

Practice MVC as features, not frameworks

新版阿尔法城背后的前端MVC实践

DexterYy @ 豆瓣

阿尔法城是神马？



阿尔法城 / 六坊

14人在线



阿尔法城有了自己的坊，这是其中之一。坊下有街，坊的名字和描述会在合适的时候由它的成员们决定。坊的风格和定位由其下街道的居民来塑造，欢迎你的到来和参与。

六坊大道的小店

>店铺租赁



豌豆原盘打口碟CD店
卖CD~ <http://onlycd.taobao.com/>
小店里有小豆领哟！

2



萌兽犹斗
完全没装修好，欢迎送豆.....

4



心爱的树
Beta

6

8

10

12



公告栏

欢迎来到六坊。六坊暂时拥有9条街道。请选择您喜欢的街道入住或参观。

坊名将由本坊的九位街道服务长共同决定。街道服务长请将自己认可的候选坊名以豆邮投递给F区临时服务长：[伊欧莫的伽马](#) 区临时服务长在得到9个街道服务长的全部方案后将公示他们的选择。具有过半优势的选项将成为正式坊名，否则由九位街道服务长在前二选项中进行二选一投票决出最终结果。

鲁棒街

三体街

飞面神教街

短路

显微径

华尔街

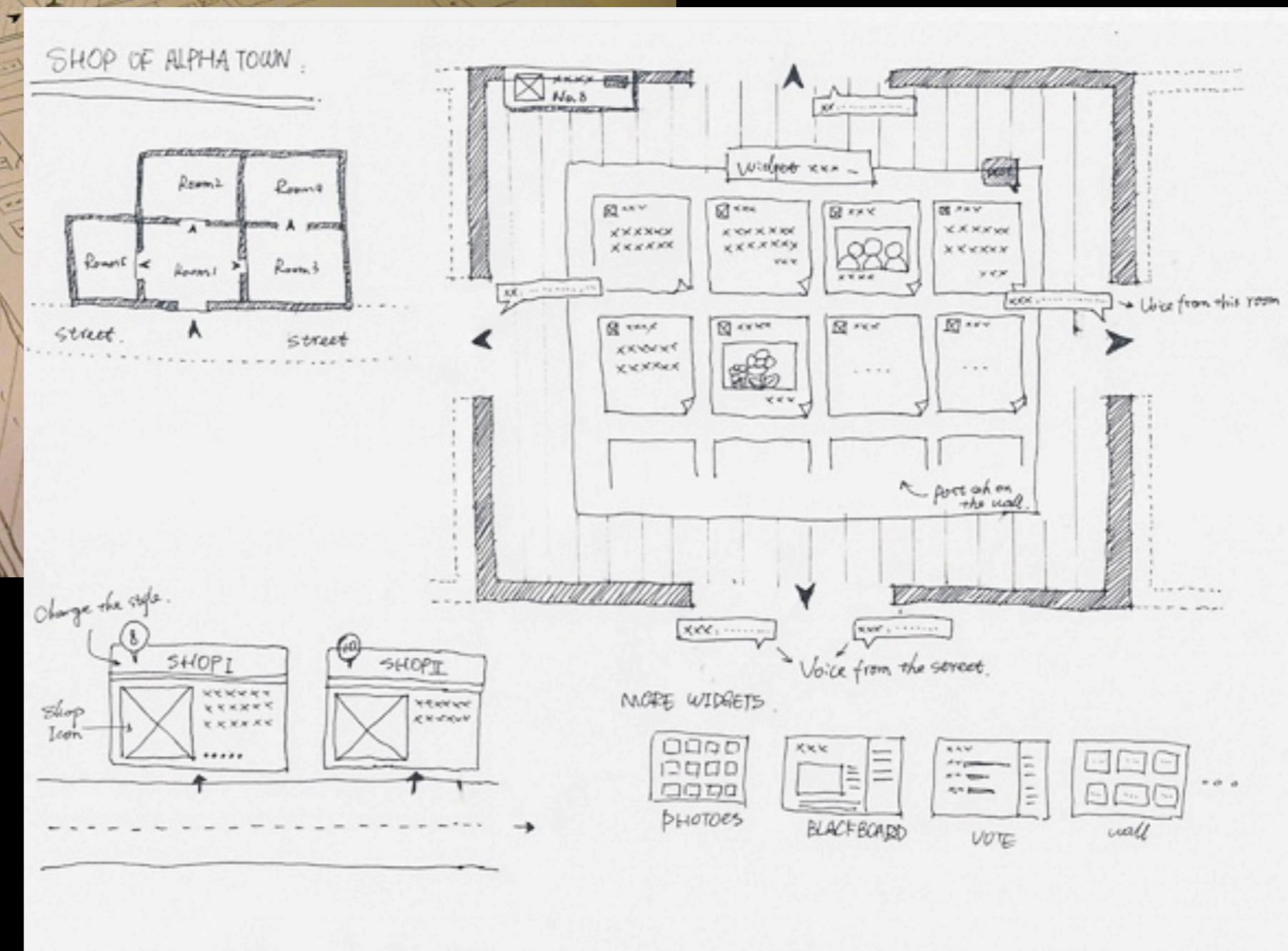
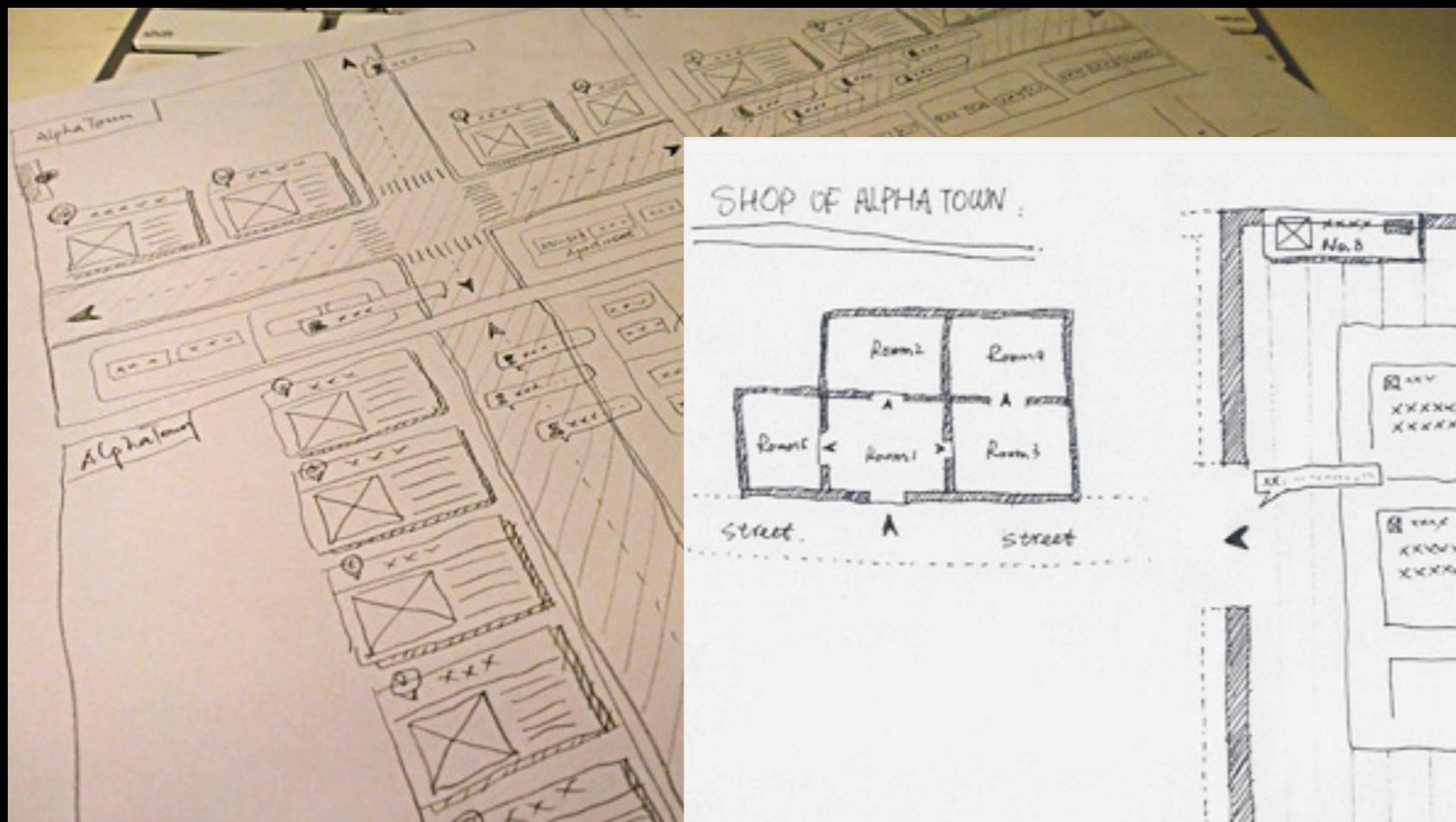
泰坦之路

超链街

猫街

当当当挡！.....好罢这是旧版

<http://site.douban.com/118836/>



牛逼闪闪的新版！

其实还没正式上线呢！

.....

