

ionic 路由

学习要点:

1. Ionic 中路由管理介绍
2. ionic 中内联模板介绍
3. ionic 路由机制: 状态
4. 导航视图: `ion-nav-view`
5. 模板视图: `ion-view`
6. 导航栏: `ion-nav-bar`
7. 回退按钮: `ion-nav-back-button`
8. 视图特定按钮: `ion-nav-buttons`
9. 定制标题内容: `ion-nav-title`
10. 定制视图切换方式: `nav-transition`
11. 定制视图切换方向: `nav-direction`
12. 导航栏脚本接口: `$ionicNavBarDelegate`
13. 访问历史: `$ionicHistory`

1. Ionic 中路由管理介绍

在单页应用（Single Page App）中，路由的管理是很重要的环节。

ionic.js 没有使用 AngularJS 内置的 `ng-route` 模块，而是选择了 AngularUI 项目的 `ui-router` 模块。`ui-router` 的核心理念是将子视图集合抽象为一个状态机，导航意味着状态的切换。在不同的状态下，ionic.js 渲染对应的子视图（动态加载的 HTML 片段）就实现了路由导航的功能

ui-router:

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

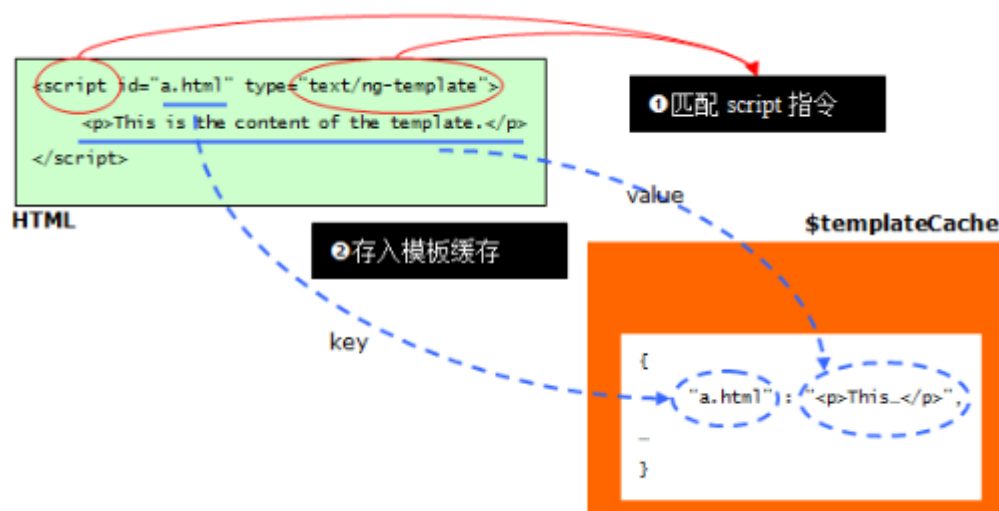
2. ionic 中内联模板介绍

可能你没有注意过,HTML 中常用的 `script` 标签在 AngularJS 中被重新定义了:除了原来的脚本声明功能之外,如果 `script` 元素的 `type` 属性定义为 `text/ng-template`,则被称为内联模板。例如:

```
<script type="text/ng-template" id="a.html">
<p>This is the content of the template.</p>
</script>
```

内联模板在单页应用 (SAP) 开发中非常有用。SAP 应用通常需要通过 AJAX 从后台载入众多的 HTML 片段,这些 HTML 片段都用文件存放的话,看起来、想起来都很不爽。使用内联模板,就可以把这些零散的 HTML 片段模板都集中在一个文件里,维护和开发的感觉都会好很多。

AngularJS 在编译时会将内联模板的 `id` 属性值和其内容,分别作为 `key` 和 `value`,存入 `$templateCache` 管理的 hash 表中:



使用内联模板

内联模板的使用,常见的有几种情况。

- 使用 **ng-include** 指令

可以利用 `ng-include` 指令在 HTML 中直接使用内联模板,例如:

```
<div ng-include="'a.html'"></div>
```

