DataProvider

> 正常业务中, 必然需要有数据获取和数据提交的操作. 本节介绍了一种通用的数据获取方式 —— DataProvider

DataProvider是`container`组件的一个扩展功能.

使用DataProvider, 可以`container`组件从外部渠道获取数据.

例子: 通过HTTP接口获取数据

假设我们目前的JSON配置是这样的.

```
{
    "type": "container",
    "model": "demo",
    "data": {
        "name": "andycall",
        "age": "#ES{$data.name} + ' and andylaw'"
    },
    "children": [
        {
            "type": "text",
            "text": "#ES{$data.name}"
        },
        {
            "type": "text",
            "text": "#ES{$data.age}"
        }
    ]
}
```

我想再通过接口获取一些值, 并通过Text组件展示出来.

接口: http://cp01-rdqa-dev420-dongtiancheng.epc.baidu.com:8899/

STEP 1. 配置AjaxDataProvider

DataProvider支持获取各种来源的数据, 在这里例子里, 我们是需要通过HTTP来获取数据. 所以我们就需要`AjaxDataProvider`这个`dataProvider`扩展来做这个事情.

首先, 我们需要在JSON中使用`dataProvider`这个字段来声明这个`container`组件需要哪些扩展组件.

```
{
  "type": "container",
  // ...
  "dataProvider": [],
  // ...
}
```

`dataProvider`类型是一个数组, 数组中每一个元素代表一个`dataProvider`扩展的配置项.

DataProvider配置项

每一个`dataProvider`扩展的配置项都只会包含2个字段:

- mode: 扩展的类型
- config: 扩展的配置

现在这个例子中, 我们要使用`ajaxDataProvider`, 所以mode类型设置成`ajax`.

`config`传入一个对象, 来表述这个请求的扩展配置.

`config`的参数配置一栏如下:

参数名	类型	备注
url	string	请求的地址
method	string	请求的方法
data	Object	请求附带的数据, 参数值可以通过Expression String动态赋值
retCheckPattern	Expression String	使用Expression String验证数据返回是否合法
retMapping	Object	使用一个普通对象来对返回值进行数据映射

```
{
    "mode": "ajax",
    "config": {
        "url": "http://cp01-rdqa-dev420-dongtiancheng.epc.baidu.com:8899/",
        "method": "GET",
        "data": {
            "name": "#ES{$data.name}"
        }
    }
}
```

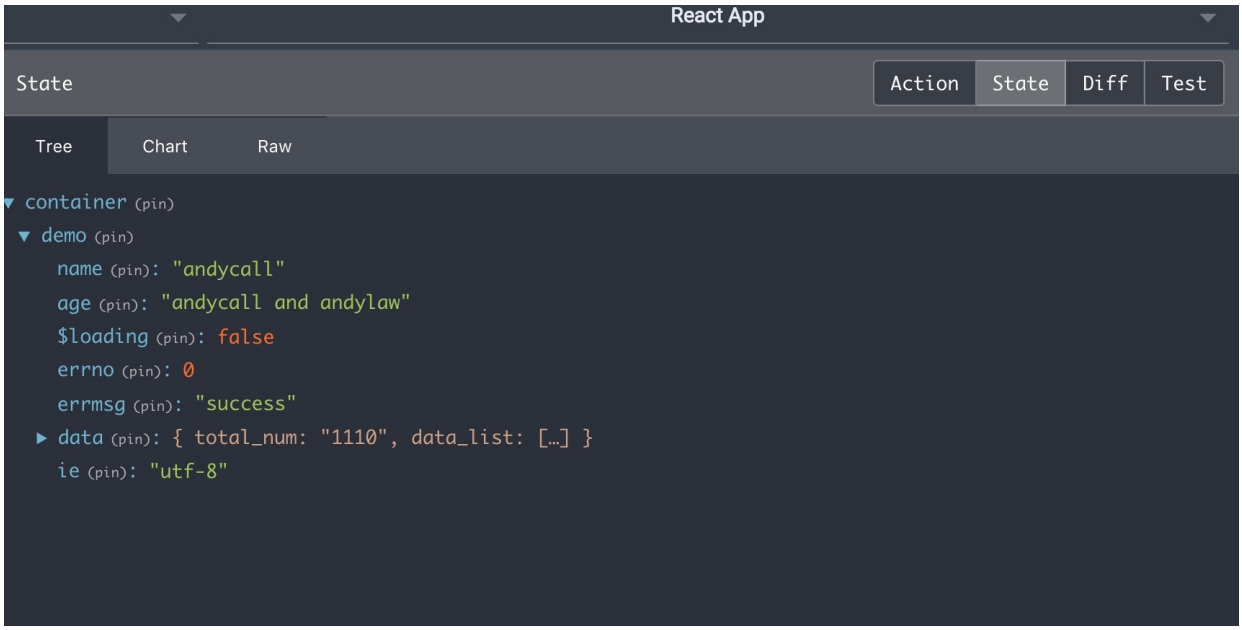
Expression String一样可以应用在请求发送之前, 这样就可以在发送请求的时候, 动态把数据模型中的值作为请求参数发送到后端server.

