Boxsys 2.0 架构设计

版本：0.1
作者：射天

日期	修改人	修改内容
2014/4/22	射天	创建文件

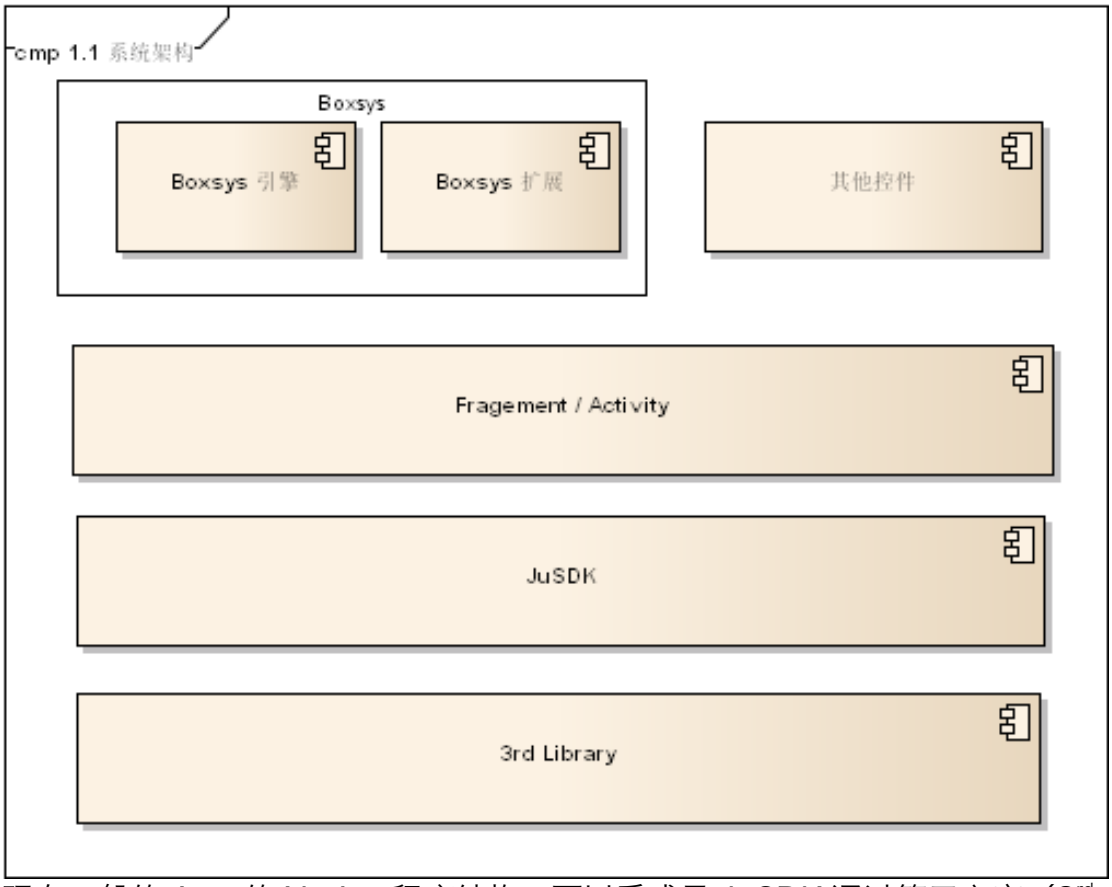
0. Boxsys 2.0 引擎

Boxsys 系统是一种使用模板和数据动态构造显示的系统。他有如下特点：
（1）动态控件布局显示；
（2）显示内容全部是等比缩放计算出来；

和 HTML 相比，他如下差异：
（1）HTML 是在服务器端将数据和模板文件（.vm）通过 velocity 合并成 HTML。客户端根据 HTML 渲染。而 Boxsys 则是在客户端实现这个渲染过程。
（2）Boxsys 如果要扩展显示，增加新功能（动画），通过集成已有的 BoxModel/Box 两个类来实现。

为什么要将 velocity 这样的模板合并过程在客户端实现呢？
（1）因为模板是相对不变的东西，不需要频繁从服务端下载。随着模板越来越复杂，模板数据也会越来越多。
（2）如果由客户端来管理模板的合并，客户端就可以自由嵌套已有的模板，组合出千变万化的显示效果。

