AngularJS框架测试工具的比较

作者：Nina Fat，Marijana Vujovic，Istvan Papp，Sebastian Novak

摘要——AngularJS，一个新的Javascript框架，它在web浏览器中的广泛使用和强大的表达能力，带来了一个缺点——缺乏编译器优化。因此，强烈建议用JavaScript编写的每个应用程序，不管使用的框架是什么，应该包含其行为和性能的测试。本文提出并评价了两种流行的网络自动化测试框架：Protractor和Karma。除了本文描述的测试框架之外，我们还给出了在实际系统中使用优化和非优化Web应用程序的运行测试结果。

索引术语——测试，Web自动测试，单元测试，端到端测试，Protractor，Karma

1. 介绍

最近引入了AngularJS以及用于Web应用程序动态开发的最新Web框架，如ReactJS、KnockoutJS、PolymerJS[1]。这些概念使开发人员不必编写样板式代码，从而简化了JavaScript应用程序的开发，从而将最终产品快速交付给客户。不管它们能够以多快的速度被学习和实现，最终产品在能够交付给最终用户之前，都需要通过一定程度的质量保证。这是一个挑战，因为没有静态类型，使得类型检查和IDE支持变得困难。

但是，通过提供测试与源代码的持续集成，可以克服这个挑战。有一些用于测试应用程序的Web框架，例如Jasmine，Mocha和Coffee。由于其可读性和整合测试工具的集成，我们选择了Jasmine。Jasmine测试包括套件和规格。套件包括相关的应用测试，每个测试都有一个规范。套件必须包含带有简短测试描述的describe（）函数，并且它可以包含带有it（）函数的多个规范。开发人员可以在每个it（）块中写入预期结果。该块还包含比较测试结果的匹配器，称为实际值与某些预期值。这样在测试执行后很容易生成文档，因为规范和套件内的简短字符串描述并解释了测试模块的主要功能。

为了更好地理解本文中介绍的测试方法，我们需要了解AngularJS框架。AngularJS应用程序的软件架构基于模型视图控制器模式（图1）。这意味着每个AngularJS应用程序都包含：

* 模型——负责维护数据的模式的最低级别。
* 视图——负责向用户显示全部或部分数据。在文本中，它也被称为HTML模板。
* 控制器——控制模型和视图之间的交互。

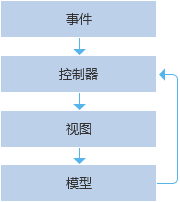


图1 模型 视图 控制器

每个AngularJS应用程序的开发都从设计一个控制器开始，该控制器通过视图控制模型，该模型会因为任何引入模型的更改[2]而更新。因此，会刷新页面的数据而不刷新整个页面。这是通过将观察器注入事件监听器来实现的。每次触发观察器时，都会启动摘要的周期，并刷新页面的数据。这在开发高效应用程序时，是至关重要。

下面几章将介绍与用于测试AngularJS应用程序的相应工具并行的每个测试方法。第三章介绍了优化AngularJS应用程序的方法和技巧。应用程序优化方法的描述包括使用每个工具测量的结果。结论中会给出详细的结果分析。

1. 单元和端到端的测试

单元和端到端测试方法之间的主要区别在于它们被允许访问的应用程序的部分不同。

单元测试可以访问模型和控制器（图1）。单元测试工具可以注入控制器，服务以及过滤器。这意味着开发人员可以独立测试应用程序的每个模块。

端到端测试只能访问视图。使用应用程序的HTML模板，可以测试主要功能，例如在单击按钮时打开新视图。端到端应用程序测试要求测试中的应用程序已完成并且正在服务器上运行。它的优点是它是类似于手动测试的方式进行测试，操作人员来测试应用程序。

1. Karma

Karma是一个使用Web服务器的工具，它针对每个浏览器测试代码的源代码[3]。Karma监视配置文件中指定的所有文件，每当任何文件发生更改时，它都会通过向测试服务器发送信号来触发测试运行。源文件在Iframe中加载，执行测试并记录结果。

测试工具可以在许多浏览器中轻松整合，并支持大多数测试框架（Jasmine，Mocha，Qunit）。开发人员只需要为需要测试的浏览器安装相应的运行器。

每个测试阶段都需要做准备。在测试任何依赖项之前，必须实例化服务和过滤器。然后创建测试规范。在规范之后，结果表达式必须与匹配器表达式匹配。匹配器通常是期望函数中使用的表达式。这些函数可以将不同的表达式与字符串、其他表达式、布尔值等进行比较。可以在每次测试之前和之后运行一系列命令（图2）。

