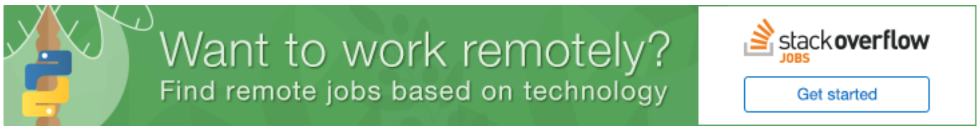
angular ng-repeat skip an item if it matches expression



I'm looking for a way to basically tell angular to skip an item in an ng-repeat if it matches an expression, basically `continue`;

In controller:

```
$scope.players = [{
    name_key:'FirstPerson', first_name:'First', last_name:'Person'
}, {
    name_key:'SecondPerson', first_name:'Second', last_name:'Person'
}]
```

Now in my template I want to show everyone that doesn't match `name_key='FirstPerson'`. I figured it has to be filters so I setup a Plunkr to play around with it but haven't had any luck. Plunkr Attempt

angularjs angularjs-ng-repeat

asked Nov 5 '13 at 7:30
aron.duby
979 2 7 17

I try to understand : do you want to show all items but point who match and who is not? or just filter? – Maxim Shoustin Nov 5 '13 at 7:41

3 Answers

As @Maxim Shoustin suggested, the best way to achieve what you want would be to use a custom filter.
But there are other ways, one of them being to use the `ng-if` directive on the same element were you put the `ng-repeat` directive (also, here's the plunker):

```
<ul>
    <li ng-repeat="player in players" ng-if="person.name_key!='FirstPerson'"></li>
</ul>
```

This may present a minor disadvantage from an estetical perspective, but has a major advantage that your filtering could be based on a rule that is not as tight coupled to the `players` array and that can easily access other data in your app's scope:

```
  <li
      ng-repeat="player in players"
      ng-if="app.loggedIn && player.name != user.name"
  ></li>
```

Update
As stated, this is **one of the solutions** for this kind of problem and may or may not suit your needs.
As pointed out in the comments, `ng-if` is a directive, which actually means that it might do more things in the background than you might expect.
For example, `ng-if` creates a new scope from it's parent:

> The scope created within ngIf inherits from its parent scope using prototypal inheritance.

This usually doesn't affect the normal behaviour but in order to prevent unexpected cases, you should keep this in mind before implementing.

edited Jul 28 at 14:28 answered Nov 5 '13 at 7:48

gion_13
26.1k 6 68 78

this was exactly what I was looking for, knew there had to be some short and sweet way to do it without the custom filter. Thanks! – aron.duby Nov 6 '13 at 0:01

Note: there are some scope behaviors of ng-if that can be tricky. I tried this method and it caused issues elsewhere. Namely, I have a "watcher" directive that broadcasts an event when the repeater is finished. After adding the ng-if to this repeater it no longer fired that event. A custom filter might actually be the cleanest way to go. – claywhipkey May 6 '15 at 15:42

@claywhipkey you're right, the cleanest way is using a filter, but that isn't suitable for all the cases. Anyway, thx for the comment and you can see that I've added a *heads up* update in the answer just to make anyone aware of the implications of using a directive for filtering data. – gion_13 May 7 '15 at 9:36

I know this is an old one, but in case someone would look for another possible solution, here is another way to solve this - use standard filter functionality:

> *Object:* A pattern object can be used to filter specific properties on objects contained by array. For example {name:"M", phone:"1"} predicate will return an array of items which have property name containing "M" and property phone containing "1". ... **The predicate can be negated by prefixing the string with !.** For example {name: "!M"} predicate will return an array of items which have property name not containing "M".

So for the TS example something like this should do:

```
<ul>
    <li ng-repeat="player in players | filter: { name_key: '!FirstPerson' }"></li>
</ul>
```

No need to write custom filters, no need to use `ng-if` with it's new scope.

edited Oct 15 at 17:06 answered Jul 9 '15 at 12:15
Ilya Luzyanin
3,979 2 15 29

Simple to the point. No custom filter or new scope created as author has stated. Perfect for skipping a single value in a list. – mbokil Aug 12 '15 at 0:37

I think you need to omit the 'player' from 'player.name_key' - so it would just be "{name_key: '!FirstPerson' }" in the above example. – smithml Aug 20 '15 at 1:01

I was never able to get Angular to play nice with empty keys. Is there some sort of trick for using the built-in filters for the equivalent of `player in players | filter: { name_key: '' }` ? That particular directive won't match only players with no/an empty `name_key` . The inverse, `name_key: '!'` intended to select any player with a non-empty `name_key` also won't work. – Eirik Aug 25 '15 at 16:04

You can check this answer. – Ilya Luzyanin Aug 25 '15 at 16:15

1 Beware that this only works with arrays, not with object properties in a case such as `ng-repeat="(key, val) in player"` – David Diez Jul 28 at 10:06

You can use **custom** filter when you implement `ng-repeat` . Something like:

```
 data-ng-repeat="player in players |  myfilter:search.name
```

`myfilter.js` :

```
app.filter('myfilter', function() {


   return function( items, name) {
    var filtered = [];

    angular.forEach(items, function(item) {

      if(name == undefined || name == ''){
        filtered.push(item);
        }

      /* only if you want start With*/
      // else if(item.name_key.substring(0, name.length) !== name){
      //   filtered.push(item);
      // }

      /* if you want contains*/
      // else if(item.name_key.indexOf(name) < 0 ){
      //   filtered.push(item);
      // }

       /* if you want match full name*/
       else if(item.name_key !== name ){
        filtered.push(item);
      }
    });

    return filtered;
  };
});
```

Demo Plunker

edited Jan 18 at 14:44
Community ♦
1 1

answered Nov 5 '13 at 7:40
Maxim Shoustin
46.7k 16 143 170

Thanks for the work you put into this. Its definitely correct but I ended up going with the ease of @gion_13 answer in the code so I figured I better mark his as the accepted answer – aron.duby Nov 6 '13 at 0:05