Teodoro Montanaro (teodoro.montanaro@polito.it)

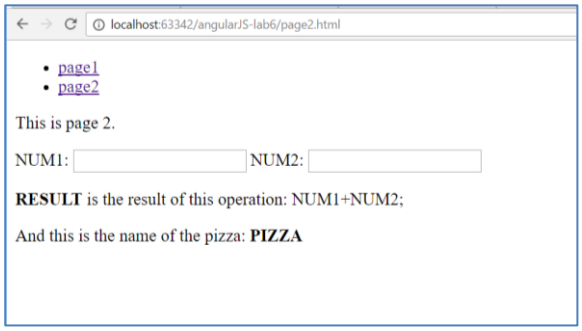
LAB 6 – MIGRAZIONE AD ANGULARJS

In questa esercitazione partiremo da una semplice applicazione web realizzata con HTML e Javascript per migrarla passo passo verso una single page application realizzata usando AngularJS.

ESERCIZIO 1 – WEB APPLICATION CON HTML, CSS E JAVASCRIPT

1. Creare 2 pagine HTML simili a quelle mostrate di seguito:

Page 1



Page 2

Fare in modo che i link puntino alle 2 pagine (esempio: page 1)

2. Copiare il file pizza.json nella cartella del nostro progetto.
   Il file è scaricabile a questo link:
   https://elite.polito.it/files/courses/01QYAPD/2017/labs/06-MigrationToAngularJS/pizza.json

3. Aggiungere, in un file script.js, il seguente codice JavaScript.

   Il codice fa in modo che:
   a) la scritta NUM1 venga sostituita con il contenuto del primo campo di testo ogni volta che questo cambia
   b) la scritta NUM2 venga sostituita con il contenuto del secondo campo di testo ogni volta che questo cambia
   c) RESULT venga sostituito con la somma di NUM1 e NUM2 ogni volta che questi cambiano
   d) La scritta "PIZZA" venga sostituita con "margherita" (prendendo il valore tramite AJAX da pizza.json).

Teodoro Montanaro (teodoro.montanaro@polito.it)

```
window.onload = function()
{
    var num1 = document.getElementById("NUM1");
    num1.setAttribute("onchange", "calcola();");
    num1.setAttribute("onkeyup", "calcola();");
    var num2 = document.getElementById("NUM2");
    num2.setAttribute("onchange", "calcola();");
    num2.setAttribute("onkeyup", "calcola();");
    calcola();
    acquisisciPizza();
}

function calcola()
{
    var num1 = document.getElementById("NUM1").value;
    var num2 = document.getElementById("NUM2").value;
    var sum = parseInt(num1) + parseInt(num2);
    document.getElementById("result").innerHTML = sum;
    document.getElementById("NUM1Span").innerHTML = num1;
    document.getElementById("NUM2Span").innerHTML = num2;
}

function acquisisciPizza()
{
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function(){
        if (xhr.readyState == 4)
        {
            if ((xhr.status >= 200 && xhr.status < 300) || xhr.status == 304)
            {
                var response = xhr.responseText;
                var pizza = JSON.parse(response);
                document.getElementById('pizza').innerHTML = pizza.name;
            }
            else
            {
                alert("Request was unsuccessful: " + xhr.status);
            }
        }
    };
    xhr.open("get", "pizza.json", true);
    xhr.setRequestHeader('Access-Control-Allow-Headers', '*');
    xhr.send(null);
}
```

Teodoro Montanaro (teodoro.montanaro@polito.it)

ESERCIZIO 2 – MIGRAZIONE AD ANGULARJS

Adesso è il momento di utilizzare AngularJS. Il semplice sito realizzato nel primo esercizio possiamo realizzarlo molto più velocemente utilizzando AngularJS.

Prima di tutto, la funzione calcola() presentata sopra è del tutto superflua in AngularJS, bastano infatti 2 direttive per copiare automaticamente i valori presenti nei campi di testo al posto di NUM1 e NUM2 e farne la somma.

In più, tramite il **routing**, non abbiamo bisogno di creare 2 pagine simili: guardando le 2 pagine realizzate nell'esercizio precedente ci accorgiamo che il "menu" è uguale! Non ha senso quindi ricrearlo in tutte le pagine del nostro sito! Basta creare un template e poi dire ad AngularJS di inserire le diverse parti in uno spazio predefinito.