1. Boxsys 2.0 架构



现在一般的 App 的 Native 程序结构，可以看成是 JuSDK 通过第三方库（3rd Library），从服务端的 mtop 接口获取数据，然后交由 Fragment/Activity，根据业务端逻辑渲染出 Native 控件。

而 Boxsys 系统是和这个结构兼容的。Fragement/Activity 可以部分由原先的 Native 控件渲染，部分由 Boxsys 渲染。也可以整个页面由 Boxsys 系统渲染。

Boxsys 首先使用 JuSDK 获得获取模板数据。模板数据定义了每一块显示需要关心的内容。比如宽度，高度，相对位置，颜色等等相对不变的内容，也定义了渲染是数据的相对名称（就是后面说到的渲染时数据在 Map 中相对名称）。

我们定义 Map 的项目名称是指唯一标定 Map 内数据的名称，例如有一个 Map:

“data”:{

“image”：“http://taobao.com/xxx.jpg”

}

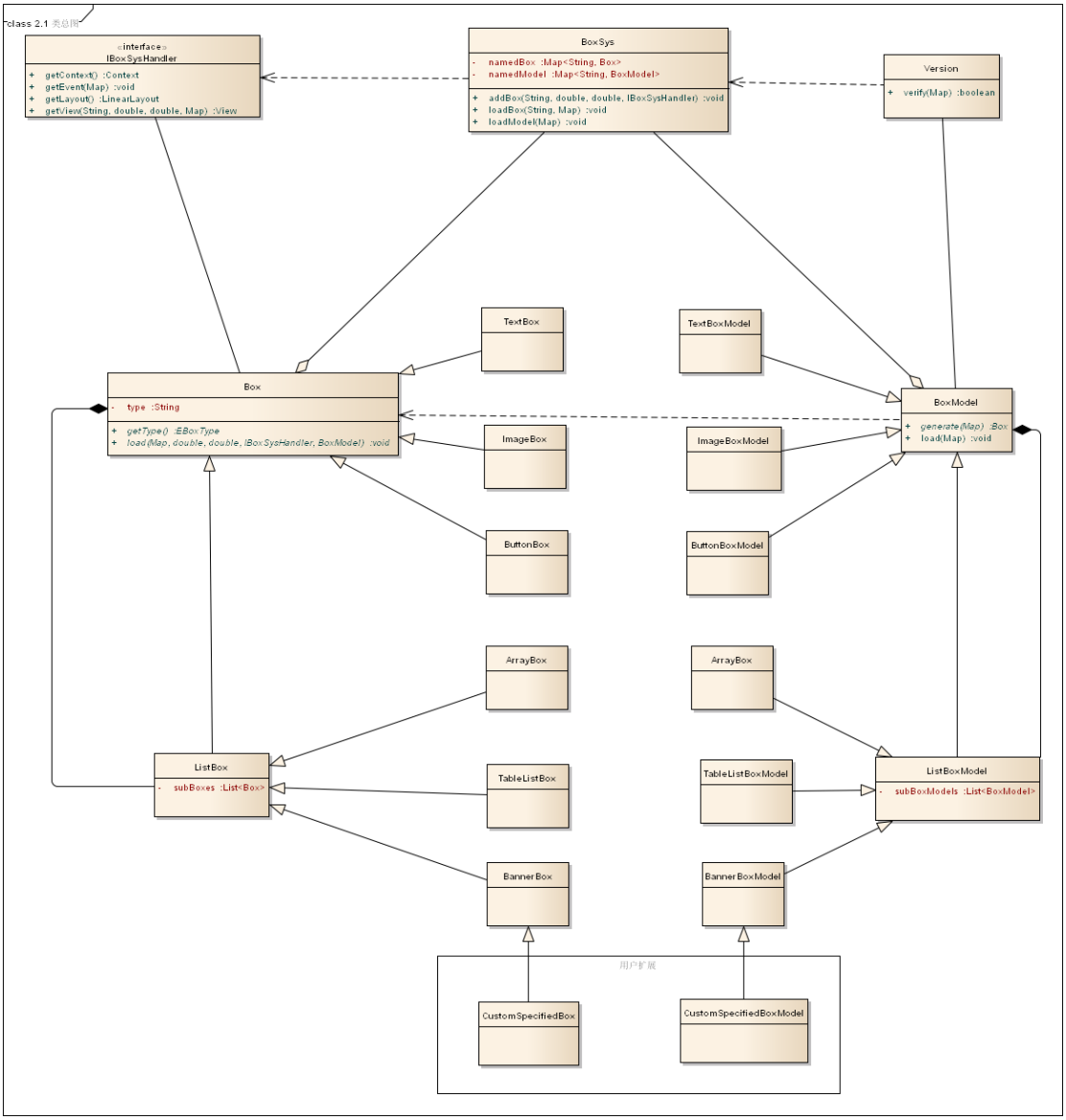
相对名称“image”，就指定了 data 数据中的值为 http://taobao.com/xxx.jpg 的图片地址。

