Technical design Quizzer

Carlos Kelkboom

August 11, 2014

Abstract

This document is written as a compendium to the Quizzer solution. Most of the technical design decisions and requirements are explained.

Contents

1 Introduction

The quizzer application is an open source project which contains all the necessary application parts to create an extensible testing suit and framework. Separate from this design goal the Quizzer application can also serve as a template for building smart web applications. These concepts are not restricted to building browser based applications.

2 AngularJS

At the root of our application and architecture lies AngularJS[1]. This framework for building web applications and web sites focusses around an MVC type approach to binding your data to your views.

There are many JavaScript MVC frameworks out there and there are more build every day. The reason I chose AngularJS is that when I started writing single page applications I looked for the most flexible system. A system which did not superimpose rules on how to build your application. I find that every time a framework tries to impose a structure on your application you will always run into the limitations of this structure and as a result your solution and architecture will become brittle and maintainability will suffer.

I have been working with AngularJS for over a year now and have not found any such restrictions. Ive used AngularJS with an ASP.NET MVC backend with a ruby and rails backend but my favourite remains NodeJS[2]. NodeJS is just like AngularJS flexible enough to build everything I need. Furthermore, when building single page web applications NodeJS gives me the simplicity needed to create even the most difficult of tasks spread over multiple micro-services with almost laughable ease.

This is something different

2.1 Controls

Within big enterprise applications it is customary to create small snippets which represent controls and their structure. The forms chapter is dedicated to the actual controls used in the Quizzer application. This part will explain the need for and the solutions used to create a custom control set with AngularJS.

Within software architecture we have two prevailing ideas about creating forms. The first one states that the HTML written in the views should represent the actual HTML as rendered on the client. The second states that the HTML written in the views should be declarative and a process, usually a rendering engine, should transform the declarative markup to the actual HTML. Both ideas are based on solid arguments; with this application we've chosen to implement the second idea and create a custom set of directives to render our form fields in the browser. The reason for taking this approach is that we can now create a single point of definition for each controls, are can now with a verifiable degree of certainty state that every control which uses the directive will be rendered comparably.

[1] https://angularjs.org/

[2] NodeJS will be explained in a following chapter

3 Forms

Forms are a big part of the application. Your forms need to be consistent and stylable. The way the Quizzer application solves this problem is by creating AngularJS directives per form field. This way you can guarantee that each control of the same type will be rendered in exactly the same way.

This section will explain the implementation and usage of each of these controls.

3.1 Text

The textbox is the main component and the first well look at within this document.

```
// comment

var foo = function(x, y) {
    return x + y;
}
```

```
<!-- usage -->
<ck-text-m></ck-text-m>
```

4 Validation

Validation is done using both AngularJS, my custom form controls and HAPI's validation module.

5 Conclusion

Write your conclusion here.

Query