目前我们的JSON配置大概是这个样子:

```
{
    "type": "container",
    "model": "demo",
    "data": {
      "name": "andycall"
    },
    "dataProvider": [
        {
            "mode": "ajax",
            "config": {
                "url": "http://cp01-rdqa-dev420-dongtiancheng.epc.baidu.com:8899/",
                "method": "GET",
                "data": {
                    "name": "#ES{$data.name}"
                }
            }
        }
    ],
    "children": [
        {
            "type": "text",
            "text": "loading errno: #ES{$data.errno}"
        },
        {
            "type": "text",
            "text": "loading status: #ES{$data.errmsg}"
        }
    ]
}
```

每一次刷新页面, 浏览器都会向这个接口发送一个GET请求, 并带有一个参数: name: andycall.

运行之后, 就能发现浏览器已经成功发送了请求, 接口返回的值的所有字段也都写入到了`demo`这个数据模型之中.



STEP 2.把接口获取的值传递到字级组件

现在我们的`demo`数据模型中有这么多的数据了, 接下来就是要将数据模型中的数据渲染到字级组件上了.

通过上一章介绍的技巧, `container`组件会给所有的子组件提供`$data`对象. 而`$data`正是`container`组件的数据模型. 所以我们就可以使用这个`$data`对象和Expression String来动态改变字级组件的属性值.

假如我们想实现这样的功能:

errno: 0 status: "success"

做这个之前, 需要阐述一下DataProvider扩展类型.

DataProvider扩展类型

在RCRE中, 一共有2种DataProvider扩展类型 —— 同步和异步.

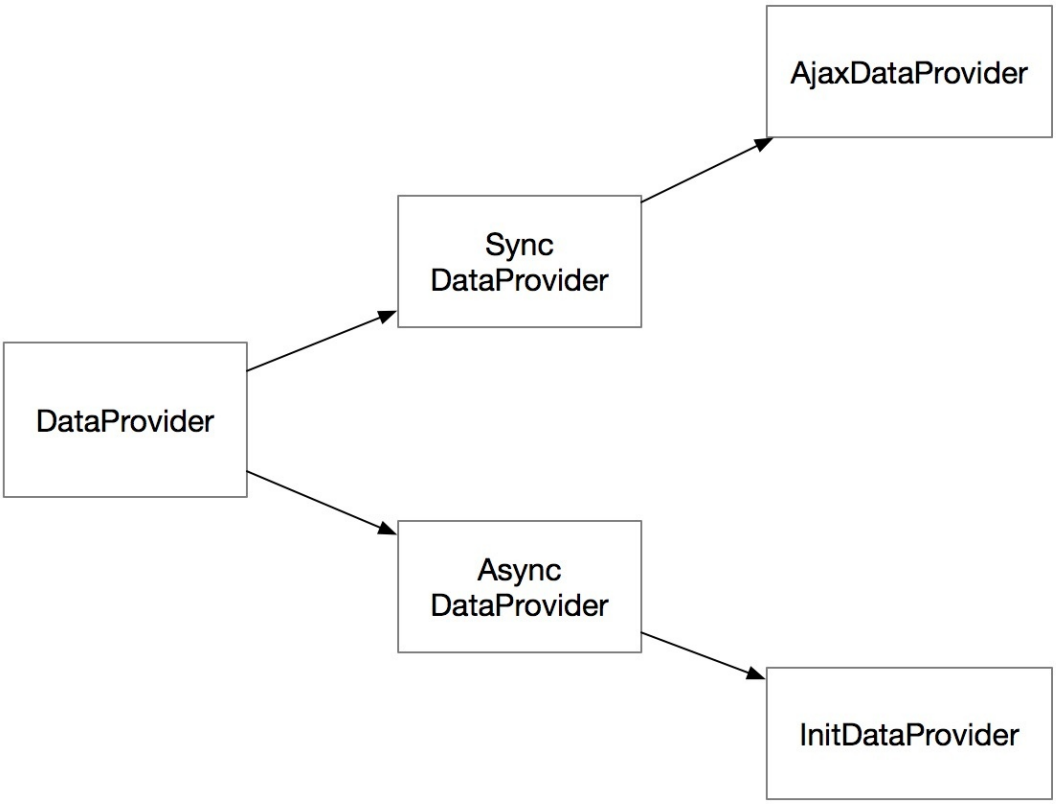
目前RCRE只含有2个DataProvider扩展;

