has to write a callback function for the action methods**.

资源的操作有两种种调用方式：

通过资源类

HTTP GET "class" actions: Resource.action([parameters], [success], [error])

non-GET "class" actions: Resource.action([parameters], postData, [success], [error])

for example:

//不需要处理回调函数：绑定视图模型,等异步获取数据完，再次渲染视图

$scope.bloodPressure = BloodPressure.get({id : $stateParams.id});

// 需要处理回调函数

$scope.bloodPressures = [];

BloodPressure.query({page: $scope.page, per\_page: 20}, function(result, headers) {

$scope.links = ParseLinks.parse(headers('link'));

for (var i = 0; i < result.length; i++) {

$scope.bloodPressures.push(result[i]);

}

});

BloodPressure.get({id: id}, function (result) {

$scope.bloodPressure = result;

});

$scope.save = function () {

if ($scope.bloodPressure.id != null) {

BloodPressure.update($scope.bloodPressure, onSaveFinished);

} else {

BloodPressure.save($scope.bloodPressure, onSaveFinished);

}

};

// communication between edit resource and view resource

bloodPressure-dialog.controller.js

var onSaveFinished = function (result) {

$scope.$emit('21pointsApp:bloodPressureUpdate', result);

};

bloodPressure-detail.controller.js

$rootScope.$on('21pointsApp:bloodPressureUpdate', function(event, result) {

$scope.bloodPressure = result;

});

通过资源实例

non-GET instance actions: instance.$action([parameters], [success], [error])

Calling these methods invoke an $http with the specified http method, destination and parameters. When the data is returned from the server then the object is an instance of the resource class. The actions save, remove and delete are available on it as methods with the $ prefix. This allows you to easily perform CRUD operations (create, read, update, delete) on server-side data

For example:

var bloodPressures = BloodPressure.query({page: $scope.page, per\_page: 20}, function(result, headers) {

var bloodPressure = bloodPressure[0] ;

bloodPressure.$save() ;

bloodPressure.$get() ; //重新载入

bloodPressure.$delete() ;

};

new BloodPressure(bloodPressure).$save() ;

若需要处理回调函数，均可以在$action().then(function(bloodPressure) {…} ) ;

1. $httpProvider.interceptors

Refer to: <https://docs.angularjs.org/api/ng/service/$http>

The $http API is based on the deferred/promise APIs exposed by the $q service

For purposes of global error handling, authentication, or any kind of synchronous or asynchronous pre-processing of request or postprocessing of responses, it is desirable to be able to intercept requests before they are handed to the server and responses before they are handed over to the application code that initiated these requests.

There are two kinds of interceptors (and two kinds of rejection interceptors):

**request**: interceptors get called with a http config object. The function is free to modify the config object or create a new one. The function needs to return the config object directly, or a promise containing the config or a new config object.

**requestError**: interceptor gets called when a previous interceptor threw an error or resolved with a rejection.

**response**: interceptors get called with http response object. The function is free to modify the response object or create a new one. The function needs to return the response object directly, or as a promise containing the response or a new response object.

**responseError**: interceptor gets called when a previous interceptor threw an error or resolved with a rejection.

$httpProvider.interceptors.push('errorHandlerInterceptor');

$httpProvider.interceptors.push('authExpiredInterceptor');

$httpProvider.interceptors.push('authInterceptor');

$httpProvider.interceptors.push('notificationInterceptor');

.factory('errorHandlerInterceptor', function ($q, $rootScope) {

return {

**'responseError'**: function (**response**) {

if (!(response.status == 401 && response.data.path.indexOf("/api/account") == 0 )){

$rootScope.$emit('21pointsApp.httpError', response);

}

return **$q.reject(response)**;

}};

});

.factory('authInterceptor', function ($rootScope, $q, $location, localStorageService) {

return {

// Add authorization token to headers

**request**: function (**config**) {

config.headers = config.headers || {};

var token = localStorageService.get('token');

if (token && token.expires && token.expires > new Date().getTime()) {

config.headers['x-auth-token'] = token.token;

}

return **config**;

}};

})

.factory('authExpiredInterceptor', function ($rootScope, $q, $injector, localStorageService) {

return {

**responseError**: function (**response**) {

// token has expired

if (response.status === 401 && (response.data.error == 'invalid\_token' || response.data.error == 'Unauthorized')) {

localStorageService.remove('token');

var Principal = $injector.get('Principal');

if (Principal.isAuthenticated()) {

var Auth = $injector.get('Auth');