现在 JuSDK 使用 MiscData 接口访问名为 model 的模板数据。对于 Boxsys 来说，他需要的模板数据就是一个 Map，他不关心这个数据是存在后台还是存在本地文件，也不关心这个数据如何存放。这样的设计，是为了保证 Boxsys 能作为一个库，很容易地在各种 App 里使用起来。

Boxsys 然后使用 JuSDK 获得渲染显示用到的数据。同上，对于 Boxsys 来说，数据只是一个 Map，所有业务相关的内容，也是在 JuSDK 里完成的。

2. Boxsys 2.0 类图

2.1 总览



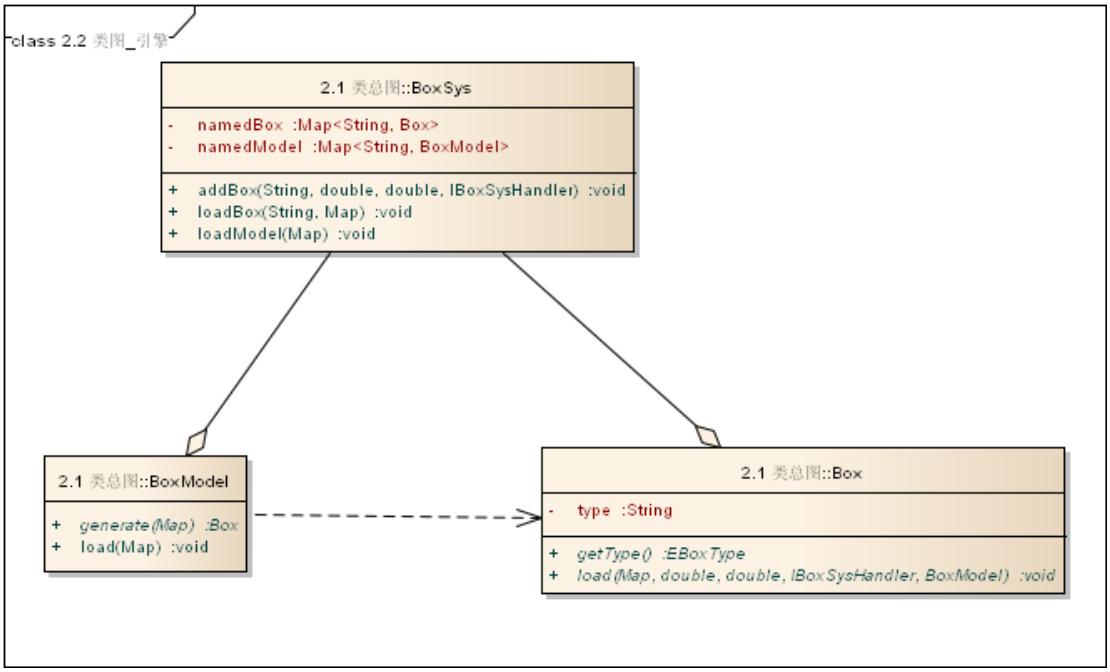
通过上图，可以看到，BoxModel 和 Box 是抽象类，所有的实现子类都是一一对应的。两者完成不同的功能。

（1）BoxModel，用于解析 Model 数据，获取一些布局相关的不变的配置。

（2）Box，用于解析 Data 数据，获取每次渲染不同的数据。

如何区分这两者呢？举个例子，对于如下这种显示。每个框的位置，宽度，高度，因为每次都不变，所以应该放到 Model 数据里。而每张图片都不同，图片的 URL 信息应该放到 Data 数据里。