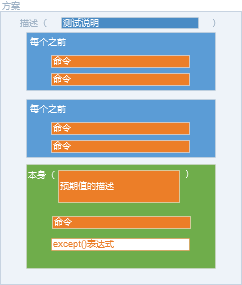


图2 Jasmine测试的展示图

1. Protractor

Protractor是一个基于Selenium Server构建的框架，由Google团队开发。 Selenium Server负责解释测试中的命令并将它们转发到一个或多个浏览器[4]。服务器和浏览器之间的通信使用WebDriver的Wire Protocol，这是一种基于JSON的协议。命令由浏览器驱动程序解释（图3）。使用Protractor作为Selenium的包装器的优势在于它增加了一种选择AngularJS元素的方法，例如绑定或中继器[8]。



图3 Protractor测试模型

Protractor的可靠性得益于它是由AngularJS团队开发的以及AngularJS的文档包含了Protractor的测试。它还提供了方便的辅助和实用的方法，例如等待页面加载等。与不同框架和不同浏览器的兼容性也是一个优点。

开发人员必须安装webdriver-manager作为帮助工具，以便运行Selenium Server的实例。另外，必须进行实例化配置，并且必须在浏览器中显示需要测试的元素。可见元素是可以测试的唯一元素类型。开发人员无法直接测试控制器，服务或过滤器。我们可以通过HTML和CSS模板间接测试这些元素。

1. 优化技术

优化是在应用程序开发的早期阶段进行的，包括以下主要思想：

1.AngularJS指令在ng-repeat中缓存数据。

2.绕过监听器。

3.在需要的地方使用一次性绑定语法

第一次优化要求测试脚本通过$index[6]跟踪ng-repeat指令使用的数组元素。使用keep track函数，用户可以深入到集合的所有项目及其相应的DOM元素中。如果将一个项目插入到集合中，则ng-repeat将知道所有其他项目已经具有DOM元素，并且它不会重现它们。由于AngularJS自动为数组提供索引，因此它在更改时被认为是次要的代码。使用$index跟踪的优点还在于每个元素都获得其唯一键，由此避免了由重复键（可以手动分配）引入的错误。

绕过应用程序代码中的监听器是必不可少的，因为对于每个监听器来说，AngularJS摘要周期都会被触发，从而延长执行时间。使用一次性绑定时会发生类似的情况，因为每次在控制器中更改值时，都会触发摘要周期并为DOM提供新值。通过使用一次性绑定语法，摘要周期仅触发一次，这会在第一次加载页面或在浏览器中刷新时。消化周期数明显减少[7]。

1. 结果

结果是在一个简单的电池应用上显示的，该应用是大型家庭自动化系统的一部分。此应用程序是一个应用程序的真实表示，它使用了所有主要的AngularJS组件，如过滤器，服务，自定义指令，控制器等。

在第一种情况下，测试应用程序初始版本的全功能。在此版本中没有进行优化，并且在开发期间未对应用程序进行测试。

第二个应用程序是在应用建议的优化后开发的。

优化应用程序用于显示测试工具对加载速度更快的应用程序以及DOM的加载速度与测试时间的关联性。

1. 单元测试的执行时间

单元测试按照Karma测试工具的说明进行。每次测试，阵列中的元素数量都会增加。使用$index，一次性绑定和使用normal指令的未优化应用程序，使用优化的ng-repeat指令测试应用程序。这种差异随着元素数量的增长而增长，结果显示ng-repeat中1000个元素的平均差异为0,303s（图4）。

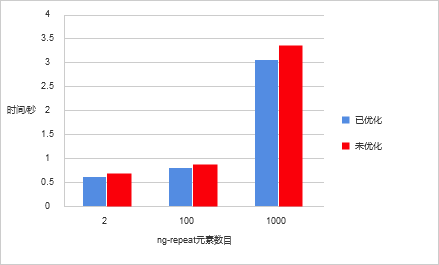


图4 单元测试期间收集的结果

作为测试工具的Karma还增加了代码覆盖功能，用户可以在其中查看已测试应用程序执行代码的百分比。在计算代码覆盖率之前，必须对部分代码进行检测（将其复制到单独的文件中作为跟踪准备），并与要测试的代码进行比较。当应用程序的可伸缩性和大小变的重要时，代码覆盖是一个很好的功能。它还使开发人员能够删除未使用的代码行。

在单元测试覆盖期间收集的结果显示应用程序版本之间没有差异（优化和未优化）。当应用程序必须处理插入ng-repeat中的更多请求时，结果才会有所不同（图5）。这是因为它需要更多分支来处理自定义指令内的数据（图像，级别）中的更多元素。