Auth.authorize(true);

}

}

return **$q.reject(response)**;

}};

});

.factory('notificationInterceptor', function ($q, AlertService) {

return {

**response**: function(**response**) {

var alertKey = response.headers('X-21pointsApp-alert');

if (angular.isString(alertKey)) {

AlertService.success(alertKey, { param : response.headers('X-21pointsApp-params')});

}

return **response**;

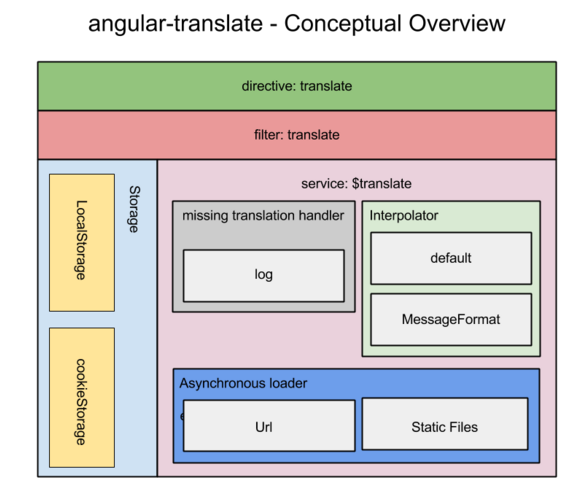
}};

});

1. Angular-translate

Refer to : <http://angular-translate.github.io/>

>bower install - -save angular-translate



var app = angular.module('myApp', ['pascalprecht.translate']);

$translateProvider全局配置$translate Service（因为是单例对象）, 然后使用$translate

$translateProvider.useLoader('$translatePartialLoader', {

urlTemplate: 'i18n/{lang}/{part}.json'

});

$translateProvider.preferredLanguage('en'); //或者采用自动查找浏览器

// try to find out preferred language by yourself, 搜索顺序:

It searches for values in the window.navigator object in the following properties (also in this order):

navigator.languages[0]

navigator.language

navigator.browserLanguage

navigator.systemLanguage

navigator.userLanguage

$translateProvider.determinePreferredLanguage();

// Switching the language at runtime

$scope.changeLanguage = function (langKey) {

$translate.use(langKey);

};

// fallback languages: 若指定key在指定的语言中找不到，则从默认语言找

$translateProvider.fallbackLanguage(['en', 'fr']);

//Language Negotiation: 若自动化自动language key,一般是从浏览器获得，而浏览器language key很具体，需要进行映射，如若浏览器使用en\_US，则app应用en

$translateProvider

.translations('en', { /\* ... \*/ })

.translations('de', { /\* ... \*/ })

.registerAvailableLanguageKeys(['en', 'de'], {

'en\_US': 'en',

'en\_UK': 'en',

'de\_DE': 'de',

'de\_CH': 'de'

})

.determinePreferredLanguage();

// Storages: To let you app remember the language users choose, angular-translate comes with a support for Storages. What ever storage you use, angular-translate will save a language key with a specific identifier in it, so it can ask for it next time the user launches the app. angular-translate has built-in support for two storages. localStorage and cookieStorage. Whereas localStorage falls back to cookieStorage if it isn't supported by the browser the user currently uses

>$ bower install angular-translate-storage-cookie

var module = angular.module('AppService', ['pascalprecht.translate','ngCookies']);

$translateProvider.useCookieStorage();

>$ bower install angular-translate-storage-local

$translateProvider.useLocalStorage();

浏览器会在local storage or cookie storage自动记录key-value,

{

"NG\_TRANSLATE\_LANG\_KEY": en

}

当然也可以获取：var language = $translate.storage().get('NG\_TRANSLATE\_LANG\_KEY');

// asynchronous loading: load your data from a server you have to use an asynchronous loader, which gets invoked later at runtime when it's needed. angular-translate comes with support for three different asynchronous loaders:

a. Using urlLoader

>$ bower install angular-translate-loader-url

$translateProvider.useUrlLoader('foo/bar.json');

$translateProvider.preferredLanguage('en');

actually requests foo/bar.json?lang=en

b. Using staticFilesLoader

>$ bower install angular-translate-loader-static-files

$translateProvider.useStaticFilesLoader({

prefix: 'locale-',

suffix: '.json'

});

$translateProvider.preferredLanguage('en');

angular-translate will concatenate the given information to {{prefix}}{{langKey}}{{suffix}}. So this will load locale-en.json.

c. Using partialLoader

>$ bower install angular-translate-loader-partial

// Initialize angular-translate

$translateProvider.useLoader('$translatePartialLoader', {

urlTemplate: 'i18n/{lang}/{part}.json'

});

$translateProvider.preferredLanguage('en');

//tell angular-translate which part you want to load when.

