## 1、概述

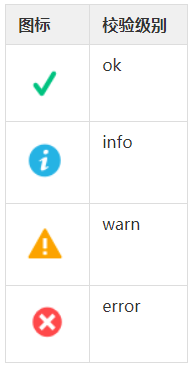
dorado7提供了完整的数据校验机制：

* 支持多种类型的数据校验，如：长度校验，中文字符校验，范围校验，正则表达式校验等等。
* 支持多种数据校验级别：ok、info、warn、error等，且不同校验状态拥有自己的个性化的校验图标。
* 校验器提供规范的属性配置，大多数场景无效编码。
* 提供校验器关闭特性，便于支持草稿保存或临时数据保存的业务场景。

下图是一个有各种数据校验的界面效果：



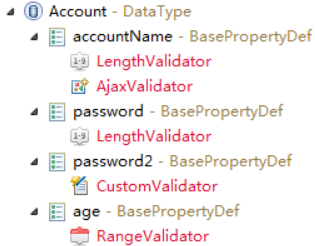
不同的图标代表不同的含义：



目前提供的校验器有8种：

|  |  |
| --- | --- |
| 校验器 | 说明 |
| RequiredValidator | 非空数据校验器 |
| LengthValidator | 长度数据校验器 |
| CharLengthValidator | 字节长度数据校验器 |
| RangeValidator | 范围数据校验器 |
| EnumValidator | 枚举数据校验器 |
| RegExtValidator | 正则表达式数据校验器 |
| AjaxValidator | Ajax数据校验器 |
| CustomValidator | 自定义数据校验器，当以上校验器都无法满足校验的需求时，可以自定义数据校验器 |

由于Dorado中客户端的MVC开发模式，因此数据校验器被设计为针对数据模型的，而不是页面上可见的控件。因此当希望对界面层的某一个控件添加数据校验逻辑的时候，就应该找到与之对应的数据模型定义（DataType下的PropertyDef），并在此对象上添加符合项目需求的数据校验器。



这样设计的好处是，当试图通过JS去修改数据对象的时候也会自动触发我们添加到数据模型对象上的数据校验器，从而提高数据的安全性。

## 2、手动触发数据校验的办法

数据校验都和数据模型有关，如果希望通过JS代码触发数据校验，可以先获得当前的数据对象，并通过Entity的validate()方法触发：

if(entity.validate() == “ok”) {

…

}

如果数据校验失败，返回的就是“invalid”。

## 3、validatorsDisabled特性（实现草稿保存）

场景：一个政府在线填报页面，200个编辑框，其中50个编辑框非空，通常而言我们都需要起码填写好这50个编辑框才能保存到数据库。能不能有一种机制，让我的业务数据即使非空字段没有填写情况下也可以临时保存一下，这样这个填报功能就可以多次填写，方便工作。利用校验器开关功能，我们可以较为方便的实现这种逻辑，在保存草稿的按钮代码中对当前要提交的Entity做如下控制：

...@TODO 获取当前Entity

entity.dataType.set("validatorsDisabled", true);

//禁用当前数据对象所有的数据校验

entity.reset();

//根据validatorsDisabled的值重置校验状态

...@TODO 保存当前的Entity对象

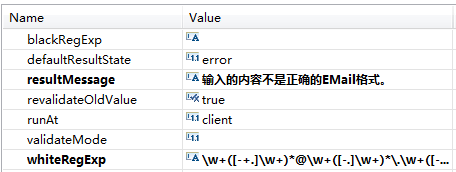
entity.dataType.set("validatorsDisabled", false);

//启用默认的数据校验

## 4、其他一些高级属性说明

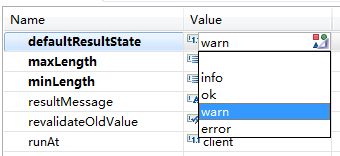
### resultMessage

validator数据校验失败会有系统默认的提示信息，定制信息可以通过resultMessage属性定制：



### defaultResultState属性

每一个Validator的默认的校验级别（defaultResultState）为error，可以通过打开defaultResultState属性对应的下拉框选择校验级别：



修改后，这个PropertyDef的数据校验如果成功就是ok，如果校验失败，则校验结果就是defaultResultState的值。

### acceptValidationState

关于validate()方法的计算规则：

对于一个Entity对象来说会有多个PropertyDef属性，并且每一个PropertyDef下可能有多个数据校验器，当我们执行Entity对象的validate()方法的时候会遍历内部所有的PropertyDef的所有数据校验器。那validate()方法如何决定最终校验的结果是ok还是invalid的呢?