注意: 其中 `a.html` 是一个字符串常量,需要使用单引号包裹起来。

- 使用 **\$templateCache** 服务

也可以直接使用 `$templateCache` 服务的方法 `get()` 从模板缓存中读出其内容:

```
1.var partial = $templateCache.get("a.html");
```

- 使用 **\$http** 服务

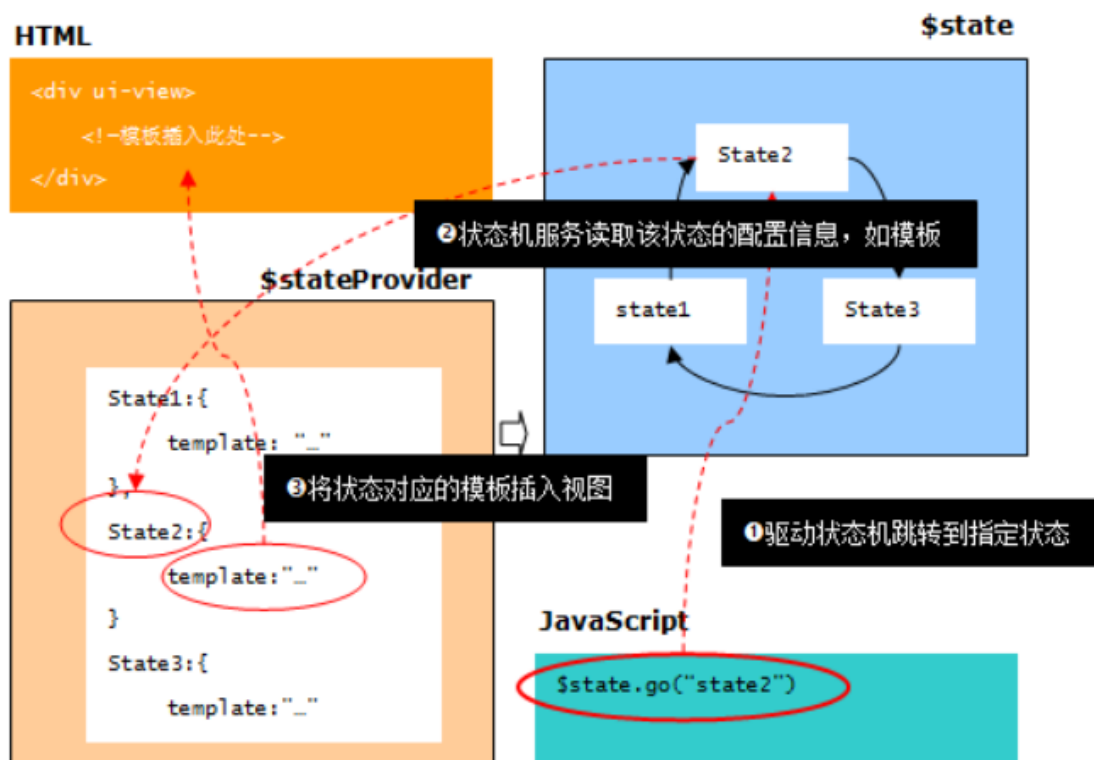
还有一种常见的用法是使用 `$http` 服务时指定 `cache` 参数，这将直接从 `$templateCache` 中取出模板，而不必进行网络访问：

```
$http.get("a.html",{cache:$templateCache})
  .success(function(d,s){..})
  .error(function(d,s){...});
```

3. ionic 路由机制：状态

对于视图的路由，ionic 没有使用 AngularJS 的路由模块(`ng-route`)，而是使用了 angular-ui 项目的 `ui-route` 模块。`ionic.bundle.js` 已经打包了 `ui-route` 模块，所以我们使用时不需要单独引入。