.state('bloodPressure', {

resolve: {

translatePartialLoader: ['$translate', '$translatePartialLoader', function ($translate, $translatePartialLoader) {

$translatePartialLoader.addPart('bloodPressure');

$translatePartialLoader.addPart('global');

return $translate.refresh();

}]

}

})

// Pluralization

MessageFormat: simple variable replacement, SelectFormat, and PluralFormat.

Please refer to:

<http://userguide.icu-project.org/formatparse/messages>, <https://github.com/SlexAxton/messageformat.js>

>$ bower install angular-translate-interpolation-messageformat

$translateProvider.addInterpolation('$translateMessageFormatInterpolation');

our app is configured using the default interpolation, but is also aware that there's another interpolation service that could be used for specific translations

for example:

var app = angular.module('myApp', ['pascalprecht.translate']);

app.config(['$translateProvider', function ($translateProvider) {

$translateProvider.preferredLanguage('en');

$translateProvider.addInterpolation('$translateMessageFormatInterpolation');

$translateProvider.translations('en', {

HEADLINE: 'I\'m a headline',

TEXT: 'I\'m using default interpolation {{ val + val }}',

PLURAL: '{GENDER, select, male{He} female{She} other{They}} liked this.',

});

$translateProvider.translations('de', {

TEXT: 'Ich benutze default interpolation {{ val + val }}',

PLURAL: '{GENDER, select, male{Er fand} female{Sie fand} other{Sie fanden}} es gut.',

});

}]);

<p translate="TEXT" translate-values="{ val: 5 }"></p>

<p translate="PLURAL" translate-values="{ GENDER: 'other' }" translate-interpolation="messageformat"></p>

result in en version:

I'm using default interpolation 10

They liked this.

result in de version:

Ich benutze default interpolation 10

Sie fanden es gut.

Security

$translateProvider.useSanitizeValueStrategy('sanitize');

the following strategies are built-in:

sanitize: sanitizes HTML in the translation text using $sanitize 净化

escape: escapes HTML in the translation 转义

sanitizeParameters: sanitizes HTML in the values of the interpolation parameters using $sanitize

escapeParameters: escapes HTML in the values of the interpolation parameters

webapp/i18n/en/global.json:

//key: 层次的，可引用内容的

{

"global": {

"foo": {

"foo": "This is my text."

},

"SOME\_NAMESPACE": {

"OK\_TEXT": "OK"

},

"ANOTHER\_NAMESPACE": {

"OK\_TEXT": "@:global.SOME\_NAMESPACE.OK\_TEXT"

}

}

}

3种使用方式：translate directive, translate filter and $translate service

Directive:

<h2 translate="21pointsApp.bloodPressure.home.title">Blood Pressures</h2>

<td><span translate="21pointsApp.bloodPressure.timestamp">Timestamp</span></td>

Filter:

<input type="text" class="form-control" ng-model="searchQuery" id="searchQuery" placeholder="{{'entity.action.search' | translate}}">

Service:

app.controller('Ctrl', ['$scope', '$translate', function ($scope, $translate) {

$translate(['HEADLINE', 'PARAGRAPH', 'NAMESPACE.PARAGRAPH']).then(function (translations) {

$scope.headline = translations.HEADLINE;

$scope.paragraph = translations.PARAGRAPH;

$scope.namespaced\_paragraph = translations['NAMESPACE.PARAGRAPH'];

});

}]);

字符串格式化

{

"TRANSLATION\_ID": "{{username}} is logged in."

}

angular.module('myApp').controller('Ctrl', ['$scope', function ($scope) {

$scope.translationData = {

username: 'PascalPrecht'

};

}]);

{{ 'TRANSLATION\_ID' | translate: translationData }}

<ANY translate="TRANSLATION\_ID" translate-values="{username: translationData.username}"> </ANY>

1. angular-dynamic-locale

<https://scotch.io/tutorials/internationalization-of-angularjs-applications>

<https://github.com/lgalfaso/angular-dynamic-locale>

angular-translate (it’s used for handling language translation stuff)

angular-dynamic-locale (it’s used for **changing angular $locale- which means formatting dates, numbers, currencies, etc.**) libraries.

