# Angular 测试

# Angular测试简介

对于一个应用，只完成了基本功能开发，还远远不够，如何在快速迭代中保持稳定的产品质量，测试的重要性不言而喻。从AngularJS 1.x到Angular 2，Angular团队都非常重视代码的可测性，并且业界也有不少优秀的测试工具，这使得通过Angular开发的应用，可以游刃有余地编写单元测试及集成测试代码。

在IT行业人们在如何测试这个问题上一直争论不休。一种方法是先写测试，再写实现过程，直至测试通过；另一种是已有现实代码，再写测试，验证代码是否正确。

测试专题的官方地址： <https://www.angular.cn/guide/testing>

## 测试的意义

任何应用都可能有缺陷，测试就显得尤为重要，其意义如下：

1. 保证产品质量。
2. 提升研发效率。
3. 丰富产品说明。完备的测试用例天然就是优秀说明文档。

## 前端测试

对前端应用来说，最基本的测试无非两种：单元测试和端对端测试。

单元测试，顾名思义，就是测试程序中的某个单元，这里所指的单元，往往是一段代码，一个函数，等等，而非一个大型的功能集合。而端到端测试，可以简单概括为以用户角度来模拟实际操作行为进行的测试，例如模拟用户访问某个网站，单击按钮等等。

适用于JavaScript的测试框架有很多，常见的包括Mocha、Jasmine、QUnit等，Angular团队推荐采用Jasmine。另外，为了更好地管理测试工程，在多个浏览器上运行测试用例，还需要引入测试过程管理工具，例如Karma等。对于端到端测试，则推荐使用Protractor。

# 单元测试

单元测试是研发过程中必不可少的一个环节，它可以帮助开发人员在更早的阶段（开发甚至是代码设计阶段）发现问题，可以快速、反复地对代码逻辑进行验证。在软件工程中，越早发现问题，意味着越小的修复成本。同时，单元测试可以反应出产品本身的功能，开发者可以通过单元测试了解程序的业务逻辑。以下是编写可维护、高质量的单元测试代码的原则。

1. 将代码切分成小的可测单元。
2. 只测试公开接口。
3. 添加到持续集成中。

## Jasmine介绍

Jasmine是一款行为驱动的测试开发框架，它不依赖任何其他JavaScript框架，也不需要对DOM操作，它具有灵巧而明确的语法，可以让开发者轻松地编写测试代码。它可运行于服务器端（如Node.js）和浏览器端。Angular团队基于Jasmine扩展了一套测试语法，集成在@angular/core/testing包中。

注意：使用Jasmine测试的文件必须以 spec.ts/spec.js结尾。

官方地址：<https://jasmine.github.io/>

官方github地址： <https://github.com/jasmine/jasmine>

在Angular框架中已经内置了，如果我们在非Angular的环境中来使用的话：

npm install jasmine-core –save-dev （直接在js来使用）。

依赖：jasmine-core.js、jasmine.js、jasmine-html.js、boot.js、jasmine.css。

概念：

1. 测试集：在Jasmine中，将功能相似的测试用例（同一个组件）聚集在测试集（Suites）中， 并通过describe()函数标识。
2. 测试点：每个具体的功能测试点（Spec）可以用全局函数it()来定义，这个函数的第一个参数标识该用例的名称；第二个参数则为测试的细节。It()函数可以包含一个或多个断言（expect），每个断言的结果只能是true或者false，只有当所有断言都为true时，该测试点才算通过。
3. 内置匹配器：（Matchers）就是跟在expect()函数后面，用于判定结果是否符合预期的函数。 toBe（使用===比较结果）、toEqual（比较两个对象是否相等）、toMatch（正则表达式匹配）、toBeNull （验证是否为null）、toBeTruthy（验证是否为true）、toBeFalsy（验证是否为false）、toContain（验证数组是否包含指定元素）