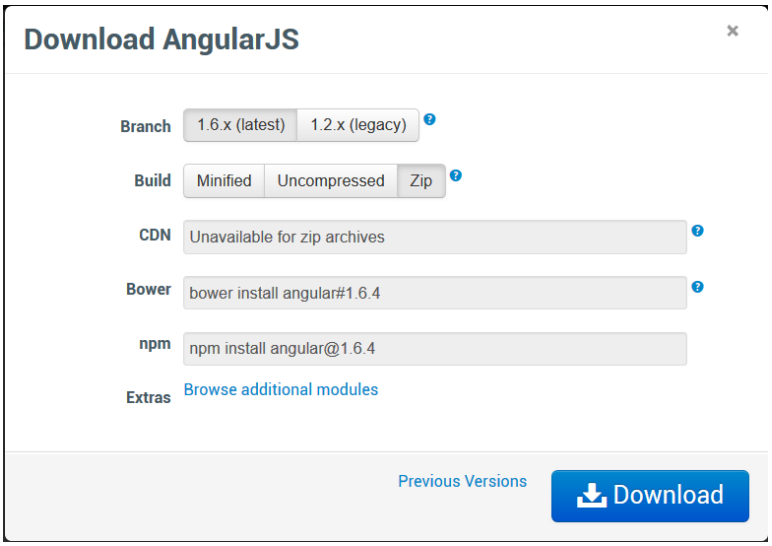
Infine, tramite il servizio $http, AngularJS semplifica l'acquisizione di contenuti da siti/pagine/script/file esterni, e quindi anche dal nostro pizza.json.

Andiamo quindi per gradi.

ESERCIZIO 2 – MIGRAZIONE AD ANGULARJS – PARTE 1

Cominciamo a modificare la pagina2 dell'esercizio precedente in modo che utilizzi AngularJS per il calcolo della somma dei numeri inseriti nelle caselle di testo.

1. Integriamo AngularJS in page2.html:
    a) scarichiamo AngularJS dal sito (scarichiamo la versione zip che contiene tutto quello che ci serve)

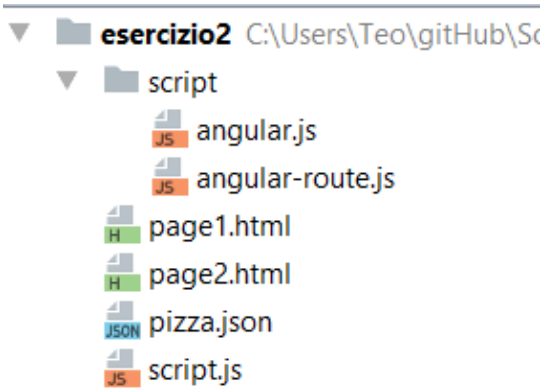


    b) Scompattiamo il contenuto dell'archivio e prendiamo dal suo interno i file:
        - angular.js
        - angular-route.js

Teodoro Montanaro (teodoro.montanaro@polito.it)

c) Torniamo nel nostro progetto, creiamo una cartella di nome “script” ed incolliamo i 2 file appena copiati lì dentro

A questo punto nel nostro progetto avremo questa situazione:



d) Importiamo questi script in page2.html, aggiungendo le 2 righe riportate di seguito

```
<script src="script/angular.js"></script>
<script src="script/angular-route.js"></script>
```

e) Diciamo ad AngularJS di “intervenire” su tutta la pagina aggiungendo la direttiva **ng-app** all’elemento principale (html):

```
<html lang="en" ng-app>
```

2. Ora diciamo ad AngularJs di assegnare i valori inseriti in NUM1 e NUM2 a delle variabili (dati.num1 e dati.num2) presenti nello $scope (il nostro modello dati)
   a. Aggiungiamo alla casella di testo con id “NUM1” la direttiva **`ng-model="dati.num1"`**
   b. Aggiungiamo alla casella di testo con id “NUM2” la direttiva **`ng-model="dati.num2"`**

3. Inseriamo i valori contenuti nelle variabili dati.num1 e dati.num2 nei posti giusti della pagina:
   a. Al posto della scritta RESULT inseriamo **{{** dati.num1 + dati.num2 **}}**
   b. Al posto di NUM1 e NUM2 inseriamo, rispettivamente, {{ dati.num1}} e {{ dati.num2}}

ATTENZIONE: ricordiamoci di rimuovere il nostro script (script.js) dalle dipendenze della pagina, altrimenti, non trovando più gli elementi del DOM che manipolavamo, il sistema genererà una serie di errori.

