LoginService module

1.Introduction

"LoginService" module is a service module which is mainly used for sending JSON data via AJAX request. We registered a service called "loginService" to LoginService module via module#service API.

-loginService

We inject $http service, which is one of several built-in angular services, into "loginService" for making requests and handling responses from web server.

Inside the "loginService" we have a method which expects to have two arguments, the request URL parameters and the POST data. This method has a success and an error method. Once http request is successful, the JSON data is assigned to "loginService.response". Otherwise it assigns a post error message to "loginService.response". After each request, return "loginService.response".

2.Test

Here we define a test suit called "loginServiceTest".

```
describe('loginService',function  () {
        var loginService, $httpBackend;
        var user = {'userName': 'test',
                        'password': 'test123'};
        beforeEach(module("LoginService"));
        beforeEach(inject(function  (_loginService_, _$httpBackend_) {
                loginService = _loginService_;
                $httpBackend = _$httpBackend_;

        }));
})
```

1. define loginService and $httpBackend variable.
2. define the user data for the test.
3. before each test we tell Angular to load the LoginService module.
4. before each test inject loginService and $httpBackend by using underscore way.

*$httpBackend is a built-in service. We can use this service to simulate requests in our tests.

-Post success test

Once the loginService posts the data, the web server responses a JSON data with http status 201 code which means that request is successful. After that, loginService assigns the JSON data to the response object and returns it.

In this test the response object is {'result': 0, 'rid': 1,'uid': 2,'token': ‘sdfkdkqikdkkqe', 'error_msg': ‘pasword'}.

Here for the test, we expect one of the response objects named ‘result’ to be 0.

```
it('should login success', function() {
        var url ='login/';
        var respond = {
                'result': 0,
                'rid': 1,
                'uid': 2,
                'token': 'sdfkdkqikdkkqe',
                'error_msg': 'pasword'}

        $httpBackend.expectPOST('http://chanmao.ca/?r=%20rrclient/login/',
user).respond(201, respond);
        loginService.post(user,url);
        console.log(loginService.test)
        $httpBackend.flush();
        expect(loginService.post(user,url).result).toBe(0);
   }),
```

Here we define a test called “should login success”

1. define URL variable.
2. define a respond object which web server will respond.
3. use $httpBackend.expectPOST method to train the $httpBackend service to expect an incoming HTTP request and tell it what to respond with.
4. The response is not returned until we call the $httpBackend.flush method.
5. Here we simulate the response at ‘http://chanmao.ca/?r=%20rrclient/login/‘ and send back the 201 http status code and the JSON object which we define in the above.
6. flush the request queue.
7. expect one of the response objects named ‘result’ to be 0.

-Post fail test

Once the loginService posts the data, the web server responses a JSON data with http status 401 code which means that request fails. After that, loginService assigns a post error message to the response object and returns it.

In this test the post error message is “post error!”

Here for the test, we expect the response to be “post error!”.

```
it('should be ERROR!', function() {

                        var respond = {
                           'result': 0,
                           'rid': 1,
                           'uid': 2,
                           'token': 'sdfkdkqikdkkqe',
                           'error_msg': 'pasword'};

                $httpBackend.expectPOST('http://chanmao.ca/?r=%20rrclient/
login',user).respond(401, respond);
                loginService.post(user);

                $httpBackend.flush();
                expect(loginService.post(user)).toBe("post error!");


});
```

Here we define a test called “should login success”

1. define URL variable.
2. define a respond object which web server will respond.
3. use $httpBackend.expectPOST method to train the $httpBackend service to expect an incoming HTTP request and tell it what to respond with.
4. The response is not returned until we call the $httpBackend.flush method.
5. Here we simulate the response at ‘http://chanmao.ca/?r=%20rrclient/login/‘ and send back the 401 http status code and the JSON object which we define in the above.
6. flush the request queue.
7. expect the response to be “post error!”.

3. Code

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


 LoginService.service('loginService', [ '$http', function( $http) {
          var loginService = this;
          loginService.test ="this is a test";
          loginService.post = function(data, url) {
          $http.post('http://chanmao.ca/?r=%20rrclient/' + url, user)
                    .success(function(response) {
                         loginService.response = response;
                    }).error(function() {
                         loginService.response = "post error!";
                    });
                        return loginService.response;
           };
}]);
```

Here we define an angular module called LoginService

1. define a angular service in LoginService module called loginService.
2. Inject $http service a angular built-in service.
   * Since we use “controller as” syntax, we do not need to inject $scope service.
3. define this loginService to a variable.
4. define a method called loginService.post which expects two arguments, data object and URL.
5. use $http post method to post the data
6. once request is successful handle the response and assign it to loginService.response.
7. once request fails assigns a message "post error!” to loginService.response.
8. return the loginService.response.