ANGULARJS
ELÉMENTS AVANCÉS

SOMMAIRE

SQLI GROUP

LES INTERCEPTEURS

LES INTERCEPTEURS

QU'EST-CE C'EST ?

+ Permet d'intercepter
  - Une **requête http** pour faire des traitements avant l'envoi au serveur
  - Une **réponse http** pour faire des traitements avant l'envoi au client

+ Peut être utile pour des problématiques d'authentification, de gestion d'erreur, de trace, etc.

+ Mis en œuvre sous forme de **services**, enregistrés auprès du provider **$httpProvider** via sa propriété **interceptors**

+ Basés sur le **service $q** (implémentation des **promesses**)

LES INTERCEPTEURS

LES TYPES

- Un intercepteur doit implémenter au moins l'un des 4 types suivants
  - **request** : appelé avec un objet http **config** en paramètre ; doit renvoyer un objet http **config** en retour, éventuellement en tant que promesse
  - **requestError** : appelé lorsqu'un intercepteur précédent est en erreur
  - **response** : appelé avec un objet http **response** en paramètre ; doit renvoyer un objet http **response** en retour, éventuellement en tant que promesse
  - **responseError** : appelé lorsqu'un intercepteur précédent est en erreur

LES INTERCEPTEURS

DÉCLARATION

```
// register the interceptor as a service
$provide.factory('myHttpInterceptor', function($q, dependency1, dependency2) {
  return {
    request: function(config) {
      return config;
    },
    requestError: function(rejection) {
      return $q.reject(rejection);
    },
    response: function(response) {
      return response;
    },
    responseError: function(rejection) {
      return $q.reject(rejection);
    }
  };
});

$httpProvider.interceptors.push('myHttpInterceptor');
```

LES INTERCEPTEURS

EXEMPLE 1

+ Ajout d'un paramètre de type jeton d'authentification à toutes les requêtes qui partent vers le serveur

```
myapp.factory('httpRequestInterceptor', function ($cookieStore) {
  return {
    request: function (config) {
      var token = $cookieStore.get("auth");
      config.url =  URI(config.url).addSearch({'_auth_token':token}).toString();
      return config;
    }
  };
});

myapp.config(function ($httpProvider) {
  $httpProvider.interceptors.push('httpRequestInterceptor');
});
```

LES INTERCEPTEURS

EXEMPLE 2

+ Reroutage d'un utilisateur non authentifié (401 – Unauthorized)

```
myapp.factory('myHttpResponseInterceptor', ['$q', '$location',
  function($q, $location) {
    return {
      responseError: function(response) {
        if (response.status === 401) {
          $location.path('/signin');
        }
        return $q.reject(response);
      }
    }
  }]);

myapp.config(['$httpProvider',function($httpProvider) {
  $httpProvider.interceptors.push('myHttpResponseInterceptor');
}]);
```

SQLI GROUP

EXERCICES PRATIQUES

TP14 – INTERCEPTEUR D'ERREUR

INTERCEPTER LES ERREURS RÉSEAUX ET AFFICHER UN MESSAGE D'INFORMATION

- ✓ implémenter le service intercepteur
- ✓ enregistrer le ($httpProvider)
- ✓ modifier l'url du backend pour provoquer l'erreur

LES ÉVÈNEMENTS

LES ÉVÈNEMENTS

PRÉSENTATION

+ Propagation au travers de la hiérarchie des **scopes**, de façon ascendante ou descendante

+ 2 caractéristiques
  - Un **nom unique**
  - Une **liste d'arguments**, qui peut être nulle

+ Utilisés par ex. pour la gestion des **routes**

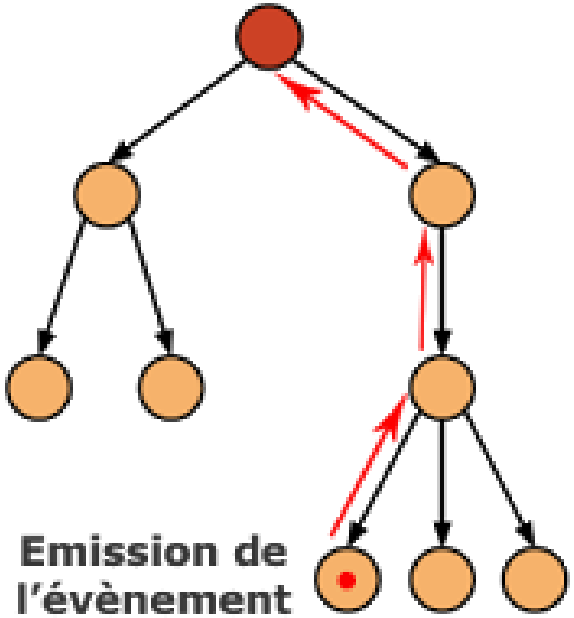
  $routeChangeStart, $routeChangeSuccess, $routeChangeError, $routeUpdate, ...