2.2 引擎部分



引擎最核心的就是这三个类：

Boxsys：

这个是外部访问的入口，从 Fragement/Activiey 只能调用到 Boxsys 入口，其他的一切都是封装在引擎里面的。

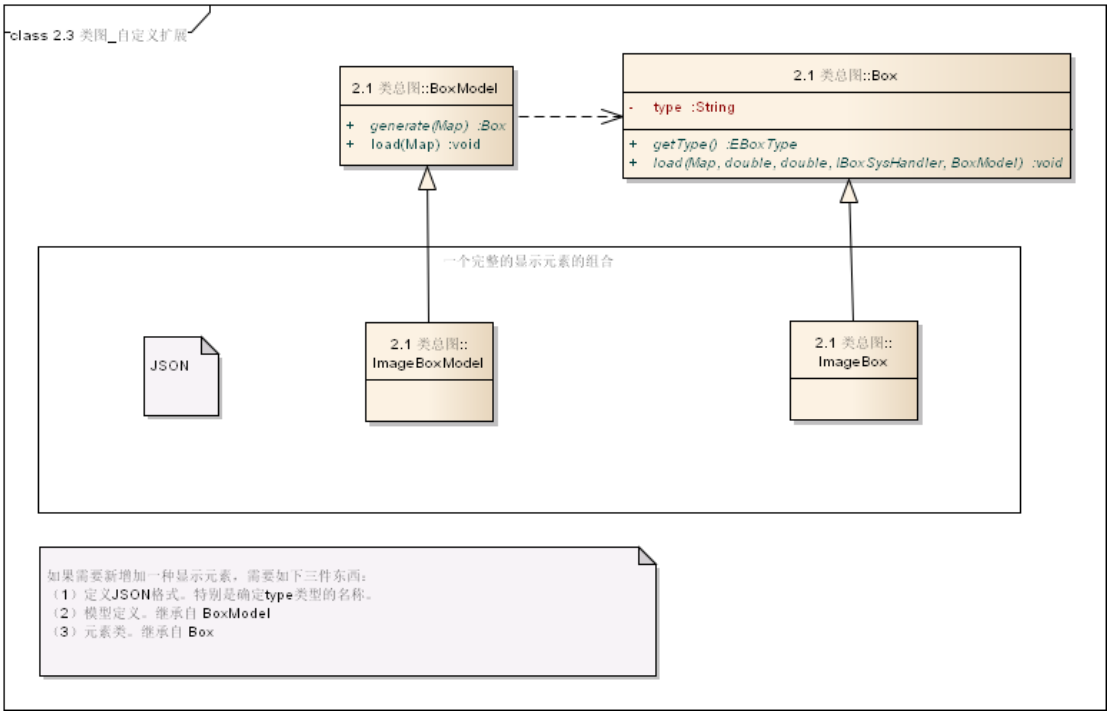
BoxModel：

所有模板 Model 类的基类。所有相同类型的 Box 对象，共用一个 BoxModel 对象。

Box：

所有显示元素的基类。页面上每个显示的部分，对应到一个 Box 对象。

2.3 自定义扩展



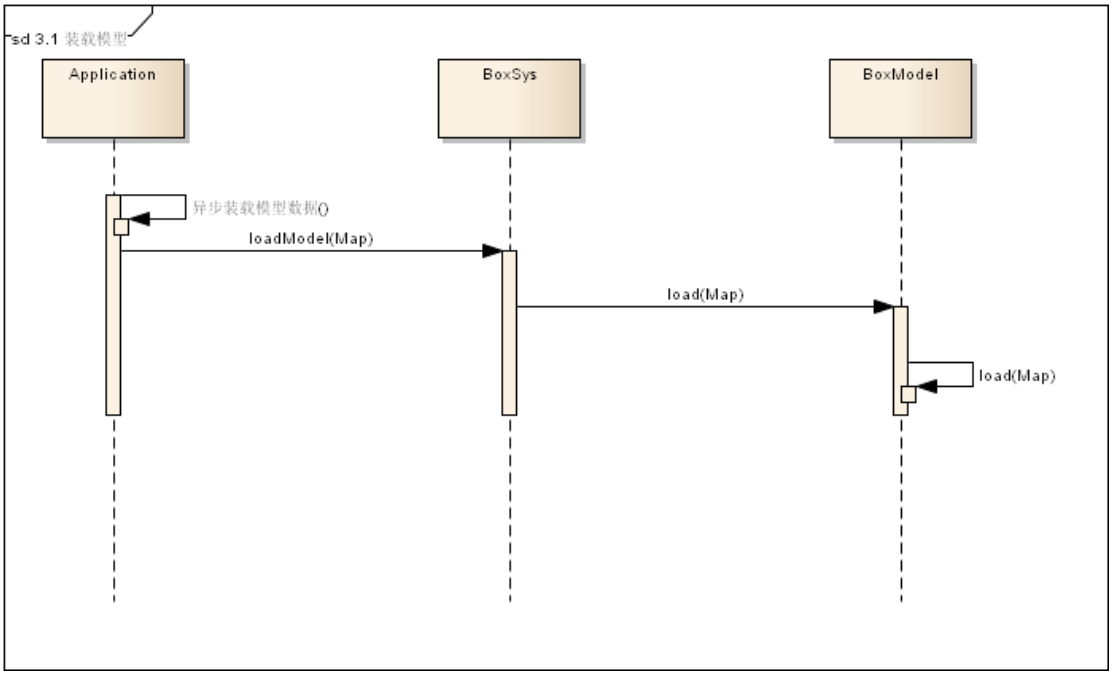
如果发现目前的 Box 不够用，Boxsys 也考虑到了，他提供了扩展 Box 的方法。