>bower install --save angular-dynamic-locale

angular.module('myApp').controller('myController', [..., 'tmhDynamicLocale',

function(..., tmhDynamicLocale) {

tmhDynamicLocaleProvider.localeLocationPattern('bower\_components/angular-i18n/angular-locale\_{{locale}}.js');

tmhDynamicLocaleProvider.useCookieStorage();

tmhDynamicLocaleProvider.storageKey('NG\_TRANSLATE\_LANG\_KEY');

}

])

1. angular-local-storage

$ bower install angular-local-storage --save

var myApp = angular.module('myApp', ['LocalStorageModule']);

myApp.config(function (localStorageServiceProvider) {

//You could set a prefix to avoid overwriting any local storage variables from the rest of your app, Default prefix: ls.<your-key>

localStorageServiceProvider.setPrefix('yourAppName');

//You could change web storage type to localStorage or sessionStorage, Default storage: localStorage

localStorageServiceProvider.setStorageType('sessionStorage');

//Set cookie options (usually in case of fallback), expiry: number of days before cookies expire (0 = does not expire). default: 30; path: the web path the cookie represents. default: '/'

localStorageServiceProvider.setStorageCookie(45, '<path>');

//Send signals for each of the following actions: setItem , default: true; removeItem , default: false

localStorageServiceProvider.setNotify(true, true);

localStorageService.isSupported;

var storageType = localStorageService.getStorageType(); //e.g localStorage

//Directly add/get a value to local storage.If local storage is not supported, use cookies instead.

localStorageService.set(key, val);

localStorageService.get(key);

localStorageService.keys();

localStorageService.remove(key1, key2, key3, ...);

localStorageService.clearAll();

localStorageService.clearAll(/^\d+$/); // clear numbers

localStorageService.set('property', 'oldValue');

localStorageService.deriveKey('property'); // ls.property

// Return localStorageService.length, ignore keys that not owned.

var lsLength = localStorageService.length();

});

1. angular-ui-router

<https://github.com/angular-ui/ui-router>

>bower install -- save angular-ui-router

**State:**

change application view based on application state, including nested states (for nested views) and multiple named views

**Activating a state:**

* **Click a link containing the ui-sref directive**

ui-sref: create a link, point to a certain state of your application)

* **$state.go()**
* **Navigate to the url associated with the state**

**Controller:**

The controller will not be instantiated if template is not defined.

**Parent.Child state:**

* dot syntax to infer your hierarchy to the $stateProvider
* **parent state must exist**.
* No two states can have the same name.
* When a state is “active”, all of its ancestor states are implicitly active as well
* **Child state will load their templates into their parent’s ui-view** (When a state is activated,its templates are automatically inserted into the ui-view of its parent state’s template)
* Abstract state can have child states but can not get activated itself, abstract state still need their own <ui-view/> for their children to plug into
* register states in any order and across modules, you can register children before the parent state exists. It will queue them up and once the parent state is registered then the child will be registered.
* Child states DO inherit the following from parent states: **resolved dependencies, custom data properties,** children of abstract states do inherit the **url** property of their parent as a prefix of their own url

**Resolve:**

use resolve to provide your controller with content or data that is custom to the state, If any of these dependencies are promises, they will be resolved and converted to a value before the controller is instantiated and the $stateChangeSuccess event is fired.

**data:**

attach custom data to the state object (we recommend using a data property to avoid conflicts)

**onEnter and onExit:**

'onEnter' and 'onExit' callbacks that get called when a state becomes active and inactive respectively. The callbacks also have access to all the resolved dependencies.

**State Change Events:**

//fired when the transition begins.

$rootScope.$on('$stateChangeStart', function(event, toState, toParams, fromState, fromParams){ ... })

//fired once the state transition is complete.

