Fill a select with ng-options in AngularJS

MAXIM VAN VEEN 30-03-2018

You can use ng-repeat on a option tag to fill a select with options in AngularJS but AnguarJS provides the ng-options method to do this.
So how does this work?

There is a difference between building a select with the values coming from an array and an object.
If your values are coming from an array you can do `value for value in array`.

So if you have an array like this `['one', 'two', 'three', 'four', 'five']` you'll get the following result.

```
<select class="ng-pristine ng-valid ng-touched" ng-options="value for value in array" ng-model="r
    <option value="?" selected="selected" label=""></option>
    <option value="0" label="one">one</option>
    <option value="1" label="two">two</option>
    <option value="2" label="three">three</option>
    <option value="3" label="four">four</option>
    <option value="4" label="five">five</option>
</select>
```

What you're doing here is in the first word "value" you're determine what is going to be in the label and as text in the options. The second word "value" is the variable for the loop. So the first and the second word needs to be the same. I've used value but you can give it your own name.
And the third word, in this case "array" is the variable name coming from the controller.

I did use the same word as first and second argument in this example. So the difference may not be really clear. So I'll give you an example of what the difference is.
The first argument is what is going to be in the label, this could be just the value but it could also be a string with the value. The second argument is just the value.

So if I pass the following into the ng-options with the same array. `'option ' + value + ' is' for value in array`. The result will be:

```
<select class="ng-valid ng-touched ng-dirty ng-valid-parse" ng-options="'option ' + value + ' is'
    <option value="?" selected="selected" label=""></option>
    <option value="0" label="option one is">option one is</option>
    <option value="1" label="option two is">option two is</option>
    <option value="2" label="option three is">option three is</option>
    <option value="3" label="option four is">option four is</option>
    <option value="4" label="option five is">option five is</option>
</select>
```

Although the value of the option is the key and the label and text is what I've filled in in the first argument the model of this select is just the value.
So when I put `{{ result }}` in the HTML the output will be `one`, `two`, `three`, `four` or `five` instead of `option one is`, `option two is`, `option three is` etc.

So that is how you loop over an array with the ng-options method. But besides arrays you can loop over objects as well.
I use the following object for this example:

```
{
    first: ['one', 'uno', 'een', 'eins'],
    second: ['two', 'due', 'twee', 'zwei'],
    third: ['three', 'tre', 'drie', 'drei'],
    fourth: ['four', 'quattro', 'vier', 'vier']
}
```

When looping over an object you'll need a key and a value as second argument. This is how the ng-options attribute looks like for the object `value for (key, value) in object`. And the result will be

```
<select class="ng-valid ng-dirty ng-valid-parse ng-touched" ng-options="value for (key, value) in
```

```
<select class="ng-valid ng-dirty ng-valid-parse ng-touched" ng-options="value for (key, value) in
    <option value="?" selected="selected" label=""></option>
    <option value="first" label="one,uno,een,eins">one,uno,een,eins</option>
    <option value="fourth" label="four,quattro,vier,vier">four,quattro,vier,vier</option>
    <option value="second" label="two,due,twee,zwei">two,due,twee,zwei</option>
    <option value="third" label="three,tre,drie,drei">three,tre,drie,drei</option>
</select>
```

The first argument is what is going to be in the label and as text in the option. The second is the key and the value in the object and the last is again the variable coming form the controller.
When you change the first argument to key instead of value the label and the text of the option is going to be the key of the object. So in this case it's going to be:

```
<select class="ng-valid ng-dirty ng-valid-parse ng-touched" ng-options="key for (key, value) in o
    <option value="?" selected="selected" label=""></option>
    <option value="first" label="first">first</option>
    <option value="fourth" label="fourth">fourth</option>
    <option value="second" label="second">second</option>
    <option value="third" label="third">third</option>
</select>
```

In both of these cases the output in the model of this select will be `["one","uno","een","eins"]` when choosing the first option.
If you want the key as output you can add an extra argument to the loop. This will be `key as (...) for` .

Again the argument before the word `for` is what is going to be in the label and as text.
So `key as key for (key, value) in object` will give this result:

```
<select class="ng-valid ng-dirty ng-valid-parse ng-touched" ng-options="key as key for (key, valu
    <option value="?" selected="selected" label=""></option>
    <option value="first" label="first">first</option>
    <option value="fourth" label="fourth">fourth</option>
    <option value="second" label="second">second</option>
    <option value="third" label="third">third</option>
</select>
```

and {{result}} will be `first` when choosing the first option.

