[Ext.data.proxy.Proxy 代理类的根类 1](#_Toc11490)

[客户端代理 1](#_Toc13258)

[服务端代理模式有 6](#_Toc7800)

[常用的代理模式Ajax 6](#_Toc8686)

[JsonP 9](#_Toc18422)

[Direct 代码使用Ext.Direct 向服务器发动数据 9](#_Toc5073)

## Ext.data.proxy.Proxy 代理类的根类

客户端代理：

**1.**LocalStorageProxy**：将数据存储在localStorage中，****此种方式可以持久的 将数据存储在客户端**

要使用代理，我们首先要有一个数据模型类，我们定义一个简单的Person类：

Ext.define('Person', {extend: 'Ext.data.Model',
fields: ['name', 'age']});

有了Model，我们还需要一个Store。我们可以把 Store 理解为数据仓库， 它对应数据库中的表，而Store 包含的 Record 则是数据表中的每一行记 录。我们同样简单的创建一个Store对象：

var personStore = Ext.create("Ext.data.Store", {
model: 'Person'});

接下来就到我们代理出场的时候了。我们需要创建一个LocalStorageProxy：

var personProxy = new Ext.data.proxy.LocalStorage({
id: 'Person\_LocalStorage', model: 'Person'});

将代理和Store 联系在一起

personStore.setProxy(personProxy);

际应用中，可以在Model 或 Store 中直接使用proxy 选项，

我们在后面的示例中将会看到具体的用法。

personStore.add({ name: 'www.qeefee.com', age: 1 });
personStore.add({ name: 'qf', age: 26 });
personStore.add({ name: 'qifei', age: 26 });
personStore.sync();

要查看保存的数据，我们需要先将数据从LocalStorage中加载到Store 中，

然后对Store 进行遍历：

personStore.load();

var msg = [];

personStore.each(function (person) {
msg.push(person.get('name') + ' ' + person.get('age'));

});

Ext.MessageBox.alert('提示', msg.join('<br />'));

使用load方法会将所有的数据都加载到Store中，如果我们需要进

行查询指定的数据，就要用到filter方法了

personStore.filter("name", /\.com$/);

使用多个过滤条件：

personStore.filter([

{ property: "name", value: /\.com$/ },

{ filterFn: function (item) { return item.get("age") > 10; } }
]);

使用过滤条件代替过滤方法

personStore.filter(function (item) {

return item.get("age") > 10 && item.get('name').length > 3;});

更新操作：

//得到第一个person对象

var person = personStore.first();

//修改名字字段

person.set('name', 'qeefee.com');

//保存到本地

personStore.sync();

删除操作：

//得到第一个person对象

var person = personStore.first();

//移除

personStore.remove(person);

//保存

personStore.sync();

**2.SessionStorageProxy：将数据存储在sessionStorage中，**

**此种方式只在当前会话中生效，当关闭浏览器以后，数据也会随之丢失。**

用法与LocalStorageProxy 的用法基本一致，我们这次在Model中添加 proxy 的配置项

//定义数据模型

Ext.define('Person', {extend: 'Ext.data.Model',
fields: ['name', 'age'], proxy: {type: 'sessionstorage',

id: 'myProxyKey'
}});

在Model中加入代理配置以后，我们就不需要单独的定义代理对象了，

可以直接使用Store来对其进行操作

//定义Store

var personStore = Ext.create("Ext.data.Store", {
model: 'Person'});

//添加数据

personStore.add({name:'www.baidu.com',age:1});
personStore.add({name:'qf',age:26 });

personStore.add({name:'qifei',age: 26 });

personStore.sync();

//读取数据

personStore.load();

Var msg=[];

personStore.each(function(person){
msg.push(person.get('name') + ' ' + person.get('age'));

});

Ext.MessageBox.alert('提示', msg.join('<br />'));

**3.MemoryProxy：将数据存储在内存中，此种方式只在当前页面有效，**

**且如果刷新页面，数据将丢失。**

//定义数据模型

Ext.define('Person', { extend: 'Ext.data.Model',

fields: ['name', 'age']});

var data = {Users: [{ name: 'www.baidu.com', age: 1 },

{ name: 'qeefee', age: 1 }]}

//定义Store

var personStore = Ext.create("Ext.data.Store", {

model: 'Person',

data: data.Users, proxy: {type: 'memory'}});