异步

- AjaxDataProvider

同步

- InitDataProvider



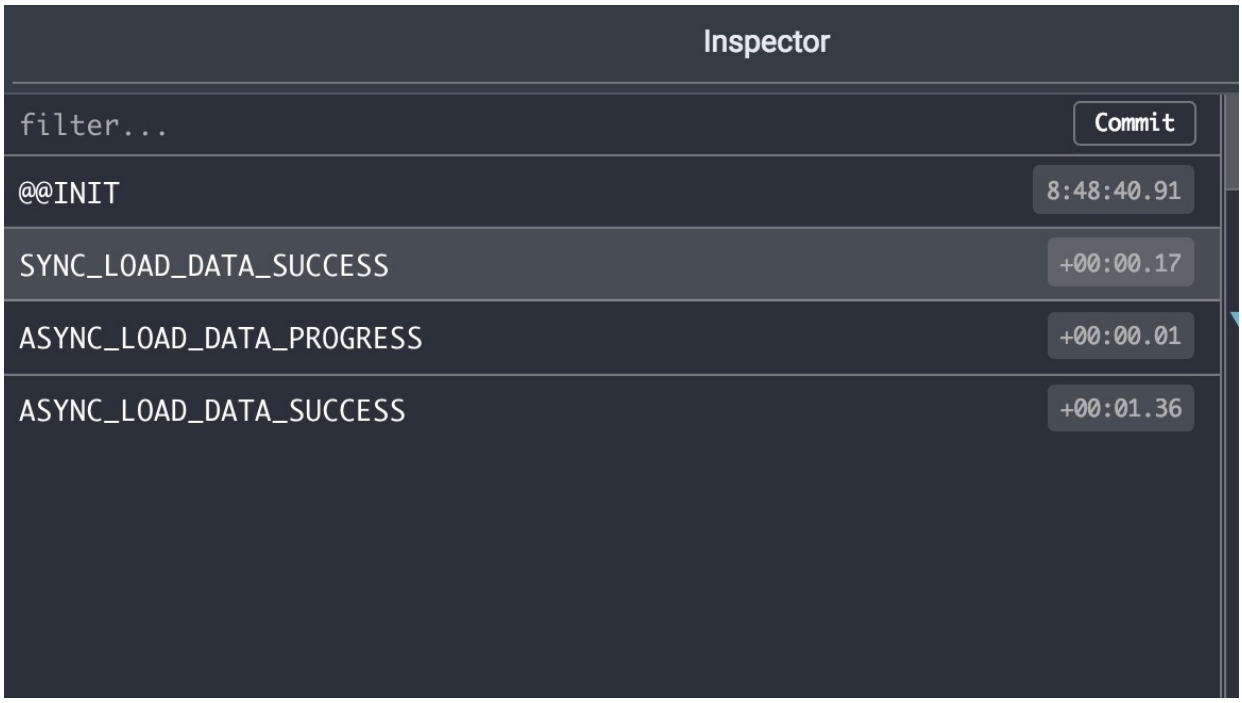
同步扩展的运行阶段

扩展的运行阶段是通过Redux Action来进行记录. 我们也可以通过下图的Redux DevTools来查看运行状态.