$rootScope.$on('$stateChangeSuccess', function(event, toState, toParams, fromState, fromParams){ ... })

Multiple Named Views: (multiple ui-views per template)

**View name**: is the name used in the view directive inside parent’s html

**state name**: is the state's absolute name, 若没有statename，指index.html

relative names: **viewname -----(statename 默认为当前状态)**

absolute names: **viewname@statename ----(current state will plug into ui-view=”viewname” within statename所指定的html里)**

for example:

$stateProvider.state('site', {

**'abstract': true,**

views: { //说明navbar子视图插入到url=\,即插入到state=site.home中

'**navbar@**': {

templateUrl: 'scripts/components/navbar/navbar.html',

controller: 'NavbarController'

}

},

resolve: { //controller的依赖项

authorize: ['Auth', function (Auth) { return Auth.authorize(); } ],

translatePartialLoader: ['$translate', '$translatePartialLoader', function ($translate, $translatePartialLoader) { $translatePartialLoader.addPart('global'); }]

}

});

$stateProvider.state('home', {

**parent: 'site',**

**url: '/', //入口\(url) = site.home (state)**

data: { roles: [] },

views: {

'**content@**': { //content子视图插入到url=\index.html 等价于state=site.home

templateUrl: 'scripts/app/main/main.html',

controller: 'MainController'

}

},

resolve: {

mainTranslatePartialLoader: ['$translate', '$translatePartialLoader', function ($translate, $translatePartialLoader) {

$translatePartialLoader.addPart('main');

$translatePartialLoader.addPart('weight');

return $translate.refresh();

}],

points: function(Points) { return Points.thisWeek().$promise; }

}

})

$stateProvider.state('about', {

**parent: 'site',**

**url: '/about', //入口/#/about (url) == site.about (state)**

data: { roles: [] },

views: {

**'content@'**: { //content子视图插入到url=\index.html 等价于state=site.home

templateUrl: 'scripts/app/about/about.html'

}

},

resolve: {

aboutTranslatePartialLoader: ['$translate', '$translatePartialLoader', function ($translate, $translatePartialLoader) {

$translatePartialLoader.addPart('main');

return $translate.refresh();

}]

}

});

$stateProvider.state('register', {

**parent: 'account',**

**url: '/register', //入口/#/register(url) == site.account.register (state)**

data: { roles: [], pageTitle: 'register.title' },

views: {

'content@': { //content子视图插入到url=\index.html等价于state=site.home

templateUrl: 'scripts/app/account/register/register.html',

controller: 'RegisterController'

}

},

resolve: {

translatePartialLoader: ['$translate', '$translatePartialLoader', function ($translate, $translatePartialLoader) {

$translatePartialLoader.addPart('register');

return $translate.refresh();

}]

}

});

切换应用程序状态方式：用户输入url, 或者点击ui-sref=”statename”,或者$state.go(‘statename’);

应用程序状态注册方式angular.module(‘21pointsApp’).config(function($stateProvider){

$stateProvider.state(‘statename1’, {}).state(‘statename2’, {})…

}) ;

类似依赖注入，所有应用程序状态会被扫描到

/index.html = ui-view="navbar" + ui-view="content"

components/navbar/navbar.html

若site.home = /, 则content = app/main/main.html + main.controller.js

若site.about = /about, 则content = app/about/about.html

若site.accout.register = /register, 则content = app/account/register/register.html + register.controller.js

若site.account.login = /login, 则content = app/account/login/login.html + login.controller.js

若site.account.logout = /logout, 则content = app/main/main.html + logout.controller.js

若site.account.settings = /settings, 则content = app/account/settings/settings.html + settings.controller.js

若site.account.password = /password, 则content = app/account/password/password.html + password.controller.js + password.directive.js

若site.account.activate = /activate?key, 则content = app/account/activate/activate.html + activate.controller.js

若site.account.requestReset = /reset/request,

则content = app/account/reset/request/reset.request.html + reset.request.controller.js

若site.account.finishReset = /reset/finish?key,

则content = app/account/reset/finish/reset.finish.html + reset.finishcontroller.js

1. **angular-fullstack (MongoDB, ExpressJS, AngularJS, NodeJS)**

// install meanjs

project-parent-directory> **mkdir template && cd template**

template > **node --version && npm --version && git --version**

template > **yo --version && bower --version && grunt --version**

template> **npm install –g generator-angular-fullstack**

template> yo angular-fullstack

Note: 若出现error, ‘rm’ is not internal command, do in the following way:

install git -> right clicked inside project folder -> git bash here -> yo meanjs

// run mongod

>mongod –dbpath ./data

1. **MEAN全栈开发 (MongoDB, ExpressJS, AngularJS, NodeJS)**

// install meanjs

project-parent-directory> **mkdir template && cd template**

template > **node --version && npm --version && git --version**

template > **yo --version && bower --version && grunt --version**

template> **npm install –g generator-meanjs**

template> yo meanjs

Note: 若出现error, ‘rm’ is not internal command, do in the following way:

install git -> right clicked inside project folder -> git bash here -> yo meanjs