事儿不能说太细！

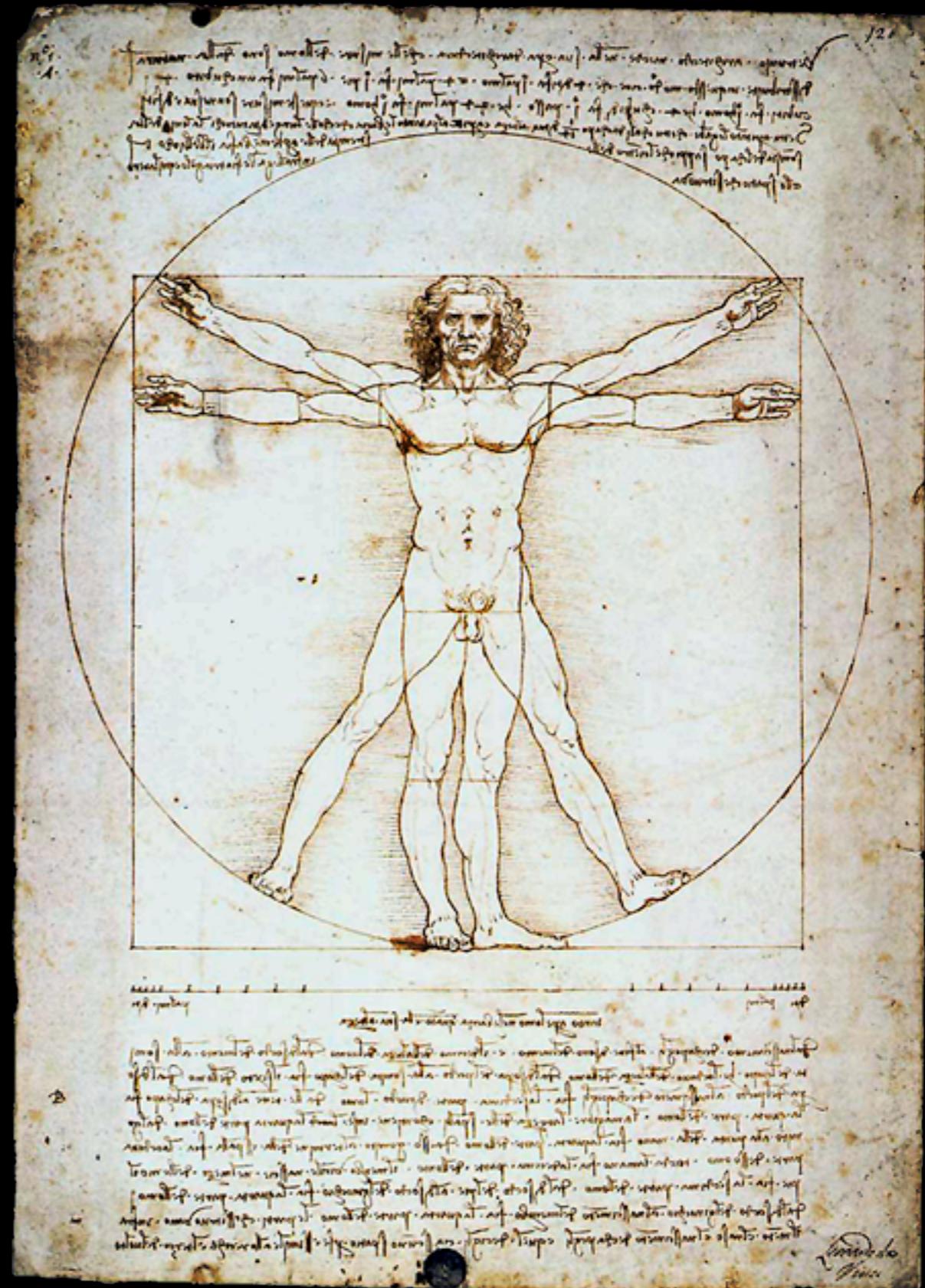
.....