其他内容：

1. beforeEach()： 定义了每个用例在执行前，所需要执行的使出化函数。可以有多个。
2. afterEach(): 每个用例在执行结束后，会来执行这个函数。
3. TestBed.configureTestingModule() 参数与@NgModule相似，功能是为测试用例创造一个生成模块环境。
4. compileComponents()， 完成外置模板以及外置样式的加载。
5. TestBed.createComponent() 参数为我们的组件引用名称，作用：返回相关组件的测试实例。
6. debugElement：获取实例对象的DOM元素对象。
7. componentInstance： 返回组件实例对应的对象实例，假设被测试类存在内部变量为title，我们可以直接通过fixture.componentInstance.title来访问变量。
8. detectChanges()： 用于当JavaScript（ts）变量及模板内容变更时触发变化监测。组件初始化、DOM元素或JavaScript（TS）值变化的时候，都需要调用这个方法，以便触发Angular的变化监测。

## Karma介绍

在Angular测试中，可以用Karma自动化管理由Jasmine编写的单元测试用例，两者相互配合，使得单元测试可以更高效地进行。Karma 是一个基于Node.js的测试执行过程管理工具（Test Runner），其前身是Google在2012年开源的Testacular。它主要带来了以下这些强大的特性。

官网： <http://karma-runner.github.io/1.0/index.html>

1. 在真实浏览器上测试：前端测试的一个目标，就是关注代码在不同浏览器的表现，检查是否有兼容性问题。Karma可以同时启动Chrome、Safari等不同的浏览器，并在一个浏览器窗口展示各个浏览器的测试结果。
2. 自动触发测试：在开发过程中，如果被测代码或者测试代码发送了变更，就能自动触发测试动作，新代码能马上得到自动验证，而且整个过程不用在多个编辑器、浏览器窗口之间切换，全由后台自动执行。
3. 丰富的插件支持：借助Karma的各类插件，可以很方便地扩展功能，如生成友好的覆盖率测试报告、支持Webpack和SystemJS打包工具等。

## Karma结合Jasmine

在传统环境下（非Angular）步骤：

1. 安装Karma并通过Karma init 命令初始化Karma配置文件。
2. 往package.json添加Karma以及Jasmine依赖。
3. 用Jasmine语法编写测试用例。

# 端到端测试

端到端测试（End to End Test，有时也简称为E2E Test）从另一个角度理解，就是模拟用户行为来进行测试。例如，模拟打开某个网页，单击指定按钮；是否按预期弹出窗口或跳转页面；提交form表单后，是否弹框提示正确的信息。端到端测试还有一个特点，就是测试代码无需依赖被测代码，对于Web应用，端到端测试就是直接测试最终的产品，即网页。由于它是模拟用户行为的一种测试，编写各种测试用例往往比较耗时，测试效率一般也会低于单元测试。也正因如此，大多数情况下大家会倾向于单元测试而非端到端测试。尽管如此，为了丰富及补充测试的手段，开发者仍需要了解如何进行端到端测试，并在合适的场景下使用端到端测试。

## Protractor介绍

Protractor是一个专门为Angular设计的端到端测试框架。Protractor提供了一些列的API，帮助开发者编写端到端的测试用例，它可以启动一个真实的浏览器进行测试。Protractor默认使用Jasmine测试框架，所以掌握了Jasmine的基本语法后，只需再学习Protractor提供的少数API，就可以很方便地编写端到端的测试用例。

实际上就是多个单元测试的集合。

常用API：

1. 打开网页： 如browser.get(xxxx)；xxx为restful类型的参数。
2. 元素选择：element.all(by.css(‘.list li’)) 返回页面上满足css规则的元素。
3. 交互动作：获取元素之后，可以通过sendKeys() 往input文本框中输入内容，可以通过click()点击元素，也可以通过getText()来获取元素。

官方文档：<http://www.protractortest.org/#/api>

# 课程内容总结