和通常基于 URL 匹配的路由机制不同，`ui-route` 是基于状态机的导航：



可以认为视图元素 `ui-view` 有多个状态，比如：

`state1/state2/state3`。在任何一个时刻，视图元素只能处于某一状态下。这些状态是由状态机管理的。

在 `ui-route` 中的 `$state` 服务就是一个状态机实例，在任何时刻，我们可以使用其 `go()` 方法跳转到指定名称的状态。

配置状态机

需要指出的是，状态的划分以及每个状态的元信息（比如模板、`url` 等）

是在配置阶段通过 `$stateProvider` 完成的：

```
angular.module("myApp", ["ionic"])
.config(function($stateProvider) {
    $stateProvider.state("state1", {...})
        .state("state2", {...})
        .state3("state3", {...});
});
```

触发状态迁移的几种方式（通俗的讲就是页面跳转的几种方式）

- 1) 调用 `$state.go()` 方法，这是一个高级的便利方法；
- 2) 点击包含 `ui-sref` 指令的链接 `<a ui-sref="state1">Go State1`
- 3) 导航到与状态相关联的 `url`。

当用户点击这个链接时，`$state` 服务将根据状态名 `state1` 找到对应的元信息，提取、编译模板，并将其显示在 `ui-view` 指令指定的视图窗口中。

4. 导航视图: `ion-nav-view`

在 `ionic` 里，我们使用 `ion-nav-view` 指令代替 `AngularUI Route` 中的 `ui-view` 指令，来进行模板的渲染：

```
<ion-nav-view>
<!--模板内容将被插入此处-->
</ion-nav-view>
```

和 `ui-view` 一样，`ion-nav-view` 总是根据状态的变化，来提取对应的模板并将其在 DOM 树中渲染。

5. 模板视图: `ion-view`

尽管在模板视图中可以随便写 HTML，但是，在 `ionic` 中，我们总是使用指令 `ion-view` 来作为模板视图内容的容器，这是为了与 `ionic` 的导航框架保持兼容：

```
<script id="..." type="text/ng-template">
  <ion-view>

    <!-- 模板视图内容 -->

  </ion-view>
</script>
```

`ion-view` 指令有一些可选的属性：

- `view-title`-视图标题文字

模板被载入导航视图 `ion-nav-view` 显示时，这个属性值将显示在导航栏 `ion-nav-bar` 中

- `cache-view`-是否对这个模板视图进行缓存

允许值为：`true` | `false`，默认为 `true`

- `hide-back-button`-是否隐藏导航栏中的返回按钮

当模板被载入导航视图时，如果之前有其他的模板，那么在导航栏

`ion-nav-bar` 上默认会自动显示返回按钮（使用指令

`ion-nav-back-button` 定义）。点击该按钮将返回前一个模板。

`hide-back-button` 的允许值为：`true` | `false`，默认为 `false`

注意：必须在导航栏中显式地声明返回按钮，否则即使将

`hide-back-button` 属性设为 `false`，这个按钮也不会出现：-)

- `hide-nav-bar`-是否隐藏导航栏

允许值为: `true` | `false` , 默认为 `false`

6. 导航栏: `ion-nav-bar`

`ion-nav-bar` 指令用来声明一个居于屏幕顶端的导航栏, 它的内容随导航视图的状态变化而自动同步变化:

```
<ion-nav-bar></ion-nav-bar>
```

`ion-nav-bar` 有以下可选的属性:

- `align-title`-标题对齐方式

允许值为: `left` | `right` | `center`。默认为 `center`, 居中对齐

- `no-tap-scroll`-点击导航栏时是否将内容滚动到顶部。

允许值为: `true` | `false`。默认为 `true`, 这意味着如果视图中的内容下拉很长, 那么在任何时刻点击导航栏都可以立刻回到内容的开头部分。

7. 回退按钮: `ion-nav-back-button`

ionic 的指令 `ion-nav-back-button` 指令可以自动地让你回退到前一个视图:

```
<ion-nav-bar>
  <ion-nav-back-button></ion-nav-back-button>
</ion-nav-bar>
```

当视图切换时, 回退按钮会自动出现在导航条中, 并显示前一个视图的标题。点击回退按钮将返回前一个视图。

`ion-nav-back-button` 定制样式

我们可以定制回退按钮的图标、文本和样式:

```
<ion-nav-back-button class="button-clear">
  <i class="icon ion-ios-arrow-back"></i>返回
```

```
</ion-nav-back-button>
```

8. 视图特定按钮: ion-nav-buttons

在 ionic 的导航框架中, 导航栏是公共资源。那么, 如果我们需要在不同的状态下(即载入不同的模板视图), 在导航栏上显示一些不同的按钮, 该怎么办?