演示一下开发服务器
上的可运行版本吧！

.....

首次公开喔！

.....



从技术角度看阿尔法城

(不谈产品)

- 一个“地理位置”应用 (不是LBS)
- 一个“虚拟现实”应用 (不是3D)
- 一个“AJAX地图”应用 (不拼图片)
- 一个重度图形化的web app (不用Flash)
- 一个前端密集型项目 (昨天还在加班…)

先声明一下
为什么要分析阿尔法城的特征

不是为了做广告XD

而是因为这些特征直接决定了开发中

先声明一下
为什么要分析阿尔法城的特征

不是为了做广告XD

而是因为这些特征直接决定了开发中

- 为什么要用MVC
- 要用怎样的MVC

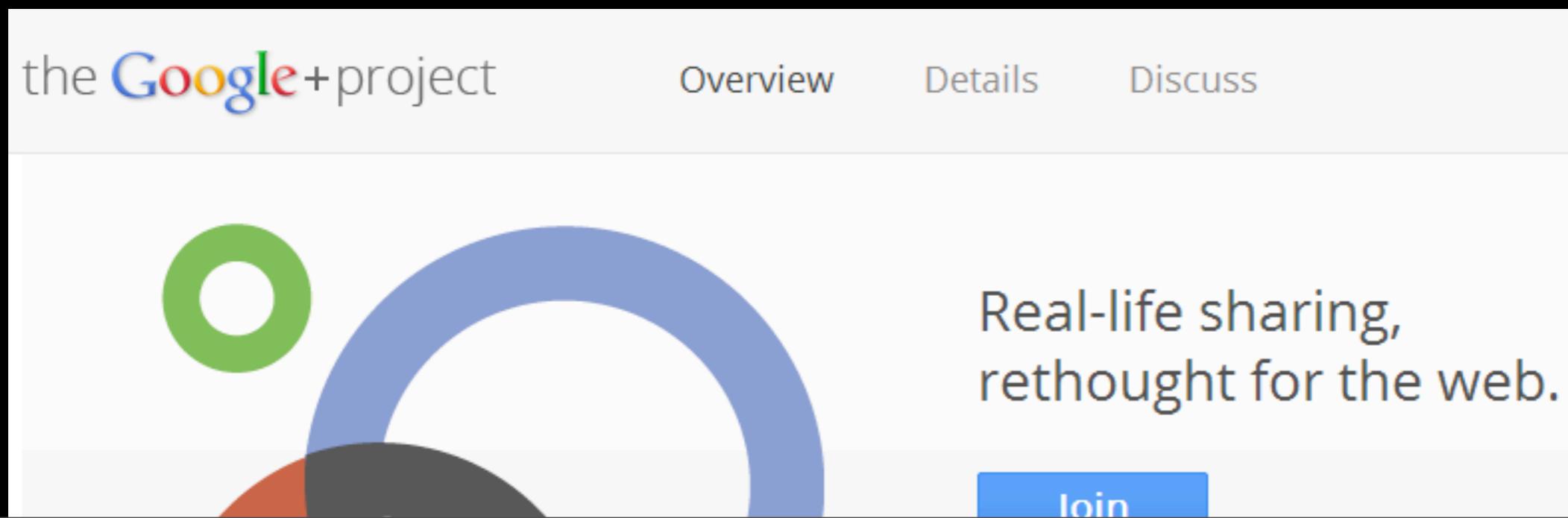
一个“地理位置”应用

- 完全没有超链接，用方向来导航，在相邻的“地理节点”之间移动
——键盘操作 (WASD) 与屏幕点击
- 视图之间有位置关系（东南西北、内部外部）
——矩阵 vs 图 (Graph)
- 未见其人，先闻其声
——跟服务器端的“实时”通讯