Come ci aspettavamo, siamo riusciti a fare tutti i calcoli che prima facevamo in calcolo() con pochissime modifiche.

Teodoro Montanaro (teodoro.montanaro@polito.it)

ESERCIZIO 2 – MIGRAZIONE AD ANGULARJS – PARTE 2

Ora è il momento di implementare il **routing.**

Guardando il nostro esempio, avevamo 2 pagine molto simili in cui c'era il menu sempre uguale e una parte centrale variabile. Con il routing possiamo creare un template che contiene tutte le parti che non cambiano e poi usare AngularJS per caricare al centro della pagina ogni volta una vista diversa.

a. Per farlo cominciamo a creare un **template (template.html)** che contiene a) tutte le parti non variabili tra le 2 pagine e, b) al centro (la parte che cambia) un div vuoto:

```
<!DOCTYPE html>
<html lang="en" ng-app>
    <head>
        <meta charset="utf-8">
        <title>My SoNet App</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <script src="script/angular.js"></script>
        <script src="script/angular-route.js"></script>
    </head>
    <body>
        <ul class="menu">
            <li><a href="page1.html">page1</a></li>
            <li><a href="page2.html">page2</a></li>
        </ul>
        <div>

        </div>
    </body>
</html>
```

Adesso vogliamo che nella parte centrale (nel div vuoto) vengano caricate di volta in volta delle "**viste**" diverse.

Ma quale vista dobbiamo caricare ogni volta? Sarà l'utente a stabilirlo: nella seconda parte dell'URL ci sarà un parametro che indicherà quale vista caricare.
Quindi, se vogliamo caricare

- la vista 1: il link sarà *template.html**#!/view1***
- la vista 2: il link sarà *template.html**#!/view2***

Per farlo usiamo AngularJS

1. Diciamo ad AngularJS quale parte del template va sostituita con le diverse viste: aggiungiamo quindi la direttiva **ng-view** al div vuoto che abbiamo lasciato al centro della pagina.

```
...
        <ul class="menu">
            <li><a href="page1.html">page1</a></li>
            <li><a href="page2.html">page2</a></li>
        </ul>
        <div ng-view>
...
```

Teodoro Montanaro (teodoro.montanaro@polito.it)

2. Creiamo le viste che di volta in volta AngularJS dovrà caricare al centro della pagina
    a. modifichiamo il file page1.html lasciando solo il seguente contenuto

```
<p>This is page 1.</p>
```

    b. modifichiamo il file page2.html lasciando solo il seguente contenuto

```
<p>This is page 2.</p>
NUM1: <input type="number" id="NUM1" ng-model="dati.num1" value="2"/>
NUM2: <input type="number" id="NUM2" ng-model="dati.num2" value="3"/>
<p>
   <b id="result">{{dati.num1+dati.num2}}</b> is the result of this
operation: {{dati.num1}}+{{dati.num2}};</p>
<p>
  And this is the name of the pizza: <b id="pizza">PIZZA</b>
</p>
```

3. Nel template modifichiamo i link alle 2 pagine

```
<ul class="menu">
   <li><a href="template.html#!/view1">page1</a></li>
   <li><a href="template.html#!/view2">page2</a></li>
</ul>
```

4. Creiamo un nuovo script che conterrà la logica della nostra app AngularJS: logica.js. E lo mettiamo nella cartella script linkandolo nel template

```
<script src="script/logica.js"></script>
```

