This repository Search Pull requests Issues Gist

witoldsz / **angular-http-auth**

Watch 129 Unstar 2,368 Fork 416

Code Issues 18 Pull requests 9 Wiki Pulse Graphs

Branch: master angular-http-auth / src / **http-auth-interceptor.js** Find file Copy path

witoldsz prevent repeating `event:auth-loginRequired` #95 #120 #130 1e26e44 Mar 23, 2016

10 contributors

136 lines (123 sloc) | 4.5 KB Raw Blame History

```
/*global angular:true, browser:true */

/**
 * @license HTTP Auth Interceptor Module for AngularJS
 * (c) 2012 Witold Szczerba
 * License: MIT
 */
(function () {
  'use strict';

  angular.module('http-auth-interceptor', ['http-auth-interceptor-buffer'])

  .factory('authService', ['$rootScope','httpBuffer', function($rootScope, httpBuffer) {
    return {
      /**
       * Call this function to indicate that authentication was successfull and trigger a
       * retry of all deferred requests.
       * @param data an optional argument to pass on to $broadcast which may be useful for
       * example if you need to pass through details of the user that was logged in
       * @param configUpdater an optional transformation function that can modify the
       * requests that are retried after having logged in.  This can be used for example
       * to add an authentication token.  It must return the request.
       */
      loginConfirmed: function(data, configUpdater) {
        var updater = configUpdater || function(config) {return config;};
        $rootScope.$broadcast('event:auth-loginConfirmed', data);
        httpBuffer.retryAll(updater);
      },

      /**
       * Call this function to indicate that authentication should not proceed.
       * All deferred requests will be abandoned or rejected (if reason is provided).
       * @param data an optional argument to pass on to $broadcast.
       * @param reason if provided, the requests are rejected; abandoned otherwise.
       */
      loginCancelled: function(data, reason) {
        httpBuffer.rejectAll(reason);
        $rootScope.$broadcast('event:auth-loginCancelled', data);
      }
    };
  }])

  /**
   * $http interceptor.
   * On 401 response (without 'ignoreAuthModule' option) stores the request
   * and broadcasts 'event:auth-loginRequired'.
   * On 403 response (without 'ignoreAuthModule' option) discards the request
   * and broadcasts 'event:auth-forbidden'.
   */
  .config(['$httpProvider', function($httpProvider) {
    $httpProvider.interceptors.push(['$rootScope', '$q', 'httpBuffer', function($rootScope, $q, httpBuffer) {
      return {
        responseError: function(rejection) {
          var config = rejection.config || {};
          if (!config.ignoreAuthModule) {
            switch (rejection.status) {
              case 401:
                var deferred = $q.defer();
                var bufferLength = httpBuffer.append(config, deferred);
                if (bufferLength === 1)
                  $rootScope.$broadcast('event:auth-loginRequired', rejection);
                return deferred.promise;
              case 403:
                $rootScope.$broadcast('event:auth-forbidden', rejection);
                break;
            }
          }
          // otherwise, default behaviour
          return $q.reject(rejection);
        }
      };
    }]);
  }]);

  /**
   * Private module, a utility, required internally by 'http-auth-interceptor'.
   */
  angular.module('http-auth-interceptor-buffer', [])

  .factory('httpBuffer', ['$injector', function($injector) {
    /** Holds all the requests, so they can be re-requested in future. */
    var buffer = [];

    /** Service initialized later because of circular dependency problem. */
    var $http;

    function retryHttpRequest(config, deferred) {
      function successCallback(response) {
        deferred.resolve(response);
      }
      function errorCallback(response) {
        deferred.reject(response);
      }
      $http = $http || $injector.get('$http');
      $http(config).then(successCallback, errorCallback);
    }

    return {
      /**
       * Appends HTTP request configuration object with deferred response attached to buffer.
       * @return {Number} The new length of the buffer.
       */
      append: function(config, deferred) {
        return buffer.push({
          config: config,
          deferred: deferred
        });
      },

      /**
       * Abandon or reject (if reason provided) all the buffered requests.
       */
      rejectAll: function(reason) {
        if (reason) {
          for (var i = 0; i < buffer.length; ++i) {
            buffer[i].deferred.reject(reason);
          }
        }
        buffer = [];
      },

      /**
       * Retries all the buffered requests clears the buffer.
       */
      retryAll: function(updater) {
        for (var i = 0; i < buffer.length; ++i) {
          var _cfg = updater(buffer[i].config);
          if (_cfg !== false)
            retryHttpRequest(_cfg, buffer[i].deferred);
        }
        buffer = [];
      }
    };
  }]);
})();
```