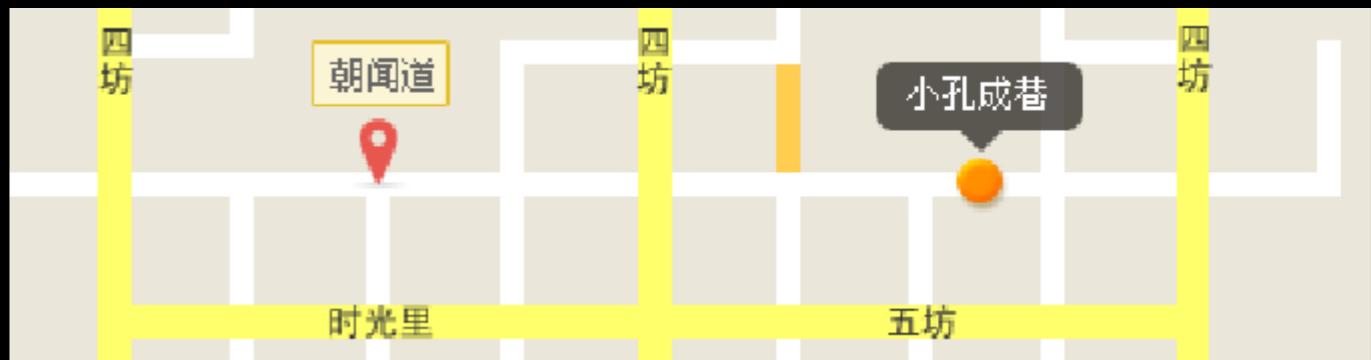
一个“虚拟现实”应用

- Bring real-life to software, rethought for the web
(Google+的slogan喔！XX所见略同，嗯嗯…)
- 虚拟的是抽象的现实，不需要逼真的3D图像
- GUI需要平滑自然亲切的隐喻（Metaphor）：无处不在的过渡动画效果



一个“AJAX地图”应用

- 无边界的画布/沙盘，分段加载，渲染当前视区（viewport），类似google maps
- 随用户行为而自然生长的城镇地图，不能用静态图片拼接，要根据JSON数据绘制图形元素
- 不能任意拖拽，只能以整屏为最小单位来移动
——传统web的本质：状态表述转移
- 相对静态的、粗粒度的“地形”；变动频繁、细粒度的遮盖物（overlap）；按需加载、自由而孤立的widget



一个“重度图形化”的web app

- 传说中的Single Page App ([wiki](#))
- 不是传统网页的文本流，不是传统软件GUI的栏式布局——需要计算layout
- 自适应屏幕大小，始终全屏，无矢量缩放——需要在客户端计算layout

一个“前端密集型”项目

- 后端不处理任何涉及外观和交互的逻辑，而是根据前端的要求，提供数据的特定视图（View，参考CouchDB）
- 前端程序不是依附于后端视图的“脚本”，而是包含渲染引擎和完整框架的客户端
- 前端程序的生命周期无限延长，需要自行管理大量的资源，处理复杂的状态组合
- 需求集中在前端：持续增长的、可插拔的“子系统”

阿尔法城前端开发面对的 问题是否有普遍意义

说没有肯定是一种zhuangbility...

当然有很多啦～比如：

- 分离不同种类的逻辑，减少依赖关系，降低扩展和维护的复杂度
- 在多种输入方式之间复用代码（鼠标、键盘、触摸、URL）
- 能用代码再现任何种类的用户输入，具备可测试性
- 在多种输出方式之间复用代码
- 处理异步的状态变化
- 单页webapp的需求越来越普遍