答案是, 在 ion-view 指令声明的元素内使用 ion-nav-buttons 指令添加一组按钮, ionic 的导航框架看到这个指令时, 就会自动地将这些按钮安置到导航栏中。

指令 ion-nav-buttons 必须是指令 ion-view 的直接后代:

```
<ion-view>
  <ion-nav-buttons>

    <!--按钮定义-->

  </ion-nav-buttons>
</ion-view>
```

ion-nav-buttons 指令有一个属性用于声明这些按钮在导航栏中的位置:

- side-在导航条的那一侧放置按钮。允许值: primary | secondary | left | right

primary 和 secondary 是平台相关的。比如在 iOS 上, primary 被映射到左边, 而 secondary 被映射到最右边, 但是在 Android 上, primary 和 secondary 都在最右侧。left 和 right 则明确地声明是在左侧还是右侧, 与平台无关。

9. 定制标题内容: ion-nav-title

导航栏中默认显示所载入模板视图的 view-title 属性值, 但 ionic

允许我们使用 `ion-nav-title` 指令，使用任意的 HTML 片段改变它！

`ion-nav-title` 必须是 `ion-view` 的直接后代：

```
<ion-view>
  <ion-nav-title>

    <!--HTML 片段-->

  </ion-nav-title>
</ion-view>
```

10. 定制视图切换方式： `nav-transition`

视图切换时的动画转场方式，可以使用 `nav-transition` 指令声明：

1. `<any ui-sref=".." nav-transition="..">...</any>`

目前支持的转场方式有三种：

- `android -android` 模拟
- `ios -ios` 模拟
- `none` -取消转场动画

11. 定制视图切换方向： `nav-direction`

使用 `nav-direction` 指令声明视图转场时的切换方向：

1. `<any ui-sref=".." nav-direction="..">...</any>`

目前支持的选项有：

- `forward` -新视图从右向左进入
- `back` -新视图从左向右进入
- `enter` -
- `exit` -
- `swap` -

12. 导航栏脚本接口： `$ionicNavBarDelegate`

服务 `$ionicNavBarDelegate` 提供了控制导航栏的脚本接口：

- `align([direction])` - 标题对齐方式。

参数 `direction` 是可选的，允许值为：`left` | `right` | `center`，缺省值为 `center`。

- `showBackButton([show])` - 是否显示回退按钮

参数 `show` 是可选的，允许值为：`true` | `false`，缺省值为 `true`。

- `showBar(show)` - 是否显示导航栏参数 `show` 的允许值为：`true` | `false`。
- `title(title)` - 设置导航栏标题参数 `title` 为 HTML 字符串。

13. 访问历史： `$ionicHistory`

`ionic` 的导航框架会自动维护用户的访问历史栈，我们可以通过服务 `$ionicHistory` 管理访问轨迹：

- `viewHistory()` - 返回视图访问历史数据
- `currentView()` - 返回当前视图对象
- `currentHistoryId()` - 返回历史 ID
- `currentTitle([val])` - 设置或读取当前视图的标题

参数 `val` 是可选的。无参数调用 `currentTitle()` 方法则返回当前视图的标题。

- `backView()` - 返回历史栈中前一个视图对象

如果从视图 A 导航到视图 B，那么视图 A 就是视图 B 的前一个视图对象。

- `backTitle()` - 返回历史栈中前一个视图的标题
- `forwardView()` - 返回历史栈中的下一个视图对象

- `currentStateName()` - 返回当前所处状态名
- `goBack()` - 切换到历史栈中前一个视图

当然，前提是存在前一个视图。

- `clearHistory()` - 清空历史栈除了当前的视图记录，
`clearHistory()` 将清空应用的全部访问历史