



第九讲 Ajax和promise服务



河北师范大学软件学院
Software College of Hebei Normal University

教学目标

1. 使用Ajax服务
2. 使用promise服务

1

使用Ajax服务

①

使用Ajax

Ajax是现代web应用程序的基础，可以使浏览器不加载新内容而与服务器进行通信。**XHR**(XMLHttpRequest) 是 AJAX 的基础。

AngularJS对XMLHttpRequest对象进行了封装



\$http服务

AngularJS 中的核心服务,读取远程服务器的数据

②

产生Ajax请求

可产生Ajax请求的\$http服务的方法

方法	描述
<code>get(URL,config)</code>	为指定的URL执行get请求
<code>post(URL,data,config)</code>	为指定的URL执行post请求提交指定数据
<code>delete(URL,config)</code>	为指定的URL执行delete请求
<code>put(URL,data,config)</code>	为指定的URL和数据执行put请求
<code>head(URL,config)</code>	为指定的URL执行head请求
<code>jsonp(URL,config)</code>	执行get请求获取js代码片段并执行。JSONP表示json和填充（json with padding），是绕过浏览器对js代码能被载入的限制的工作方式。

②

产生Ajax请求

\$http服务常用于产生和处理Ajax请求，它是被异步执行的标准HTTP请求, **\$http服务**的**get()**方法可以返回一个promise对象。

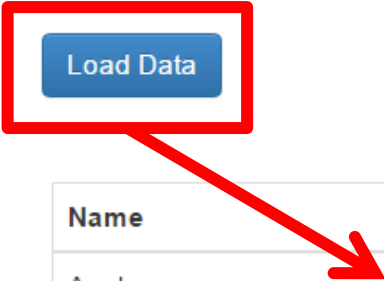
```
$http.get("productData.json")  
  .success(function (data) {  
    $scope.products = data;  
  });
```

Demo:listing 01.html

2

产生Ajax请求

Name	Category	Price
No Data		



Load Data

Name	Category	Price
Apples	Fruit	\$1.20
Bananas	Fruit	\$2.42
Pears	Fruit	\$2.02
Tuna	Fish	\$20.45
Salmon	Fish	\$17.93
Trout	Fish	\$12.93

Load Data


Demo:listing01.html

3

接收Ajax响应

Demo:listing01.html

```
$http.get("productData.json")  
  .success(function (data) {  
    $scope.products = data;  
  });
```



由\$http服务方法返回的promise
对象的方法

3

接收Ajax响应

\$http服务方法返回的promise对象的方法

方法	描述
success(fn)	当HTTP请求成功完成时，调用指定的函数
error(fn)	当请求无法成功完成时，调用指定函数
then(fn,fn)	注册成功子函数和失败函数

Demo:listing02.html

3

接收Ajax响应 使用promise对象的then方法

使用promises对象的then方法可以把success和error函数都注册在一个方法中调用，它提供了获取更详细的关于从服务器而来的响应信息。

then方法传入的对象的属性

名称	描述
data	从请求中返回数据
status	由服务器返回的HTTP状态码
headers	可被用于获取头部函数
config	用于产生请求的配置对象

Demo:listing03.html

3

接收Ajax响应

使用then方法时，AngularJS自动处理JSON数据。

使用其他格式的数据需要进行解析。

例如：xml文件

```
<products>
  <product name="Apples" category="Fruit" ...../>
  <product name="Bananas" category="Fruit" ..... />
  <product name="Pears" category="Fruit" ..... />
  <product name="Tuna" category="Fish" ..... />
  <product name="Salmon" category="Fish" ..... />
  <product name="Trout" category="Fish" ..... />
</products>
```

Demo:producData.xml

3

接收Ajax响应 处理其他类型的数据

```
$http.get("productData.xml").then(function (response)
{
    $scope.products = [];
    var productElems =
angular.element(response.data.trim()).find("product");
    for (var i = 0; i < productElems.length; i++) {
        var product = productElems.eq(i);
        //将获取的每一个product对象添加到数组
        $scope.products.push({
            name: product.attr("name"),
            category: product.attr("category"),
            price: product.attr("price")
        });
    }
});
```

Demo:listing04.html

4

配置Ajax请求

angular是基于json的ajax调用。当服务是基于非json提交方式(其他自定义数据格式)，那只能改变ng内部\$http默认request/response格式转化方式。

