基于语义的表单建模

作者 徐立宇

摘要：

本文分析当前表单技术的发展和使用需求，总结了表单建模中所遇到的问题和解决思路。

提出了语义表单的设计，并在实践中分析了设计过程中需要考虑到的问题点。

关键词： 语义 表单建模 可视化 表单设计器

引言：

纵观 WEB 表单技术十几年来的发展，大致遵循了两条主线：

1 纵览最早的 HTML 到 XML、XHTML 再到 HTML5 中的有关表单的规范部分，越来越倾向支持机器阅读和规范的标准化。

2 逐渐重视客户端的计算能力，需要更友好的交互体验（UE），表单技术也开始由简单标签向功能构件发展。

从这两个主线中，我们看到的是的最终用户要求和使用环境日益多元，对展现和交互需求越来越高。

从企业角度出发，希望从表单技术发展中获得收益，以提高 WEB 表单开发的效率和质量，同时为客户提供更友好的使用体验和更高效的定制服务，提升产品的竞争力。

具体分析客户以上的需要，总结了一些 WEB 表单的能力需求：

1 表单需要支持丰富的外观表现能力和交互能力。

2 在满足 1 的前提下，还需要提供针对客户的二次开发支持。

3 在满足 1、2 的前提下，需要维护表单的整体设计，不能将表单实现变成项目现场“定制”。

4 在满足 1、2、3 的前提下，需要支持表单与第三方标准协议的对接，比如 xforms，满足集成业务的需求。

5 在满足以上前提下，需要支持表单构件的积累，避免重复开发，浪费资源。

6 表单设计过程应该友好，需要提供表单设计器加快设计效率和质量。

在分析以上问题后，我们提出了一个基于语义的表单建模方案以及可视化表单设计器。

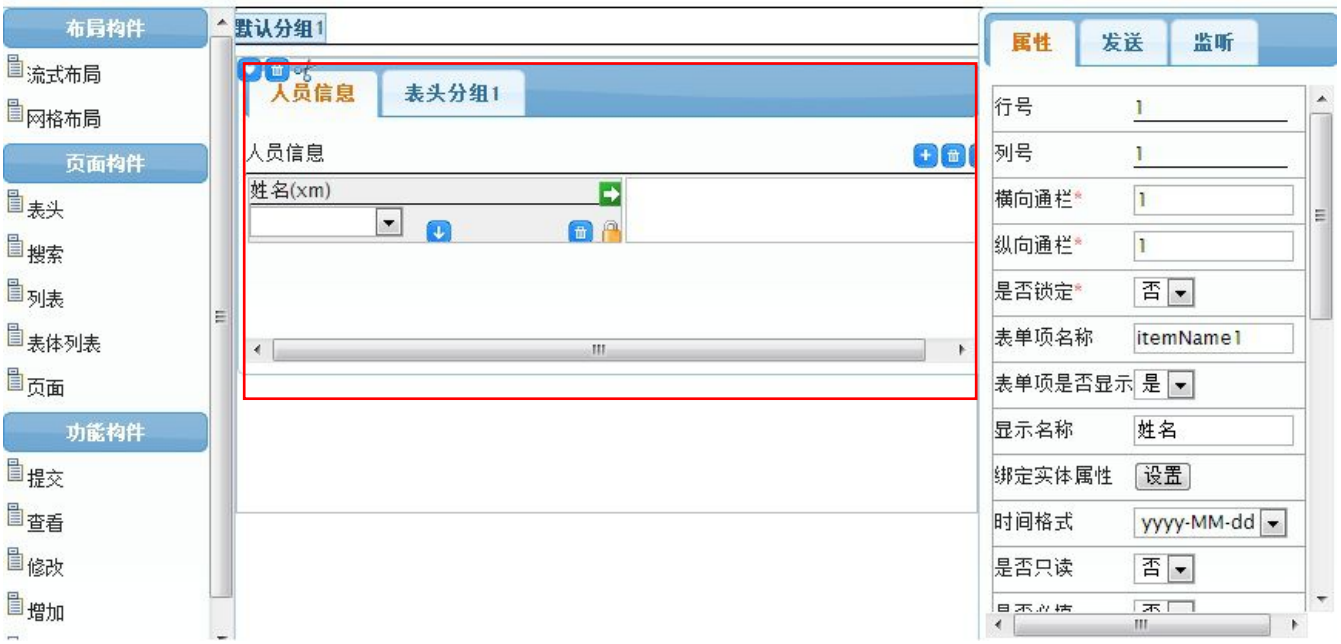
表单建模的意义

为了达到引言中提到的能力需求，关键的一条在于表单是可以被“理解”的。

这里的理解不是指人类直接对于表单进行阅读，而是在表单的规范层面，需要定义能够描述表单自身的语义，从而让表单可以被任何遵循此表单定义规范的人或者机器阅读或者识别、或者格式转换。

