

Connect AngularJS to Servers

Examples on GitHub





agenda

- Making XHR requests
- Accessing to REST API
- Overcoming to CORS



MAKING XHR REQUESTS



\$http

Core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.

```
$http({method: 'GET', url: '/someUrl'})  
  .success(function(data, status, headers, config) {  
    // this callback will be called asynchronously  
    // when the response is available  
  })  
  .error(function(data, status, headers, config)  
    // called asynchronously if an error occurs  
    // or server returns response with an error status.  
  });
```

Success asynchr



Configuring request

- ▮ **Add some authorization headers**
- ▮ **Setting cache**
- ▮ **Transforming the request or response**

```
$http({  
  method: string,  
  url: string,  
  params: object,  
  data: string or object,  
  headers: object,  
  transformRequest: function transform(data, headersGetter),  
  transformResponse: function transform(data, headersGetter),  
  cache: boolean or Cache object,  
  timeout: number,  
  withCredentials: boolean  
});
```



Dedicated methods

One method for each type of XMLHttpRequest

```
GET: $http.get(url, config)
POST: $http.post(url, data, config)
PUT: $http.put(url, data, config)
DELETE: $http.delete(url, config)
HEAD: $http.head(url, config)
```



GET vs POST methods

Request data from a specified resource

```
///// HTTP GET /////  
$http.get('http://localhost:3000/api/topics/', {cache: false})  
  .success(function(data) {  
    hvCtrl.topics = data;  
  })  
  .error(function(data, status){  
    showError(data, status);  
  });
```

Submit data to be processed to a specified resource

```
///// HTTP POST /////  
$http.post('http://localhost:3000/api/topics/', newTopic)  
  .success(function(data) {  
    loadTopics();  
    hvCtrl.newTopicTitle = '';  
  })  
  .error(function(data, status){  
    showError(data, status);  
  });
```



Example

POSTMAN
\$http



Chained promises

Using promises \$http calls can be chained

```
$http.get('/api/user/?login='+userProfile.login)
  .then(function(response) {
    // Store the userid, get the permissions.
    userProfile.userID = response.data;
    return $http.get('/api/profile/' + userProfile.userID);
  })
  .then(function(response) {
    // Store the permissions, now get the list.
    userProfile.permissions = response.data;
    return $http.get('/api/lists/' + userProfile.permissions);
  })
  .then(function(response) {
    // Show something
    console.log("The full list is " + response.data);
  })
  .catch(function(error) {
    console.log("An error occurred: " + error);
  });
```

Only if resolve

Any promise error



ACCESS TO REST API



\$resource service

A factory which creates a resource object that lets you interact with RESTful server-side data sources.

The returned resource object has action methods which provide high-level behaviors without the need to interact with the low level \$http service.



Declaration in factory

Calling the injected \$resource function

```
angular.module('tutorialConnectApp', [  
  'ngRoute',  
  'ngResource',  
  'tutorialConnectApp.homeView',  
  'tutorialConnectApp.httpView',  
  'tutorialConnectApp.coreView',  
  'tutorialConnectApp.re  
])  
.factory('Topics', ['$resource', function($resource) {  
  return $resource('http://localhost\\:3000/api/topics/:topicid', null,  
    {  
      'update': { method: 'PUT', params: { topicid: '@_id' } },  
      'remove': { method: 'DELETE', params: { topicid: '@_id' } }  
    });  
}]);
```

Careful with ':'

Custom methods



Benefits of ngResource

- **Simplified code (encapsulation)**
- **No callbacks (unless wanted)**
- **Unit tests with ngResource**
- **Custom methods**



Example

\$resource



Problems of ngResource

- Any custom behavior on the client a lot of work
- Once you get the data you can not a lot, so you should deliver it in its final state
- Does not return promises
- Building custom URLs is not easy

Check Restangular

<https://github.com/mgonto/restangular>

More to come in Angular 2.0



OVERCOMING CORS



Overcoming same-origin policy

Cross-origin Resource Sharing (CORS) allows to get a resource from another domain, forbidden by browsers.

- External resources**
- Different domain or port**

Solutions

- 1. Modify server**
- 2. JSONP**



Allow all origin

Modify server so it can be access by all origins.

```
Remote Address: 127.0.0.1:3000
Request URL: http://localhost:3000/api/topics/541464cd632cdb0c6ced0517
Request Method: OPTIONS
Status Code: 204 No Content
▼ Request Headers view source
  Accept: */*
  Accept-Encoding: gzip,deflate,sdch
  Accept-Language: es,en-US;q=0.8,en;q=0.6,ca;q=0.4,de;q=0.2
  Access-Control-Request-Headers: accept, content-type
  Access-Control-Request-Method: PUT
  Connection: keep-alive
  Host: localhost:3000
  Origin: http://localhost:3001
  Referer: http://localhost:3001/
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
▼ Response Headers view source
  Access-Control-Allow-Headers: accept, content-type
  Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
  Access-Control-Allow-Origin: *
  Connection: keep-alive
  Date: Sat, 13 Sep 2014 15:40:51 GMT
  X-Powered-By: Express
```



Example

Modify Server

Access-Control-Allow-Origin: *



JSONP

JSONP = JSON with padding

Request data from a server in a different domain, taking advantage of the fact that browsers do not enforce the same-origin policy on `<script>` tags.

```
<script type="text/javascript"  
    src="...../api/?callback=angular.callbacks._1">  
</script>
```



Example

JSONP



JSONP limitations

- **Only GET HTTP Request**
- **Error handling is problematic**
- **Security threads**
- **Modify my NodeJS to accept JSONP**

MANY THANKS

http://about.me/Carlos_Morales

<https://github.com/carlos-/ajs-connectserver>