所有DataProvider 相关Action一览表

Action 名称	扩展类型	备注
SYNC_LOAD_DATA_SUCCESS	同步	同步写入成功
SYNC_LOAD_DATA_FAIL	同步	同步写入失败
ASYNC_LOAD_DATA_PROGRESS	异步	异步写入中
ASYNC_LOAD_DATA_SUCCESS	异步	异步写入
ASYNC_LOAD_DATA_FAIL	异步	异步写入失败

`Container`组件初始化的时候, 会自动触发一个SYNC_LOAD_DATA_SUCCESS来初始化`data`属性中的数据.



所有的异步DataProvider扩展, 都会默认写入一个`$loading`这个默认属性, 我们可以通过这个属性来捕捉当前组件的运行状态.

捕捉思路也非常简单:

- $loading == true // 正在加载, ASYNC_LOAD_DATA_PROGRESS
- $loading == false // 加载成功或者失败, ASYNC_LOAD_DATA_FAIL || ASYNC_LOAD_DATA_SUCCESS

下面我们就可以实现上图gif的效果啦.

```
{
    "body": [
        {
            "type": "container",
            "model": "demo",
            "data": {
                "name": "andycall"
            },
            "dataProvider": [
                {
                    "mode": "ajax",
                    "config": {
                        "url": "http://cp01-rdqa-dev420-dongtiancheng.epc.baidu.com:8899/",
                        "method": "GET",
                        "data": {
                            "name": "#ES{$data.name}"
                        }
                    }
                }
            ],
            "children": [
                {
                    "type": "text",
                    "hidden": "#ES{$data.$loading === false}",
                    "text": "loading..."
                },
                {
                    "type": "text",
                    "hidden": "#ES{$data.$loading === true}",
                    "text": "errno: #ES{$data.errno}"
                },
                {
                    "type": "text",
                    "hidden": "#ES{$data.$loading === true}",
                    "text": "status: #ES{$data.errmsg}"
                }
            ]
        }
    ]
}
```

在RCRE中, 每一个组件都有一个`hidden`属性, 如何`hidden`为false, 这个组件就会隐藏. 所以我们可以通过Expression String来计算出hidden的值, 从而能动态控制Text组件的显示和隐藏.

命名空间

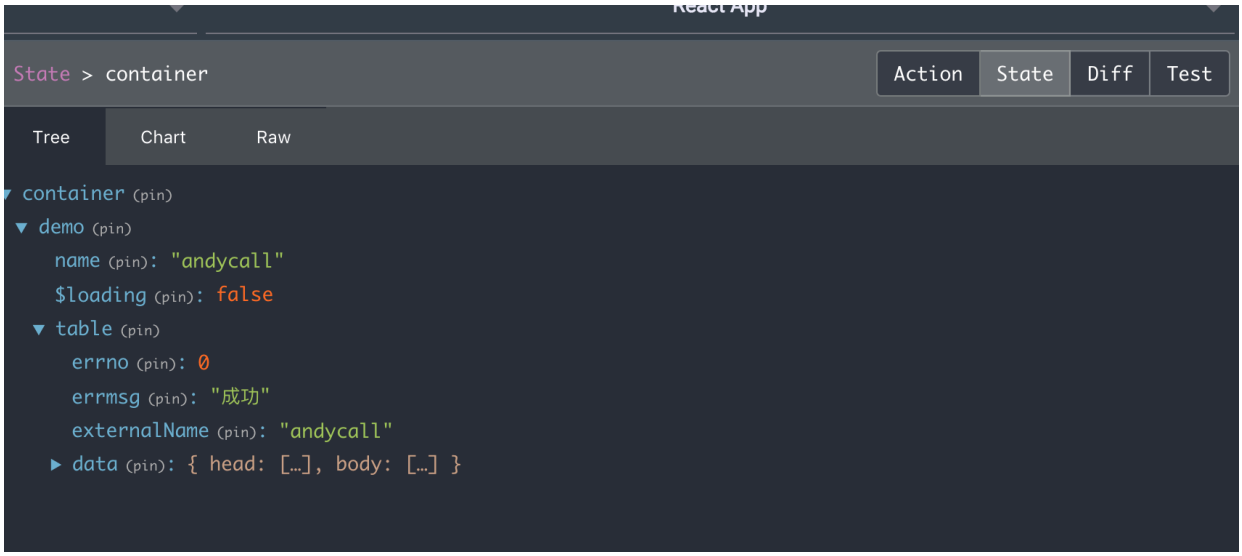
在复杂的业务场景中, 一个`container`组件会使用多个接口, 多个接口之间的字段非常有可能会出现冲突.因此RCRE引入了命名空间这个功能, 来让每一个接口都含有一个独立的字段名称.