5. La documentazione di AngularJs ci dice che per collegare al template lo script appena creato dobbiamo utilizzare i moduli. In particolare ne creiamo uno che interverrà su tutta l'applicazione:

    a. Scegliamo un nome per il modulo che stiamo implementando (moduloEsempio)
    b. Andiamo in template.html e associamo il modulo all'app, modificando la dichiarazione fatta con la direttiva ng-app:

```
<html lang="en" ng-app="moduloEsempio">
```

    c. Nel file logica.js scriviamo il seguente codice (scritto seguendo la documentazione di AngularJS relativa a ngRoute):

Teodoro Montanaro (teodoro.montanaro@polito.it)

```
angular.module("moduloEsempio", ['ngRoute'])

.config(['$routeProvider', function ($routeProvider) {
    $routeProvider
        .when('/view1', {
            templateUrl: 'page1.html'
        })
        .when('/view2', {
            templateUrl: 'page2.html'
        })
        .otherwise({redirectTo: '/view1'});
}]);
```

ESERCIZIO 2 – MIGRAZIONE AD ANGULARJS – PARTE 3

Ora rimane solo da caricare il nome della pizza dal file pizza.json.

Nel primo esercizio lo facevamo utilizzando AJAX, in AngularJS, invece, dobbiamo usare il servizio $http.

1. Ricordando cosa abbiamo visto a lezione, ogni parte "logica" delle nostre app va messa dentro degli specifici controller.
   Quindi per prima cosa creiamo un nuovo controller nel nostro file "logica.js":

```
angular.module("moduloEsempio", ['ngRoute'])
...
.controller('Ctrl', ['$scope',
    function($scope) {
        $scope.dati = {}; //inizializziamo i dati
        //logica da implementare
    }
]);
```

2. Il controller non viene utilizzato automaticamente: bisogna dire ad AngularJS in quale punto della pagina utilizzarlo.
   Noi vogliamo usarlo solo nella parte della pagina caricata dinamicamente (quella con la direttiva ng-view) e solo quando viene caricata la seconda vista, quindi modifichiamo lo script in questo modo:

```
angular.module("moduloEsempio", ['ngRoute'])

.config(['$routeProvider', function ($routeProvider) {
    $routeProvider
        .when('/view1', {
            templateUrl: 'page1.html'
        })
        .when('/view2', {
            templateUrl: 'page2.html',
            controller: 'Ctrl'

        })
        .otherwise({redirectTo: '/view1'});
}])
...
```

Teodoro Montanaro (teodoro.montanaro@polito.it)

3. Ora manca solo la logica. Vogliamo che la nostra app carichi in maniera **asincrona** il contenuto del file pizza.json, per questo creiamo un nuovo servizio aggiungendo al nostro script la parte che, nella seguente finestra, è evidenziata in grassetto:

```
angular.module("moduloEsempio", ['ngRoute'])

.config(['$routeProvider', function ($routeProvider) {
    $routeProvider
        .when('/view1', {
            templateUrl: 'page1.html'
        })
        .when('/view2', {
            templateUrl: 'page2.html',
            controller: 'Ctrl'

        })
        .otherwise({redirectTo: '/view1'});
}])
.factory('Pizza', function($http) {
    var pizzaService = {
        getData: function () {
            return
$http.get('./pizza.json').then(function (response) {
                return response.data;
            });
        }
    };
    return pizzaService;
})
.controller('Ctrl', ['$scope',
    function($scope) {
        $scope.dati = {}; //inizializziamo i dati
        //logica da implementare
    }
]);
```

Cosa fa questo codice? Crea un servizio di nome Pizza che dipende da un altro servizio ($http) e che implementa una funzione getData. Questa funzione si preoccupa di effettuare una chiamata http REST di tipo GET per recuperare il contenuto di pizza.json.

4. Penultimo tassello: dobbiamo modificare il controller in modo che recuperi i dati tramite la funzione implementata nel servizio. Modifichiamo il controller in questo modo:

```
angular.module("moduloEsempio", ['ngRoute'])
..
.controller('Ctrl', ['$scope', 'Pizza',
    function($scope, Pizza) {
        Pizza.getData().then(function(data) {
            $scope.dati.pizza = data.name;
        });
    }
]);
```

Teodoro Montanaro (teodoro.montanaro@polito.it)

5. Infine, andiamo in page2.html e sostituiamo PIZZA con {{dati.pizza}}.