\$http方法的属性配置

名称	描述
transformRequest	用于在请求发送到服务器前的操作
transformResponse	用于当响应从服务器到达时的操作
data	设置发送到服务器的数据（自动转json）
headers	设置请求头部
url	为请求设置url

4

配置Ajax请求

\$http服务中的**transformResponse**属性用来转换响应。

```
▼ Object ⓘ  
  ► headers: Object  
    method: "GET"  
  ▼ transformRequest: Array[1]  
    ► 0: (d)  
      length: 1  
    ► proto : Array[0]  
  ▼ transformResponse: Array[1]  
    ► 0: (data)  
      length: 1  
    ► __proto__: Array[0]  
  uri: "productData.json"  
  ► __proto__: Object
```

默认返回的服务器响应

Demo:listing05.html

4

配置Ajax请求

```
$scope.loadData = function () {
```

```
    var config = {  
        transformResponse: function (data, headers){  
            //转换函数  
        }  
    };  
};
```

```
    $http.get("productData.xml", config )  
        .success(function (data) {  
            $scope.products = data;  
        });  
}
```

Demo:listing06.html

4

配置Ajax请求

\$http服务中的**transformRequest**属性用来转换请求。

▼ Object 

▶ headers: Object
method: "GET"

▼ transformRequest: Array[1]

▶ 0: (d)

length: 1

▶ __proto__: Array[0]

▼ transformResponse: Array[1]

▶ 0: (data)

length: 1

▶ __proto__: Array[0]

url: "productData.json"

▶ __proto__: Object

默认返回的服务器请求

④

配置Ajax请求

Demo:listing07.html

```
$scope.loadData = function () {.....};
$scope.sendData = function() {
  var config = {
    headers: {
      "content-type": "application/xml "
    },
    transformRequest: function (data, headers) {
      //使用jqLite生成XML
    }
  };
  $http.post("ajax.html", $scope.products, config);
}
```

应指定头部，让ng知道如何序列化数据



4

配置Ajax请求

Demo:listing07.html

The screenshot shows the Chrome DevTools network panel. The file list on the left includes Listing%2007.html, angular.js, bootstrap.css, productData.json, and ajax.html. The 'ajax.html' file is selected, and a red arrow points from it to the 'Request Payload' section of the 'Headers' tab. The 'Request Payload' section shows the following XML data:

```
<products><product name="苹果" category="水果" price="1.2"></product></products><products><product name="香蕉" category="水果" price="2.42"></product></products><products><product name="梨" category="水果" price="2.02"></product></products><products><product name="金枪鱼" category="鱼" price="20.45"></product></products><products><product name="鲑鱼" category="鱼" price="17.93"></product></products><products><product name="鳕鱼" category="鱼" price="12.93"></product></products>
```

附加的请求数据

5

设置默认的Ajax

\$http服务的提供器**\$httpProvider**为Ajax请求定义默认设置

\$http是在普通控制器上或者服务上注入使用。

\$httpProvider是在module的config中注入使用。

```
angular.module( "exampleApp" , [])  
  
.config(function ($httpProvider) {  
  console.log($httpProvider.defaults.headers.common);  
})
```

设置默认的Ajax

\$httpProvider的属性

名称	描述
defaults.headers.common	定义用于所有请求的默认头部
defaults.headers.post	定义用于POST请求的头部
defaults.headers.put	定义用于PUT请求的头部
defaults.transformResponse	应用于所有响应的转换函数的数组
defaults.transformRequest	应用于所有请求的转换函数的数组
interceptors	拦截器工厂函数的数组

5

设置默认的Ajax

输出angular默认的头信息: [Demo:listing08.html](#)

```
angular.module("exampleApp", [])  
  .config(function($httpProvider) {  
    console.log($httpProvider.defaults.headers.common)  
  })  
  .controller("defaultCtrl", function ($scope, $http) {  
    $scope.loadData = function () {  
      $http.get("productData.xml").success(function(data){  
        $scope.products = data;  
      });  
    };  
  });  
});
```

► Object {Accept: "application/json, text/plain, */*"}

5

设置默认的Ajax

```
angular.module("exampleApp", [])  
  .config(function($httpProvider) {  
    $httpProvider.defaults.transformResponse  
      .push(function (data, headers) {  
        //defaults.transformResponse被定义为数组，所以必须用push方法  
      });  
  })  
  .controller("defaultCtrl", function ($scope, $http) {  
    $scope.loadData = function () {  
      $http.get("productData.xml").success(function (data) {  
        $scope.products = data;  
      });  
    }  
  });
```