如果需要新增加一种显示元素，需要如下三件东西：
（1）定义 JSON 格式。特别是确定 type 类型的名称。
（2）模型定义。继承自 BoxModel。
（3）元素类，继承自 Box。

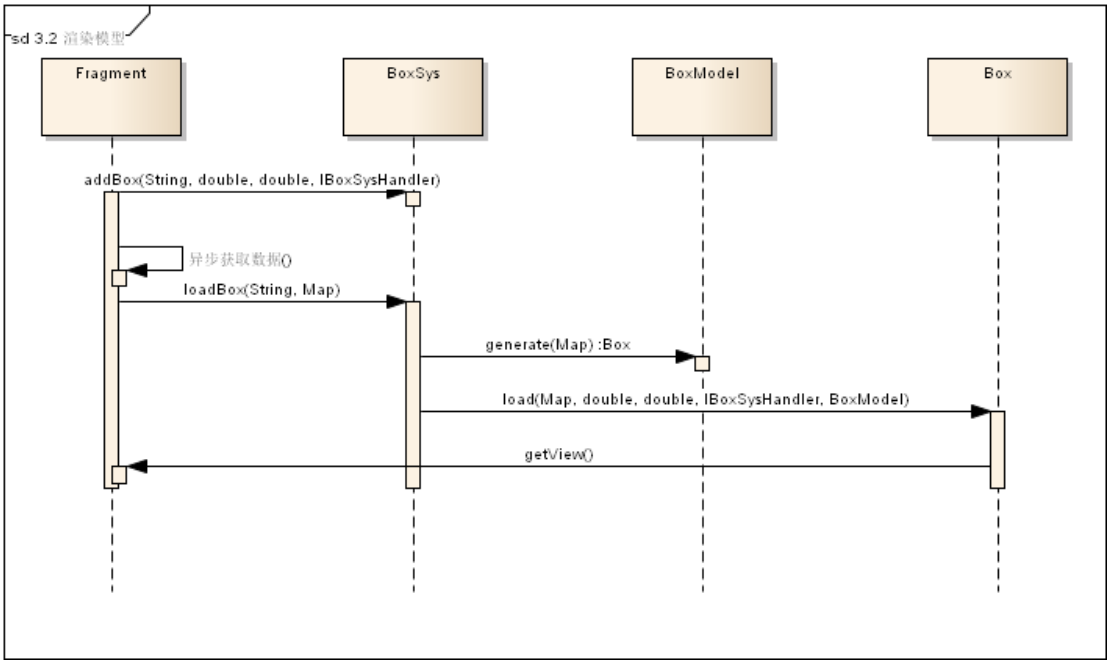
3. Boxsys 2.0 序列图

Boxsys 的使用过程，如上面所述，存在如下两个阶段：
（1）使用 Model 数据，初始化 BoxModel。
（2）使用 Data 数据，渲染 Box。
下面将详细描述这两个过程。

3.1 初始化时



3.2 显示 Activity / Fragement 时



首先，每个显示区域，需要使用 addBox 加以定义。这样，以后显示相同名字的区域时，可以从内部的 cache 直接获得显示内容，节约流量、内存和时间。

然后，当数据得到后，使用 loadBox 将数据渲染出指定名字的区域。在渲染过程中，Box 会调用 IBoxSysHandler 接口，获取 Native 的控件。