为了方便这种计算，首先我们定义了一种规则，就是四种校验级别的顺序：

info <ok < warn < error

另外每一个数据校验器的校验结果的运算规则是：如果校验通过就返回ok，否则就是上面我们介绍到的defaultResultState属性值，默认是error。

如某一个属性定义了三个数据校验器：长度校验器，Enum校验器，正则表达式校验器。假设他们的校验结果为：

长度校验器成功，返回ok；

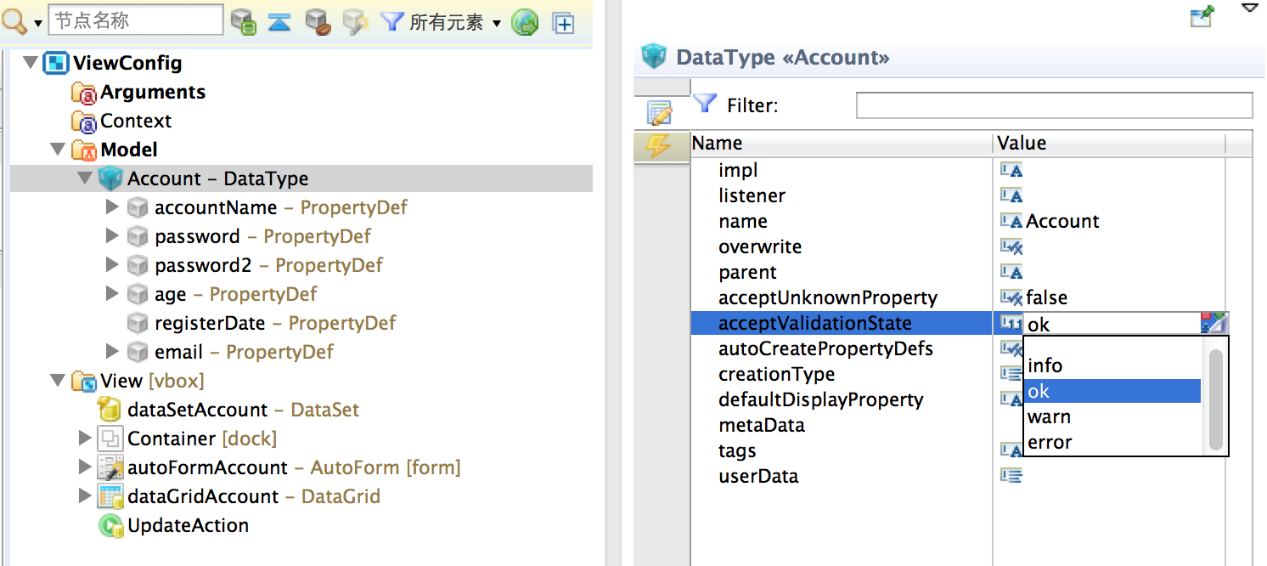
Enum校验器失败，返回info；

正则表达式校验器失败，返回warn

以上三个校验结果合并后，去级别最高的校验结果作为整个PropertyDef的数据校验结果，本例最终数据校验结果为warn。

之后遍历Entity下的每一个PropertyDef的校验结果。

最后再与DataType的acceptValidationState属性做比较：



将DataType的acceptValidationState的值设置为warn，而Entity的校验结果为warn，则最终Entity的validate()方法返回值就是ok。

关于validate()的触发：

通常情况下当我们在编辑框中输入数据的时候就会自动触发数据校验。

但当我们没有对一个数据对象的属性赋值时，是不会触发它的数据校验规则的。例如一个编辑框我们设置为长度必须大于5，小于10，我们没有对编辑框做过编辑工作，则默认不会触发这个编辑框所对应的PropertyDef的数据校验。这种情况下只有到最终数据提交的时候才需要通过Entity的validate()再次确认一下。

注：

大部分场景下我们并不会直接接触到validate()方法。因为大部分场景我们都是通过UpdateAction去提交数据的，这个控件内部会自动触发validate()进行数据校验。并且根据以上validate()的计算规则计算，如果计算结果为ok，就允许提交，否则就不允许提交。

在特殊情况下，如果你的数据不是通过UpdateAction的默认处理机制提交的，那么你就需要在提交数据之前通过JS调用Entity的validate()方法才会触发数据校验逻辑。你要根据运算结果再决定如何做下一步的逻辑处理，如：

if (entity.validate() == "ok"){

...

}

### revalidateOldValue原始值校验机制

当将某个编辑框的值从A改为B时，这个时候触发的数据校验没有通过的情况下。用户又将值从B修改为A。这个时候会再次触发数据校验。特殊情况下，如果不想再次触发数据校验，则可以将对应的数据校验器的revalidateOldValue的值设置为false，默认为true。