Demo:listing09.html

使用Ajax拦截器

\$httpProvider.**interceptors**是\$http服务的**请求拦截器**

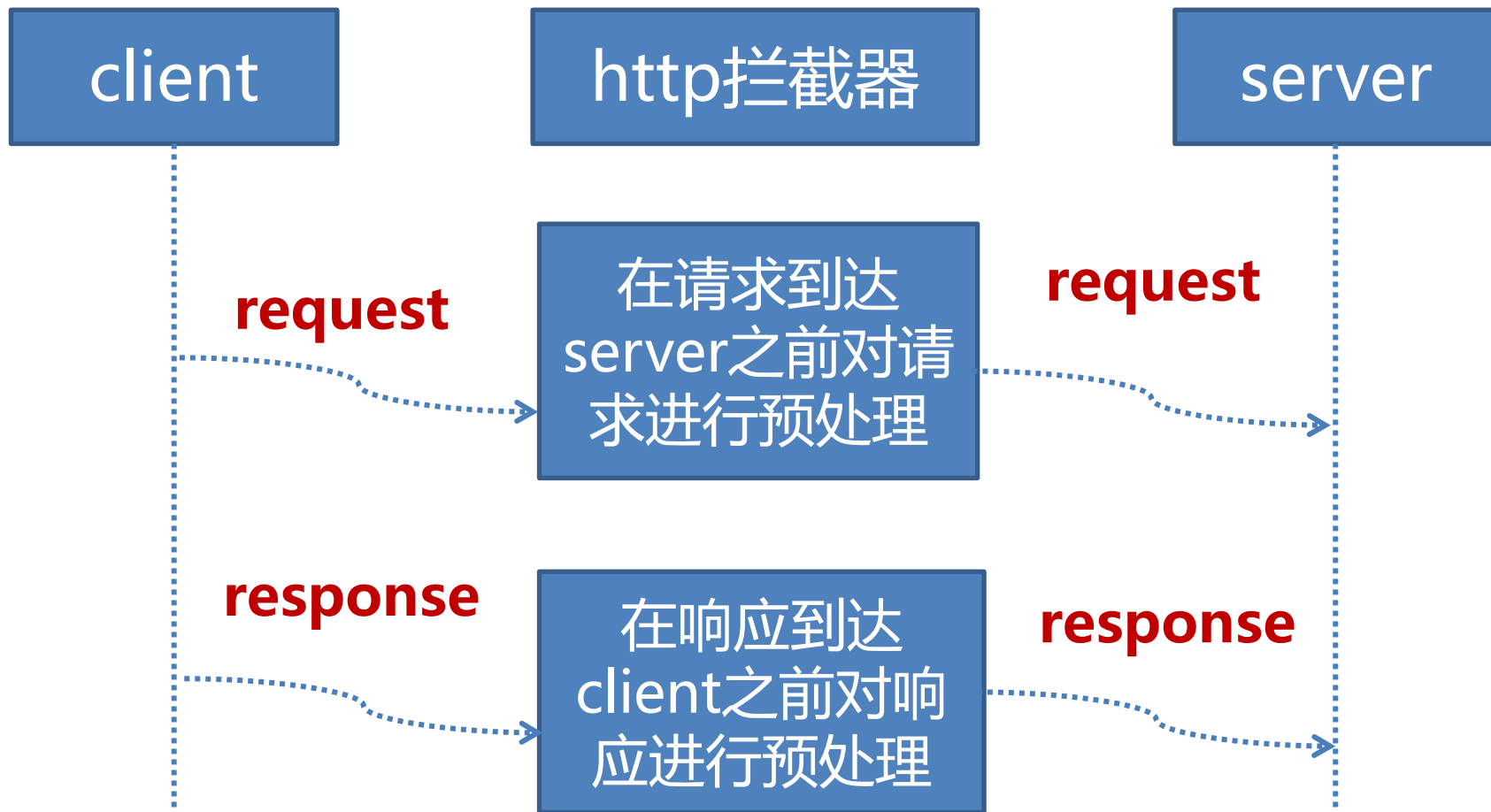
拦截器属性

名称	描述
request	在产生请求并传入配置对象前调用的拦截器函数
requestError	当上一个请求拦截器抛出错误时调用的拦截器函数
response	在响应被接收并传入响应对象时调用的拦截器函数
responseError	当上一个响应拦截器抛出错误时调用的拦截器函数