And `key as value for (key, value) in object` will give this result:

```
<select class="ng-valid ng-dirty ng-valid-parse ng-touched" ng-options="key as value for (key, va
    <option value="?" selected="selected" label=""></option>
    <option value="first" label="one,uno,een,eins">one,uno,een,eins</option>
    <option value="fourth" label="four,quattro,vier,vier">four,quattro,vier,vier</option>
    <option value="second" label="two,due,twee,zwei">two,due,twee,zwei</option>
    <option value="third" label="three,tre,drie,drei">three,tre,drie,drei</option>
</select>
```

again {{result}} will be `first` when choosing the first option.

In this loop `key as value for (key, value) in object` the first argument is what is going to be in the `ng-model` of this

In this loop `key as value for (key, value) in object` the first argument is what is going to be in the `ng-model` of this select, the second is going to be the text the third is the separation of the key and the value of the object items and the fourth is the name of the object in your controller.

I used just an array for every object item in this example. The value of `first` was `['one', 'uno', 'een', 'eins']`. But what if we put an object inside that object.
So the object is going to look like this:

```
{
    first: {
        name: 'the first',
        count: ['one', 'uno', 'een', 'eins']
    },
    second: {
        name: 'the second',
        count: ['two', 'due', 'twee', 'zwei']
    },
    third: {
        name: 'the third',
        count: ['three', 'tre', 'drie', 'drei']
    },
    fourth: {
        name: 'the fourth',
        count: ['four', 'quattro', 'vier', 'vier']
    }
}
```

We can use that object as output. For example if we do `value as key for (key, value) in object`. The label and text of the first option will be `first` and the output will be `{"name":"the first","count":["one","uno","een","eins"]}`.

But when you want the 'name' key of every object as text in the option you can do value.name to put that in there. You can do that for the output and for what is going to be in the select.
To make that more clear, `key as value.name for (key, value) in object` will fill the select options with `the first`, `the second`, `the third` and the output will be `first`, `second`, `third`. And when doing `value.name as key for (key, value) in object`, the select options will be filled with `first`, `second`, `third` and the output will be `the first`, `the second`, `the third`.

Of course you can't fill the options with the entire object. When you try to do that it will result in `[object Object]`.

Now you may have noticed that every select has a

```
<option value="?" selected="selected" label=""></option>
```

as first option. This is because we didn't set a default option that has to be selected before making a choice.
If you want one of the options to be pre selected when loading the page or view you can do it with ng-init. In there you can set the model of that select to the value that has to be selected.

For example I want to select the first option of that array that I used as example

```
['one', 'two', 'three', 'four', 'five']
```

I'll add `ng-init="result = array[0]"` to the select. The code will be

```
<select ng-init="result = array[0]" ng-options="value for value in array" ng-model="result">
```

and the first option will be selected.

For the object there is a little bit more to take into account.
For the object the ng-init needs to be the same as what is going to be the output.

If you use this object:

```
{
    first: {
        name: 'the first',
        count: ['one', 'uno', 'een', 'eins']
    },
    second: {
        name: 'the second',
        count: ['two', 'due', 'twee', 'zwei']
    },
    third: {
        name: 'the third',
        count: ['three', 'tre', 'drie', 'drei']
    },
    fourth: {
        name: 'the fourth',
        count: ['four', 'quattro', 'vier', 'vier']
    }
}
```

And in ng-options you have `value as value.name for (key, value) in object`, the first is value and so it is the entire object inside the object.

```
first: {
    name: 'the first',
    count: ['one', 'uno', 'een', 'eins']
}
```

is the first and

```
second: {
    name: 'the second',
    count: ['two', 'due', 'twee', 'zwei']
}
```

is the second.

This means that the ng-init needs to be that. And so you'll fill in `ng-init="result = object['first']"` to select the first.

If you have `value.name as value.name for (key, value) in object` inside the `ng-option` the `ng-init` needs to be `result = object['first'].name`. And if you use the key as output.
Like this `key as value.name for (key, value) in object` your ng-init is going to be `result = 'first'`.

The code will look like this:
`<select ng-init="result = 'first'" ng-options="key as value.name for (key, value) in object" ng-model="result">`.

SHARE

LEAVE A REPLY

NAME*	EMAIL*

COMMENT*

SEND

YOU MIGHT ALSO LIKE

Migrate AngularJS to Angular using UpgradeModule (Part 6)

DANNY CORNELISSE 17-12-2018

Migrate AngularJS to Angular using UpgradeModule (Part 5)

DANNY CORNELISSE 10-12-2018

Migrate AngularJS to Angular using UpgradeModule (Part 4)

DANNY CORNELISSE 10-12-2018

VISIT US

Verrijn Stuartlaan 20
2288 EL Rijswijk (ZH)
The Netherlands

PHONE US

+31 (0)70 42 77 555

EMAIL US

info@competa.com

FOLLOW US