解决前四个问题
正是MVC的初衷

阿尔法城的前端MVC

阿尔法城的前端MVC

- 没有使用现成的MVC框架
- 没有开发全新的MVC框架

阿尔法城的前端MVC

- 没有使用现成的MVC框架
- 没有开发全新的MVC框架

异议！MVC实践在哪里？



阿尔法城的前端MVC

- 没有使用现成的MVC框架
- 没有开发全新的MVC框架

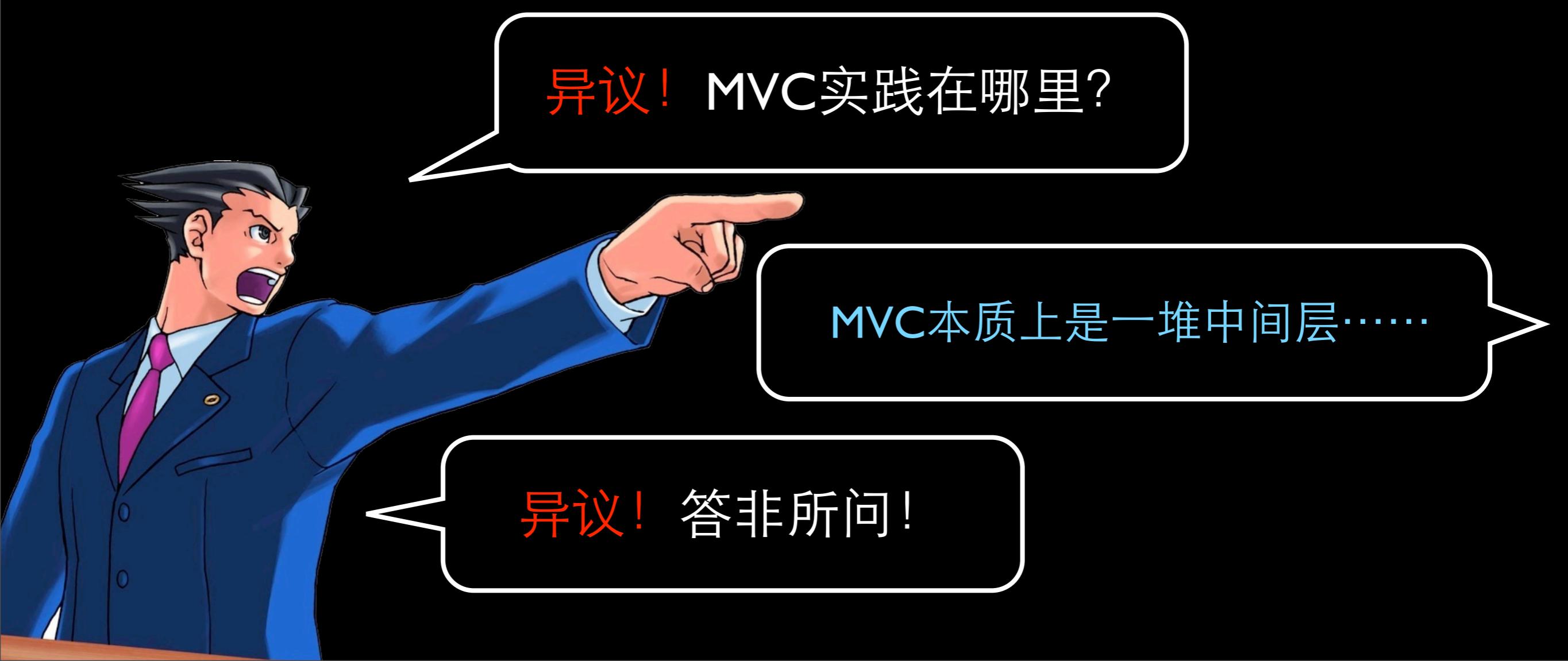
异议！ MVC实践在哪里？

MVC本质上是一堆中间层……



阿尔法城的前端MVC

- 没有使用现成的MVC框架
- 没有开发全新的MVC框架



每增加一个中间层，
都需要明确的理由

个人瞎编的伪语录……

奥卡姆剃刀原理

- 如果两种说法都能解释相同的事 实，相信假设少的那个（科学家的信条）
- 如果两种方法都能解决相同的问题，使 用更简单的那个（产品经理的信条）
- 如无必要，勿增实体（开发者的信条）

前端MVC需要哪些基本功能 & 阿尔法城的解决方案

I. 模块化设计

- 阿尔法城基于OzJS
- 所有代码单元都封装成CommonJS风格的module
- 只有显式声明依赖关系，才能持有其他模块中对象的引用，禁用全局命名空间
- 自动管理依赖，平行加载，惰性加载，按需加载
- 《通用JS时代的模块机制和编译工具》

```
<script src="js/lib/oz.js"></script>
<script>
oz.def('jquery-source', 'js/lib/jquery.js');
oz.def('jquery', ['jquery-source'], function(){r
oz.def('jquery-sandbox', 'js/lib/jquery_sandbox.
oz.def('jquery.easing', ['jquery'], 'js/lib/jque
oz.def('domReady', 'js/mod/domready.js');
oz.def('lang', 'js/mod/lang.js');
oz.def('event', 'js/mod/event.js');
oz.def('network', 'js/mod/network.js');
oz.def('uiproxy', 'js/mod/uiproxy.js');
oz.def('key', 'js/mod/key.js');
oz.def('dialog', 'js/mod/dialog.js');
oz.def('contextmenu', 'js/mod/contextmenu.js');
oz.def('drag', 'js/mod/drag.js');
oz.def('url', 'js/mod/url.js');
oz.def('template', 'js/mod/template.js');
oz.def('cookie', 'js/mod/cookie.js');
oz.def('storage', 'js/mod/storage.js');
oz.def('dataSource', 'js/mod/datasource.js');
oz.def('mapViewer', 'js/mod/mapviewer.js');
oz.def('AlphaTown::App', 'js/app/alphatown.app.j
oz.def('AlphaTown::Model', 'js/app/alphatown.mod
oz.def('AlphaTown::GC', 'js/app/alphatown.gc.js'
oz.def('AlphaTown::Toolkit', 'js/app/alphatown.t
oz.def('AlphaTown::VirtualRoute', 'js/app/alphat
oz.def('AlphaTown::View', 'js/app/alphatown.view
oz.def('AlphaTown::View::TownMap', 'js/app/alpha
```

