面向数据字段的表现层组件设计 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/461/2021_2022__E9_9D_A2_E5_90_91_E6_95_B0_E6_c104_461633.htm 摘要：表现层的开发和维护一直是一件头痛的事情，尤其是Java这种手工编制的程序，本文主要讲述一种表现层架构的设计方式。 将UI的表现行为以及数据绑定行为封装成UI组件，使其有良好复用性的概念提出已久，比如TagLib，。Net的server component.对于胖客户端来说，也是一样。出于今后技术升级的考虑，RAB应该抽象UI组件接口，将具体组件接口的实现类的装载行为封装起来，通过配置文件声明式的装载。同时UI组件接口还应封装一些常用的表现行为及相关的属性设置，如国际化资源设定，文本框输入长度等。 作为UI组件来说，它本身一种数据资源（这里统指业务数据）的表现和入口。所以对于一个UI组件就是一个数据资源的表现，而这些表现行为属性就直接和数据资源对应，比如一个业务字段叫UserID，50个字符长，要求文本框显示，并且要大写，这些就是表现属性。这就使得一个UI的视图由会由许多数据字段决定产生。同时由于采用MVC的模式，View不会与包含任何的业务逻辑，所以可以将View上要显示的所有数据字段全部放入一个配置载体表示UI的结构。对于View的配置载体，xml提供了很好的形式，清晰的格式和层次，可以直观的反映UI的布局层次。 对于大多数数据库应用系统而言，可以为数据库中每个业务数据字段分配一套相关的UI表现行为的属性，将其放入配置文件或者数据库，每次系统启动之后加载这些属性，并作缓存。在页面装载的时候，可以用一个外部装载类解析xml进行UI

实例化，在图2中这个装载类就是GUIEngine.GUIEngine读取xml中的字段标签，从缓存中读取相应的字段的表现型为属性，根据这些属性生成UI组件，然后添加至业面。完成页面的装载。这称为用数据源模型直接与UI组件绑定的形式。RAD主要表现是所见即所得。Web表现技术在做到这点上很不直观，而胖客户端的RAD开发，早在Web之前就已经非常成熟。不过本文在此提出了一种新的所见即所得的概念，那就是将这种方式不仅体现在开发中，还要体现在最终的成型产品中，也就是允许客户化定制UI的表现。这就要求框架提供UI的组件，要支持动态拖拽的行为。 对于动态的拖拽行为，本文认为合理的方式应采用包装类方式，将动态处理行为封装在包装类里面，当UI组件实例往页面添加的时候，将组件装入包装类。用户进行UI设计的时候，包装类卸载UI组件原先所有的和动态处理有关的事件，并且缓存这些事件对象，然后给UI组件装在上动态处理行为。 与传统表现层技术相比：RAD的动态性将UI的设计全部移交给用户，而程序员，专注于实现业务细节，并且给予用户足够的客户化支持。而原先的方式，程序员不得不为UI的设计煞费苦心。 