Bind Angularjs to newly created html element dynamically

I have a tab page with multiple tabs that once clicked on call a service to return some data. Some of that data returns html forms and are very random. I want to collect those values that are entered and send the data via a service back to the server. The problem I have is that I can't get the data from the input elements in the html I'm creating dynamically.

I've created a Plunker to show what the issue is. Note that the html value can change at any time so hard-coding the html won't work. Here the code from the plunker, but please look at the plunker for the best view of whats going on.

app.js

```
var app = angular.module('plunker', []);

app.controller('MainCtrl', function($scope, $sce, $compile) {
    $scope.name = 'World';
    $scope.html = "";

    $scope.htmlElement = function(){
        var html = "<input type='text' ng-model='html'></input>";
        return $sce.trustAsHtml(html);
    }

});
```

index.html

```
<!DOCTYPE html>
<html ng-app="plunker">

  <head>
    <meta charset="utf-8" />
    <title>AngularJS Plunker</title>
    <script>document.write('<base href="' + document.location + '" />');</script>
    <link rel="stylesheet" href="style.css" />
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.0-rc.3/angular.js">
</script>
    <script src="app.js"></script>
  </head>

  <body ng-controller="MainCtrl">
    <p>Hello {{name}}!</p>

    <div ng-bind-html="htmlElement()"></div>

    {{html}}

  </body>

</html>
```

angularjs model-binding

asked Nov 8 '13 at 19:59
Ty Danielson
444 1 7 16

Why are you making DOM modifications within the controller rather than a directive? – David Chase Nov 8 '13 at 21:41

This is only for example. The actual html is coming from the server via a service. No DOM manipulation in the controller. I have been wondering if I could use a directive and $compile to get this to compile the html and bind the html to the model at that point. – Ty Danielson Nov 8 '13 at 22:08

3 Exactly you can use `$compile(element.contents())(scope)` , before that you can add string to the `element.html()` method and setup a `$watch` on the `ngModel` i dont know if thats what you are going for... – David Chase Nov 8 '13 at 23:51

2 Answers

One solution would be to use ngInclude with $templateCache, as demonstrated in this Plunker.

There are a couple things to note.

The first is that you can fetch your template using a service and add it to the $templateCache, as described here (example copied):

```
myApp.service('myTemplateService', ['$http', '$templateCache', function ($http,
$templateCache) {
  $http(/* ... */).then(function (result) {
    $templateCache.put('my-dynamic-template', result);
  });
}]);
```

Then you can include it in your template as follows:

```
<div ng-include="'my-dynamic-template'"></div>
```

ngInclude will allow databinding on the html string, so you don't need ngBindHtml.

The second is that, as ngInclude creates a new scope, accessing the `html` property outside of the newly created scope won't work properly unless you access it via an object on the parent scope (e.g. `ng-model="data.html"` instead of `ng-model="html"` . Notice that the `$scope.data = {}` in the parent scope is what makes the html accessible outside of the ngInclude scope in this case.

(See this answer for more on why you should always use a dot in your ngModels.)

Edit

As you pointed out, the ngInclude option is much less useful when using a service to return the HTML.

Here's the edited plunker with a directive-based solution that uses $compile, as in David's comment above.

The relevant addition:

```
app.directive('customHtml', function($compile, $http){
  return {
    link: function(scope, element, attrs) {
      $http.get('template.html').then(function (result) {
        element.replaceWith($compile(result.data)(scope));
      });
    }
  }
})
```

edited Nov 9 '13 at 0:18

answered Nov 8 '13 at 21:38
Sarah
961 7 9

I think this is getting closer, but doesn't really do it because the template still needs to be applied when the app if first run and can't be set at a later time like when the service call has completed. I edited your plunker to show you that the service call won't work. – Ty Danielson Nov 8 '13 at 22:56

Indeed :) Updated above. – Sarah Nov 9 '13 at 0:20

@Sarah - thanks so much. I was looking for something just like this and your Plunker example is fantastic. I forked your plunker and added a way to append multiple template files which will append to the element. You can see it at: plnkr.co/edit/Ch5uykLwlF8Td2zQEoH9?p=preview – daylight Feb 7 '14 at 18:50

Based on Sarah's answer, i created a structure to put the directive

```
.directive('dynamic', function(AmazonService, $compile) {
  return {
    restrict: 'E',
    link: function(scope, element, attrs) {
      AmazonService.getHTML()
     .then(function(result){
       element.replaceWith($compile(result.data)(scope));
     })
     .catch(function(error){
       console.log(error);
     });
   }
  };
});
```

And in the html:

```
<dynamic></dynamic>
```

Thanks Sarah, helped a lot!!!

answered Jul 22 '14 at 20:45

victorkurauchi
346 3 10