6

使用Ajax拦截器

拦截器示意图



6

使用Ajax拦截器

```
.config(function ($httpProvider) {  
    $httpProvider.interceptors.push(function () {  
        return {  
            request: function (config) {  
                config.url = "productData.json";  
                return config;  
            },  
            response: function (response) {  
                console.log("数据的个数: " + response.data.length)  
                return response;  
            }  
        }  
    });  
});
```

拦截请求并重新配置 ←

拦截响应，在之前输出信息 ←

Demo:listing10.html

2

使用promise服务

①

了解promise

什么是promise?

\$http服务常用于产生和处理Ajax请求，它被异步执行的标准HTTP请求, **\$http服务**的**get()**方法可以返回一个promise对象。

promise是对象，代表了一个函数最终可能的返回值或者抛出的异常。

Promise是一种异步方式处理值（或者非值）的方法。

①

了解promise

为什么使用promise？

让人头痛的异步回调，比如：

```
funA (arg1, arg2, function () {  
    funcB (arg1, arg2, function () {  
        funcC (arg1, arg2, function () {  
            xxxx....  
        })  
    })  
})
```

②

promise对象的方法

promise对象—— \$http服务的get()方法返回的对象。

方法	描述
then(success,error,notity)	注册被调用的函数以响应deferred对象的resolve,reject,notify方法
catch(error)	注册仅用于错误的处理函数，其所传参数用于调用deferred对象的reject方法
fomally(fn)	注册无论promise被解决还是拒绝都会调用的函数。其所传参数用于调用deferred对象的resolve或reject方法

②

promise对象的方法

promise对象—— \$http服务的get()方法返回的对象。

```
promise.then(function(value){  
    //success  
},function(value){  
    //error  
},function(value){  
    //notify  
})  
  
.catch(function(e){  
    //error  
})  
  
.finally(function(value){  
    //结束状态既被调用  
});
```

Demo:listing11.html

②

Promise服务

\$q 服务是AngularJS中自己封装的Promise

\$q常用的几个方法：

- ◆ **defer()** 创建一个deferred对象。
- ◆ **all()** 传入Promise的数组，批量执行，返回一个promise对象
- ◆ **when()** 传入一个不确定的参数，如果符合Promise标准，就返回一个promise对象。

Demo:listing12.html

②

Promise服务

在Promise中，定义了三种状态：

等待状态、完成状态、拒绝状态

关于状态：

- 1、状态的变更是不可逆的
- 2、等待状态可以变成完成或者拒绝

Demo:listing12.html

③

defer()方法

\$q.defer()方法创建一个deferred对象

deferred对象是promise接口的实现。它是非同步操作的通用接口，可以看作是一个等待完成的任务。

deferred对象属性和方法

名称	描述
resolve(result)	带有指定值的延迟活动 完成 的信号（ 变为完成状态 ）
reject(reason)	延迟活动失败或由于特定原因将 不被完成 的信号（ 变为拒绝状态 ）
notify(result)	提供来自延迟活动的 临时结果 （ 更新 ）
promise	返回接收其他方法信息号的 promise对象

4

all()方法

- all()方法可以把多个promise的数组组合成一个。
- 当所有的promise执行成功后，会执行后面的回调。
- 回调中的参数，是每个promise执行的结果。

```
$q.all([funcA(), funcB()])  
    .then(function(result){  
        console.log(result);  
    });
```

Demo:listing13.html

5

when()方法

when方法中可以传入一个参数，这个参数可以是一个值或符合promise标准的外部对象。

```
$q.when(funcA())  
    .then(function(result){  
        console.log(result);  
    });
```

Demo:listing14.html

动手试一试：应用指令、模块依赖、promise服务

开始

结束

中止

结果: 开始

开始

结束

中止

结果: 结束

开始

结束

中止

结果: 失败(中止)

Demo:listing15.html

本课小结

1. 使用Ajax服务

`$http` 是一个用于读取web服务器上数据的服务。

`$http.get(url)` 是用于读取服务器数据的函数。

2. 使用promises服务

`$q`服务是AngularJS中自己封装的Promise

- ◆ `defer()`
- ◆ `all()`
- ◆ `when()`

TNAKS

主讲：王智娟

QQ: 24132228

Email: wangzhijuan@onest.net