我们把对表单的语义的定义过程称为表单建模。

举例如下：



上图红色圈住的是一个叫表头的表单构件，并且当前选中的是第一个页签下的第一个单元格，在右边属性栏列出该单元格下的所有属性，这些属性就是针对这表头构件的语义规范定义，这些属性的定义将最终保存到一个 XML 的文件中（或者保存到数据库中），作为此表头构件的语义描述。

这样，我们得到了一个标准的表单语义数据，该数据将会在以下场景发挥作用：

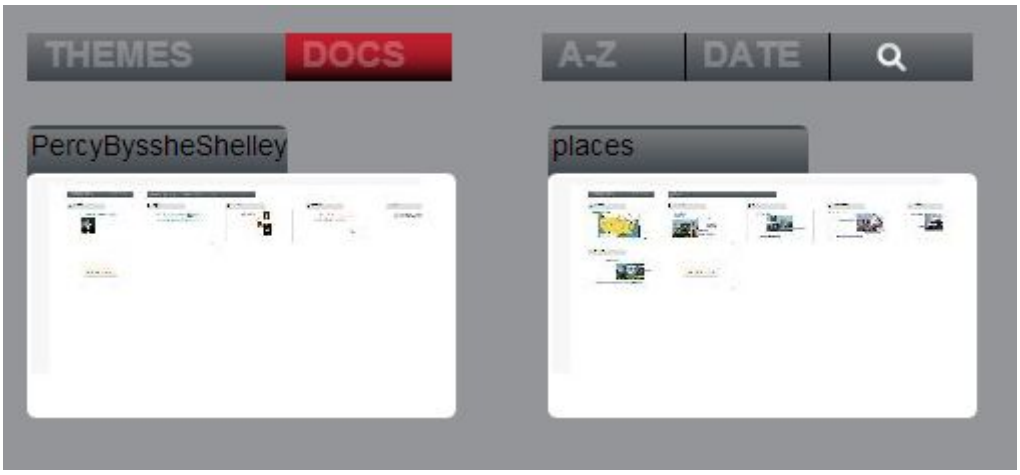
1 可以在运行时读取该语义数据，渲染出该表单。

2 该语义数据可以转换为其他格式，比如 Word、openOffice、xForms、PDF 等第三方表单技术。

3 针对不同的客户端，可以有针对性的生成表单，从而解决了适应多种表单运行环境的需求。

4 由于语义定义与最终的展现 UI 分离，从而可以达成不同的客户针对同一个表单要求不同的外观主题样式的需求。

5 在读取了语义数据的运行状态后，实际上已经完成了针对表单进行状态保存和恢复；同时，可以将这些语义数据进行重新的组合，从而得到客户所期望的个性表单，甚至可以支持到互联网场景下才具有的一些高级特性，比如表单的放大、缩小，复制、剪切、redo、undo 等(参见下图)。



该截图从之前完成的真实项目取得，此为一个幻灯片的缩略图，该缩略图直接由真实的幻灯片缩小而来。

幻灯片见下图：

6 进一步需要指出，在定义了表单语义后，进一步定义统一的 API，从而厘清了产品和二次开发的边界，规范了开发实施过程。

表单建模的设计

为了达到表单可以语义化的目的，需要首先对表单包含的各个部分进行分析。

总结表单的构成，包括以下部分：

1 布局

2 构件

3 构件的事件和消息

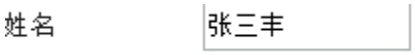
4 构件之间的交互

如果只是这样的划分，看上去还是比较的简单的。

但是深入分析后发现，事物的复杂度往往出乎直觉。

首先，我们来看看构件，以下是所列举构件的几种形式:

1 文本框构件



2 表头构件

很明显，文本框构件是“独立”的基本构件，同时它又是组成表头构件的一个“部分”。

于是，问题开始变得纠结起来：

作为一个独立的构件，是具有自己对应的语义和行为及事件等的。

但是当某个构件成为了另外一个构件的组成部分的时候，它自身所带的语义、行为和事件将成为干扰，在这种情况下，需要有对应的设计去管理他们的组合关系。

比如文本框的语义包括构件类型、大小、位置、显示值等，当成为表头构件的一个部分的时候 ,却相应的定义了对应单元格的名称和显示值。从表头构件的角度来看，是不存在文本框构件的，而只有对应的单元格语义定义。