- 显式声明依赖关系，在开发过程中有助于减少不必要的依赖，实现最大化的解耦
- 强制约束，让Model和View之间无法互相调用
- 帮助养成“MVC的习惯”

```
oz.def("AlphaTown::App", [
  "lang",
  "url",
  "key",
  "cookie",
  "storage",
  "AlphaTown::Model",
  "AlphaTown::View",
  "AlphaTown::Observer",
  "AlphaTown::GC"
], function(_, URLkit, Key, cookie, storage, model, view, observer, gc){
```

```
oz.def("AlphaTown::View", [
  "jquery",
  "jquery-sandbox",
  "lang",
  "event",
  "uiproxy",
  "template",
  "contextmenu",
  "AlphaTown::Observer",
  "AlphaTown::View::TownMap",
  "AlphaTown::View::Fullmap",
  "AlphaTown::View::Town",
  "AlphaTown::View::Shop",
  "AlphaTown::View::House",
  "AlphaTown::View::Widget",
  "AlphaTown::View::Sound",
  "AlphaTown::View::Dialog"
])
```

```
| |-alphatown.view.maplayers.js  
| |-alphatown.view.node.js  
| |-alphatown.view.shop.js  
| |-alphatown.view.sound.js  
| |-alphatown.view.town.js  
| |-alphatown.view.widget.js  
| `--alphatown.virtualroute.js  
| +embed/  
| +lib/  
| ~mod/  
| |-contextmenu.js  
| |-cookie.js  
| |-datasource.js  
| |-dialog.js  
| |-domready.js  
| |-drag.js  
| |-event.js  
| |-key.js  
| |-lang.js  
| |-mapviewer.js  
| |-network.js  
| |-oop.js  
| |-storage.js  
| |-template.js  
| |-uiproxy.js  
| `--url.js
```

- 为项目量身定制的“微框架”
(micro-framework)
- Thomas Fuchs:
[Zepto and the rise of the JavaScript Micro-Frameworks](#)
- Dustin Diaz:
[Ender.js - The open submodule library](#)

2. 消息传递 (message passing)

- Alan Kay: "OOP to me means only messaging, ..."
- 或者称呼它PubSub / EventEmitter / CustomEvent / ... 随便什么都行
- 在阿尔法城的代码里它叫Observer
- MVC分层的基础，逻辑分离的前提

```
oz.def('event', 'js/mod/event.js');
```

```
oz.def("AlphaTown::Observer", ["event"], function(Event){  
    return Event();  
});
```

- Observer无处不在
- 最大的用途：
从View到Controller的通信
- 实例：阿尔法城的对话框组件、
地图加载

```
observer.bind("dialog:open", function(){  
    key.disable();  
    observer.bind("viewport:beforereset", closeDialog)  
});  
  
observer.bind("dialog:close", function(){  
    key.enable();  
    observer.unbind("viewport:beforereset", closeDialog)  
});  
  
observer.bind("fullmap:loaded", function(json){  
    observer.bind("app:init", function(nid, roadId){  
        view.setupFullmap({  
            data: json  
        });  
        fullmap.drawMap(parseInt(roadId));  
    });  
});
```

```
observer.bind("walk:start", function(link){...})  
observer.bind("walk:end", function(nid){...})  
observer.bind("viewport:resize", function(ratio){...})  
observer.bind("viewport:beforereset", function(ratio){...})  
observer.bind("viewport:reset", function(ratio){...})  
observer.bind("viewport:ready", function(ratio){...})  
observer.bind("viewport:rotate", function(ratio){...})  
observer.bind("viewport:polling", function(ratio){...})  
observer.bind("shop:enable", function(nid){...})  
observer.bind("shop:zoom-start", function(pic){...})  
observer.bind("shop:zoom-end", function(pic){...})  
observer.bind("shop:zoom-waiting", function(pic){...})  
observer.bind("shop:zoom-ready", function(pic){...})  
observer.bind("shop:exit", function(){...})  
observer.bind("houseDetail:open", function(){...})
```

