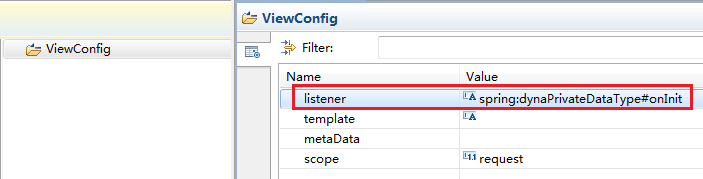
通常的开发模式中，会在View的Model中添加若干个DataType，并定义好内部所需要的PropertyDef。

但是在实际的使用过程中，有可能需要动态调整和完善DataType的设定，例如动态的修改PropertyDef的label等属性，甚至动态的添加和删除部分的PropertyDef，还有可能希望动态的添加整个DataType对象。这种场景尤其是在用户的系统是否存在数据字典的情况下，开发人员常常要面对如何通过数据字典等外部的信息来动态的创建DataType的需求。

例子：通过Java代码动态的创建和完善View中的私有DataType。

首先定义ViewConfig的Listener，并实现onInit方法。



onInit方法提供ViewConfig参数，通过ViewConfig可以方便的修改现有的DataType或创建新的DataType：

createDataType(String name)

createDataType(String name,String parent)

getDataType(String name)

创建或取到View中已有的DataType对象之后，再对DataType的属性操作就简单了：

EntityDataType dataTypeAddress =

(EntityDataType) viewCofig.createDataType("Address");

dataTypeAddress.setAutoCreatePropertyDefs(true);

dataTypeAddress.setDefaultDisplayProperty("city");

同样要进一步操作内部的PropertyDef也不麻烦：

EntityDataType dataTypeCondition =

(EntityDataType) viewCofig.getDataType("Condition");

PropertyDef propertyDef = dataTypeCondition.getPropertyDef("city");

propertyDef.setDefaultValue("上海");

propertyDef.setLabel("城市");

propertyDef.setRequired(true);

也可以创建新的PropertyDef:

PropertyDef propertyDef;

// Address

EntityDataType dataTypeAddress =

(EntityDataType) viewCofig.createDataType("Address");

propertyDef = new BasePropertyDef("id");

propertyDef.setDataType(viewCofig.getDataType("long"));

propertyDef.setLabel("编码");

propertyDef.setReadOnly(true);

dataTypeAddress.addPropertyDef(propertyDef);

propertyDef = new BasePropertyDef("city");

propertyDef.setLabel("城市");

dataTypeAddress.addPropertyDef(propertyDef);