在代码中高亮显示的部分，就是为store设置的缓存。当personStore 定义的时候，它就会自动的将数据加载到Store中，所以在我们访问数据的时候不需要再调用load方法了：

//读取数据

Var msg=[];

personStore.each(function(person) {

msg.push(person.get('name')+'

'+person.get('age'));});

Ext.MessageBox.alert('提示', msg.join('<br />'));

和上两个的操作基本一致，我们可以通过下面的代码添加一行新数据：

personStore.add({ name: 'qifei', age: 26 });

personStore.sync();

服务端代理模式有**：**

Ajax ： 在当前域中发送请求。

JsonP: 跨域的请求。

Rest : 与服务器进行RESTful（GET/PUT/POST/DELETE）交互

Direct: 使用 Ext.direct.Manager 发送请求。

常用的代理模式Ajax

示例：

Ext.onReady(function(){

Ext.define('Person',{extend: 'Ext.data.Model',

fields: ['name', 'age']

});

var store= Ext.create('Ext.data.Store',

{model:'Person',

proxy: {

type: 'ajax',

url: rootUrl +'sample/getjson',

reader:{ type: 'json',
root: 'users'}}

});

store.load();

var msg = [];

store.each(function(person){

msg.push(person.get('name')+''+person.get('age'));

});

Ext.MessageBox.alert('提示', msg.join('<br />'));
});

在这段代码中，首先定义了Model类Person，然后创建了Person类的数据仓库 store，store 使用了ajax代理，通过url向服务器请求数据，ajax代理的reader 配置项是告诉程序以何种方式读取请求到的数据。

服务器返回的JSON 字符串如下：

{users:[{name:'www.baidu.com', age:1}, {name:'Tom', age:26}]}

如果你打开网络跟踪，会发现程序已经将成功得到了这段JSON字符串。那是为什么仍然是空呢？

原因是当我们调用load()方法的时候，我们告诉store去请求数据吧，然后store就使用ajax的方式请求url，注意，ajax 是异步的，所以当我们调用load()方法以后，马上执行输出的时候，store仍然是空的，可能它还在等待服务器响应数据。

所以，在我们使用ajax方式加载数据的时候，就需要为load() 方法添加一个callback 方法，当数据加载完成以后，再callback方法中进行数据输出，修改load方法的调用：

store.load({

callback:function(records,operation,success){

if(success){varmsg=[];store.each(

function(person){

msg.push(person.get('name') + ' ' + person.get('age'));

});

Ext.MessageBox.alert('提示', msg.join('<br />'));}}

});

在使用ajax进行请求的时候，ajax代理会调用自身的read方法，该方法的第一个参数是Ext.data.Operation 类型，它用来配置我们如何进行请求。我们在load 方法中可以通过配置项传递一些参数，load在调用read方法时将这些参数传递过去，read则会根据这些参数生成Ext.data.Operation 的一个实例。所以，我们可以这样来传递参数

store.load({

page: 2,

limit:10,params:{name:'QF'},

callback:function(records,operation,success){

if(success){varmsg=[];store.each(function(person) {msg.push(person.get('name') + ' ' + person.get('age'));

});

Ext.MessageBox.alert('提示', msg.join('<br />')); }}});

/SampleExtJS/sample/getjson?\_dc=1374141754304&name=QF&pag e=2&start=25&limit=10

通过传递这些参数，我们可以完成分页、查询等操作。

### JsonP

在ExtJS 中我们可以轻松的使用JsonP代理进行跨域的数据请求：

//创建Store

var store = Ext.create('Ext.data.Store', {model: 'Person',
proxy: {

type: 'jsonp',

url: 'http://www.abc.com/sampleextjs/sample/getjsonp',

reader: {type: 'json',
root:'users'}}

});

务器返回的字符串为：

Ext.data.JsonP.callback1({

users: [{ name: 'www.baidu.com', age: 1 },

{ name: 'QF', age: 26 }]

})

Direct 代码使用Ext.Direct 向服务器发动数据**。**

Ext.Direct 技术的好处是，允许你像调用Javascript 方法一样调用服务器端代码

Ext.define('User', {

extend: 'Ext.data.Model',

fields: ['firstName', 'lastName'],

proxy: { type: 'direct', directFn: MyApp.getUsers, paramOrder: 'id' }

});

User.load(1);