Blueprints

Montage blueprints是一种为应用对象添加metadata信息的机制。blueprints已经支持组件和控制器对象。它是数据层的一个重要角色（还没用实现）。blueprint为对象添加的信息包括，对象的属性以及与其它对象的关系。

Blueprints就是一组描述信息。Montage为每个组件和控制器创建唯一的blueprint标识，这个标识会在反序列化的时候用到。Blueprints信息一般是从一个JSON文件反序列化得到，当然也可以用代码动态生成。

组件和控制器对象的Blueprints

可以通过组件或者控制器对象的 `blueprint` 属性访问blueprint。它返回一个关于blueprint反序列化的promise。

对象的blueprint包含一组关于对象属性的描述。blueprints数据也可以通过一个函数进行逻辑分组。blueprint同时也提供验证规则和必要绑定。验证规则对组件和控制器对象进行边界检查。

Blueprints属性

一个blueprint属性描述对象的一个属性。除了定义blueprint属性名之外还包括基数，数据类型，合法值。基数定义属性可以使用的值。

Blueprints关系

association属性定义两个对象之间的关系。它定义blueprint关系的目标对象。

Validation Rules Objects

TDB

Creating Blueprints

创建Blueprints

Although most developers will only interact with pre-existing blueprints deserialized from a file, it is quite easy to create a blueprint in memory: 大多数时候开发者只需要从一个文件反序列化得到Blueprints，我们也可以非常容易地在内存中创建一个blueprint：

```
var companyBinder = BlueprintBinder.create()
    .initWithName("CompanyBinder");

var personBlueprint = companyBinder
    .addBlueprintNamed("Person", "meta/blueprint/person");
personBlueprint.addToOnePropertyBlueprintNamed("name");
personBlueprint.addToManyPropertyBlueprintNamed("phoneNumbers");

var companyBlueprint = companyBinder
    .addBlueprintNamed("Company", "meta/blueprint/company");
companyBlueprint.addToOnePropertyBlueprintNamed("name");
companyBlueprint.addToManyAssociationBlueprintNamed(
    "employees",
    personBlueprint.addToOneAssociationBlueprintNamed("employer")
);

var projectBlueprint = companyBinder
    .addBlueprintNamed("Project", "meta/blueprint/project");
projectBlueprint.addToOnePropertyBlueprintNamed("name");
projectBlueprint.addToOnePropertyBlueprintNamed("startDate");
projectBlueprint.addToOnePropertyBlueprintNamed("endDate");

companyBlueprint.addToManyAssociationBlueprintNamed(
    "projects"
```

```
    "projects",
    personBlueprint.addToOneAssociationBlueprintNamed("company")
);

personBlueprint.addToManyAssociationBlueprintNamed(
    "projects",
    projectBlueprint.addToManyAssociationBlueprintNamed("contributo
rs")
);

BlueprintBinder.manager.addBlueprintBinder(companyBinder);
```

组件更简单一些。比如下面是一个定义按钮组件blueprint的例子。

```
var serializer = Serializer.create().initWithRequire(require);

//Create a new empty blueprint with the button identifier as a name
.
var newBlueprint = Blueprint.create().initWithName(button.identifie
r);

// Then creat all the property description we need
var autofocus = newBlueprint.addToOnePropertyBlueprintNamed("autofo
cus");
autofocus.valueType = "string";
autofocus.helpKey = "Specifies that a button should automatically g
et focus when the page loads";

var enabled = newBlueprint.addToOnePropertyBlueprintNamed("enabled"
);
enabled.valueType = "boolean";
enabled.helpKey = "Specifies that a button should be enabled";

var form = newBlueprint.addToOnePropertyBlueprintNamed("form");
form.valueType = "string";
form.helpKey = "Specifies one or more forms the button belongs to";
```

```
var formaction = newBlueprint.addToOnePropertyBlueprintNamed("forma
ction");
formaction.valueType = "url";
formaction.helpKey = "Specifies where to send the form-data when a
form is submitted. Only for type='submit'";

var formenctype = newBlueprint.addToOnePropertyBlueprintNamed("form
enctype");
formenctype.valueType = "enum";
formenctype.enumValues = ["application/x-www-form-urlencoded", "mul
tipart/form-data", "text/plain"];
formenctype.helpKey = "Specifies how form-data should be encoded be
fore sending it to a server. Only for type='submit'";

var formmethod = newBlueprint.addToOnePropertyBlueprintNamed("formm
ethod");
formmethod.valueType = "enum";
formmethod.enumValues = ["get", "post"];
formmethod.helpKey = "Specifies how to send the form-data (which HT
TP method to use). Only for type='submit'";

var formnovalidate = newBlueprint.addToOnePropertyBlueprintNamed("f
ormnovalidate");
formnovalidate.valueType = "boolean";
formnovalidate.helpKey = "Specifies that the form-data should not b
e validated on submission. Only for type='submit'";

var formtarget = newBlueprint.addToOnePropertyBlueprintNamed("formt
arget");
formtarget.valueType = "string";
formtarget.helpKey = "Specifies where to display the response after
 submitting the form. Only for type='submit'";

var name = newBlueprint.addToOnePropertyBlueprintNamed("name");
name.valueType = "string";
name.helpKey = "Specifies a name for the button";

var label = newBlueprint.addToOnePropertyBlueprintNamed("label");
```

```
label.valueType = "string";
label.helpKey = "";

var type = newBlueprint.addToOnePropertyBlueprintNamed("type");
type.valueType = "enum";
type.enumValues = ["button", "reset", "submit"];
type.helpKey = "Specifies the type of button";

var value = newBlueprint.addToOnePropertyBlueprintNamed("value");
value.valueType = "string";
value.helpKey = "Specifies an initial value for the button";

// And assign the property in groups following the logic for user p
resentation
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("label"), "base"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("type"), "base"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("name"), "base"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("enabled"), "base"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("autofocus"), "base"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("form"), "form"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("formaction"), "form"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("formenctype"), "form"
```

```
);
newBlueprint.addPropertyBlueprintToGroupNamed(
    newBlueprint.propertyBlueprintForName("formmethod"), "form"
);
newBlueprint.addPropertyBlueprintToGroupNamed(
```