使用命名空间, 只需要在一个`dataProvider`的配置中添加`namespace`这个属性.

例如:

```
{
    "mode": "ajax",
    "namespace": "table",
    "config": {
        "url": "http://cp01-ebg-nativeads-50.cp01.baidu.com:8088/realtime/channel/table",
        "method": "GET",
        "data": {
            "date": "20171026",
            "fields": "pv",
            "product": "8",
            "placeId": "1504243449802"
        }
    }
}
```

配置了`namespace`之后, 我们再看下Redux的state中, 就可以发现



接口中所有的数据都会保存在`demo.table`下面.

不过需要注意一点的是, 在container组件的其他地方, 对接口的数据进行获取的时候, 就需要使用

`#ES{$data.table}`来进行获取.

例如在这个例子中, 获取`errmsg`的Expression String是这样的`#ES{$data.table.errmsg}`

返回值校验

外部的数据往往是不可信的, 不稳定可靠的. 我们需要对DataProvider扩展返回的数据进行数据校验. 比如验证ajax接口返回的errno是否等于0.

返回值校验需要使用DataProvider提供的`retCheckPattern`属性.

`retChechPattern`是一个Expression String, 这个Expression String中会嵌入一个`$output`变量.

这个`$output`实际上就代表着数据的原始返回值.

这个Expression String的返回值可以依靠JavaScript隐性转换的逻辑.

也就是说, 返回值只要不是

- 0
- false
- null
- undefined
- ''

之外, 其他的值都会被当作是true

在上面的例子中, 添加一个接口的验证只需要添加这个就可以了.

```
"dataProvider": [
    {
        "mode": "ajax",
        "config": {
            "url": "http://cp01-rdqa-dev420-dongtiancheng.epc.baidu.com:8899/",
            "method": "GET",
            "data": {
                "name": "#ES{$data.name}"
            }
        },
        "retCheckPattern": "#ES{$output.errno === 0}"
    }
]
```

返回值映射

DataProvider获取的值是没必要都写入到数据模型中的, 数据字段多过可能还会导致数据被覆盖这样的情况.

我们可以手动告诉RCRE, 可以将接口的哪些字段写入到数据模型中. 我们就可以使用`retMapping`这个属性.

`retMapping`是一个普通javaScript对象. 对象的每个属性值都是一个Expression String, 这个Expression String会嵌入以下变量.

- $data 当前container组件的数据模型
- $output DataProvider返回值

例如:

```
"dataProvider": [
    {
        "mode": "ajax",
        "config": {
            "url": "http://cp01-rdqa-dev420-dongtiancheng.epc.baidu.com:8899/",
            "method": "GET",
            "data": {
                "name": "#ES{$data.name}"
            }
        },
        "retCheckPattern": "#ES{$output.errno === 0}",
        "retMapping": {
            "errno": "#ES{$output.errno}",
            "errmsg": "#ES{$output.errmsg}",
            "externalName": "#ES{$data.name}",
            "data": "#ES{$output.data}"
        }
    }
]
```

这个例子中, RCRE会解析Expression String, 并写入数据`errno`, `errmsg'`, `externalName`和`data`到数据模型中.