+ Déclenchés et interceptés via des fonctions sur les objets de type **scope** : **$emit**, **$broadcast**, **$on**

LES ÉVÈNEMENTS

$EMIT

- Emission **ascendante** vers les **scopes parents directs**
- L'évènement peut être **intercepté** et sa **propagation** éventuellement **stoppée**
- **$emit(name, args)**
  - **name** : nom de l'évènement
  - **args** (optionnel) : arguments de l'évènement



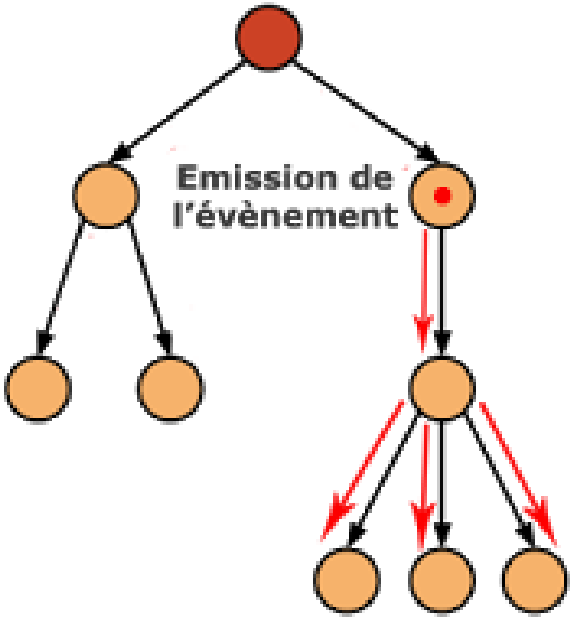
LES ÉVÈNEMENTS

$BROADCAST

+ Emission **descendante** vers **tous les scopes enfants**

+ L'évènement peut être **intercepté** mais sa **propagation ne peut pas être stoppée**

+ **$broadcast(name, args)**
  - **name** : nom de l'évènement
  - **args** (optionnel) : arguments de l'évènement



LES ÉVÈNEMENTS

$ON

- Déclare un **écouteur** sur un évènement
- **$on(name, listener)**
  - **name** : nom de l'évènement à intercepter
  - **listener** : fonction appelée lors de l'interception de l'évènement, au format **function(event, args)**

    Propriétés de l'objet event :
    - targetScope : scope à l'origine de l'évènement
    - currentScope : scope courant
    - name : nom de l'évènement
    - **stopPropagation()** : met fin à la propagation de l'évènement (pour $emit)
    - preventDefault() : positionne defaultPrevented à true
    - defaultPrevented : renvoie la valeur booléenne

LES ÉVÈNEMENTS

EXEMPLE

```
<div ng-controller="EventController">
  Root scope <tt>MyEvent</tt> count: {{count}}
  <ul>
    <li ng-repeat="i in [1]" ng-controller="EventController">
      <button ng-click="$emit('MyEvent')">$emit('MyEvent')</button>
      <button ng-click="$broadcast('MyEvent')">$broadcast('MyEvent')</button>
      <br />
      Middle scope <tt>MyEvent</tt> count: {{count}}
      <ul>
        <li ng-repeat="item in [1, 2]" ng-controller="EventController">
          Leaf scope <tt>MyEvent</tt> count: {{count}}
        </li>
      </ul>
    </li>
  </ul>
</div>
```

$rootScope

Child Scope

Child Scope

```
function EventController($scope) {
  $scope.count = 0;
  $scope.$on('MyEvent', function() {
    $scope.count++;
  });
}
```

SQLI GROUP

EXERCICES PRATIQUES

TP15 - EVÉNEMENTS

TRAITER L'AFFICHAGE DU MESSAGE D'INFORMATION DANS LE CODE DE L'APPLICATION ET NON DANS L'INTERCEPTEUR

- ✓ $broadcast depuis l'intercepteur
- ✓ $on dans l'application

FIN