

Mapping Object

3 - 加载解析的代码到前端Ruby-JS

```
$data = jsrb_undata($document.at_css('#data-transfer').text)
```

4 - 前端加载页面，运行编译后JS代码

用dom里面查找id=“data-transfer”的元素，获取text
然后text进入jsrb_undata函数，来初始化全局变量\$data

后端

前端

DB



```
Class Foo < MappingObject
  mapping_accessor :value

  def calc_value()
    return value**2
  end
end

foo=Foo.new()
foo.value = load_data_from_db()
RenderWrap["foo"] = foo
RenderWrap.data
```

HTML



```
<pre id='data-transfer'
style='display:none'>
  xxxxxxxxxxxxxxxxxxxxxx
</pre>

<script>
  $data = jsrb_undata($document.at_css('#data-
transfer').text)
</script>
```



```
$data["foo"].calc_value
```

Mapping Object

5 - jsrb_undata(data) 函数

{key: "(MappingObject)|||Foo|||{value:123}"}
如果字符串有(MappingObject), 就切出 类名 和 类的序列化值

```
_, class_name, obj_data = str.split("|||")  
class_name = "Foo"  
obj_data = "{value:123}"
```

5 - jsrb_undata(data) 函数

```
obj = (Object.const_get class_name).new  
# 等价于 obj = Foo.new, 动态语言特性
```

```
obj.from_data(obj_data)
```

后端

前端

DB



```
Class Foo < MappingObject  
  mapping_accessor :value  
  
  def calc_value()  
    return value**2  
  end  
end  
  
foo=Foo.new()  
foo.value = load_data_from_db()  
RenderWrap["foo"] = foo  
RenderWrap.data
```

HTML



```
<pre id='data-transfer'  
  style='display:none'>  
  xxxxxxxxxxxxxxxxxxxxxx  
</pre>  
  
<script>  
  $data = jsrb_undata($document.at_css('#data-  
transfer').text)  
</script>
```



```
$data["foo"].calc_value
```