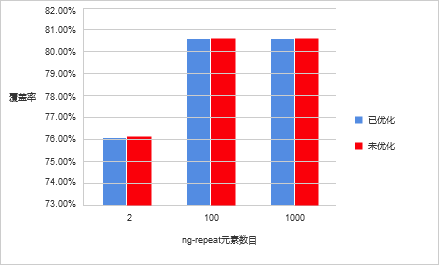


图5 覆盖测试期间的结果

1. 端到端测试的执行时

Protractor作为一种工具使我们能够深入了解整个应用程序及其功能。所以，如上所述，必须在应用中单独运行Selenium Webdriver和服务器。因此，在进行单元测试时，测试的执行速度要慢得多。执行还取决于应用程序加载所需的时间，因为Protractor在执行测试之前需要等待应用程序加载完全。Protractor的缺点在于它不提供代码覆盖，因为所有HTML代码都加载在浏览器中，并且无法访问DOM，所以无法从脚本中检测到代码。

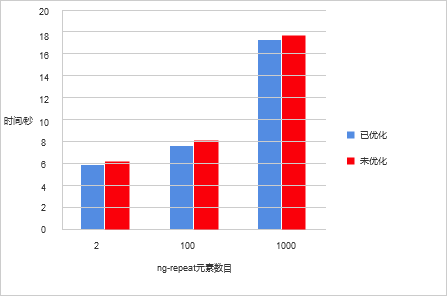


图6 端到端测试期间收集的结果

1. 可用工具选项的比较

在对应用程序进行任何测试之前，建议重新考虑工具的所有选项，以及每种测试策略的内容。

很多人认为使用Karma进行的单元测试更加灵活。通过使用断点，调试单元测试很容易。Karma提供代码覆盖以及集成的Webdriver。大多数浏览器都支持单元测试并提供自动调整功能来观察源代码的更改情况（表1）。另一方面是单元测试不能同时在多个浏览器中进行。此外，此单元测试无法测试对服务器的HTTP GET和HTTP POST请求，但可以使用ngMock模块对这些请求进行模拟。它支持将Angular服务注入和模拟到单元测试中。

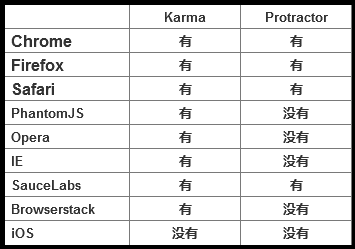


表1 支持运行测试的浏览器

与单元测试相比，端到端的测试灵活性更低，找到应用程序错误的机会更少（表2）。目前并不是所有浏览器都支持它，但它的优势是可以同时在更多浏览器中运行。虽然它没有集成的自动调整功能，但是自动调用可以集成在名为Grunt的Javascript任务运行器中，它可以自动调用而不是一个调整器工具。端到端测试还需要在测试期间单独运行Selenium Webdriver实例。由于长错误消息以及浏览器和运行测试的进程之间的分离，测试的调试可能非常复杂。一种解决方案是在测试期间捕获浏览器的屏幕截图以简化调试。另一种解决方案建议使用诸如browser.pause（）之类的函数，这些函数暂停代码的执行并在控制流中的附加一个调试点。browser.pause（）的缺点是测试不完全，用户必须小心可能出现的错误。

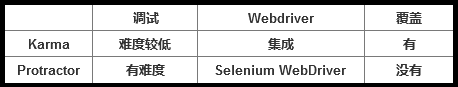


表2 比较选项

1. 结论

应用程序测试是创建可维护和可重用代码的关键组件。使用像Jasmine和Mocha那样的描述性框架来编写测试是理解功能和编写文档的良好开端。未经测试的软件会产生以下问题：复杂的DSL结构，重复的代码和不可读的代码。测试的优点是可以轻松地在更多浏览器上进行测试，而不需要不断刷新页面并使用原始的调试技术。

在对应用程序进行测试之前，建议记下所有预期的功能以及用户特定的要求。这可以在以后帮助编写更有效的测试。根据应用程序模块的大小，始终建议从单元测试开始。这些是应用程序内部结构运行良好的指标，并且能够显示应用程序是否满足主要功能。从结果（图4和图5）可以看出，单元测试可以很好地指示应用程序的优化程度，以及提供代码覆盖率。这对于编写更小和更好的应用程序的模块至关重要。跨多个浏览器进行测试也是一个优势。另一方面，端到端测试给出了应用的整体情况。它们在确定应用效率方面并不是那么重要。但它们在测试最终产品时非常重要，应用程序与所有与其连接的模块和单元一起工作。它们的主要优点是它们可以帮助将更多模块与单元测试集成在一起。如此，无论应用程序的大小和复杂性如何，应用程序都可以完全正常运行。