更进一步，由于构件都是能够接收和发送消息的，那么如果是“独立”的文本框构件，它也是有这样的功能的，但是在表头构件中，文本框构件这个功能需要屏蔽，不能直接暴露出来。否则外部程序不能区分到底是表头构件还是文本框构件，造成干扰，导致逻辑紊乱。

当我们再进一步分析后，还会发现，除了消息需要考虑组合的情况、其他比如事件、属性、方法等都需要考虑到。

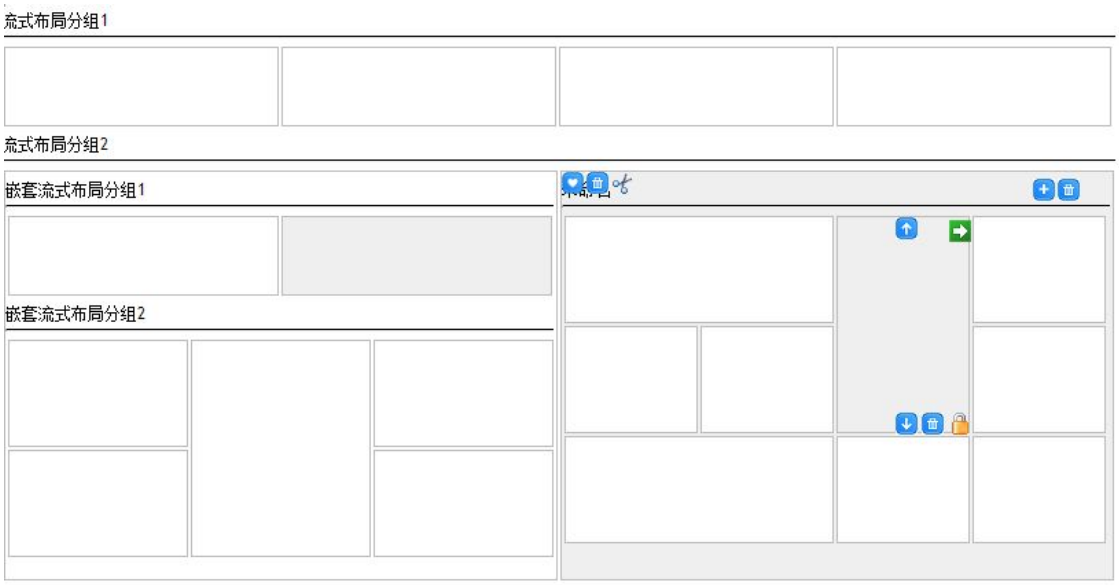
为了统一概念，我们将基本的不是由其他构件组合而来的构件称为基础构件，而将组合构件称为页面构件。

所以，在表单建模框架的设计，首先需要解决以上这些问题。

第二，从执行效率上考虑，表单的渲染需要和语义数据做分离，这样，避免修改一条语义数据就执行渲染，而是可以选择为一批量修改语义数据后做一次渲染，保证了执行效率。

第三，表单布局，我们设计目标是达到覆盖绝大多数的布局需求，这样就对布局的设计提出了更高的要求，在此前提下，我们设计了支持无穷嵌套的流式布局构件。

如下图：



同时也支持剪切、复制、缩放、高度适应内容、另存模板等高级特性。

第四，设计器和运行时都需要支持和积累新的构件开发和部署。

考虑到用户友好性，还需要支持属性定义面板的定制。

第五，关于构件之间的交互，一般情况下，都可以事先预制到我们称之为“功能构件”的一类特殊构件中，通过简单的属性设置就可以完成比如查看、提交、修改、导出等等典型的操作。

如下图场景：

选中了“删除”这个功能构件，右边属性栏可以设置绑定到哪个列表构件，在绑定后，列表已经具有了功能构件中固化的对应处理逻辑。具体到这个例子中，在删除构件的绑定构件属性设置后 ,对应的列表就具有了删除记录的能力。

最后，需要指出的是，构件不仅仅指前端的 js 部分，还包括了对应的服务器端业务逻辑，同时还包括了设计时和运行时的相应资源，以及该构件所依赖的资源和条件的定义。

第六，定义了如上的这些构件类型，必然会涉及到构件本身的完整的生命周期管理，

表单工具需要提供设计时、创建、初始化、运行时、删除的完整支持。

同时，构件实例也是运行在客户端环境下的，需要考虑到内存的管理和回收，防止内存泄漏。

最后，表单系统需要提供完查找构件对象的方法，便于用户二次开发使用。

结束语

综上所述，基于语义的表单建模较好的提升了开发效率和开发质量。同时，满足了对于构件的积累需求。为公司产品化转型提供了技术的支撑。

作者联系方式（移动电话） 13951743010

邮箱 xlyyc@163.com