- Observer的另一个用途：异步调用
- 实例：阿尔法城任一场景的重绘机制（reset）

```
reset: function(){
    var nid = parseInt(this.vision);
    this.needReset = false;
    this.clear();
    observer.fire("viewport:reset", [nid]);
    return observer.alone("viewport:init");
},  
  
    if (!/,/.test(node)) && node.activelist.length >
        node.reset().wait(function(){
            observer.fire("viewport:ready", [nid]);
        });
    } else {
        node.locate(nid);
        observer.fire("viewport:ready", [nid]);
    }
}
```



- Observer的功能由event模块提供

监听事件/订阅消息/
等待状态转换

“即时”消息

“状态转换”消息

只等待下一次消息
==回调

```
oz.def("event", function(require){  
    var fnQueue = require("lang").fnQueue;  
  
    function Event(){...}  
    Event.prototype = {  
        alone: function(type){...}  
        // actor receive  
        bind: function(type, handler){...}  
        unbind: function(type, handler){...}  
        fire: function(type, params){...}  
        enable: function(type, params){...}  
        disable: function(type){...}  
        // actor react  
        wait: function(type, fn){...}  
    };  
  
    return function(){  
        return new Event();  
    }  
});
```

3. 前端URL Router

- 所有服务器端web框架的标配
- 为Single-page WebApp设计URL不（仅）是为了SEO
- URL可代替DOM事件
- URL是一种用户请求的输入方式
- URL是一种状态
- UNIX设计者心目中的理想操作系统：Plan 9
- “Plan 9 倡导应用程序在运行时都把自己的内部状态抽象成一个文件系统”
- URL是WebApp的“文件系统”

- 阿尔法城的url模块
- router部分只需30行代码

```
observer.bind("shop:zoom-end", function(pid){  
    url.hash({  
        "2": "shop",  
        "3": pid,  
        "tm": false,  
        "dest": false,  
        "step": false  
    }, true);  
});  
url.route("default", function(params, hash){  
    if (!_home_inited) {  
        app.goHome();  
    } else {  
        location.reload();  
    }  
}).route([  
    "/node/:nid/",  
    "/node/:nid/:place/:pid/",  
    "/node/:nid/:place/:pid/room/:did/",  
    "/node/:nid/:place/:pid/room/:did/:action"  
], function(params, nid, place, pid, did, action){  
})
```

4. UI事件的监听和委托 (delegate)

- 阿尔法城使用uiproxy模块
- 截获所有普通UI事件（点击），委托给统一的配置对象来管理
- 避免传统前端程序中DOM事件的肆意滋生，分离和约束相关代码，降低内存风险

```
var view = {  
  
    init: function(opt){  
  
        this.wrapper = opt.wrapper;  
        this.event = uiproxy.add(document, 'click', ui_event);  
    }  
};
```

- 绝大多数情况下只需要用className映射事件
- 故意不支持部分匹配，多个class可以表示状态的改变或叠加
- UI事件不直接映射到View对象的方法，保证View对象的方法不接受DOM参数
- 在View层之外必须能再现任意UI事件的功能
- 实例：用户点击——>进入小店的房间——>用户刷新页面——>自动恢复房间视图

```

var ui_event = {
  ".toward.arrowW": function(){
    hostView.move("W");
  },
  ".toward.arrowE": function(){
    hostView.move("E");
  },
  ".toward.arrowN": function(){
    hostView.move("N");
  },
  ".toward.arrowS": function(){
    hostView.move("S");
  },
  ".shop-overlap": function(){
    var data = this.href.match(/\node\\d+\\shop\\(.*)/);
    shopView.zoom(parseInt(townView.vision), data[1]);
  },
  ".shop-overlap.forsale": function(){...},
  ".shop-admin-btn": function(){
    shopView.showAdmin(this.href.replace(/.*#/,""));
  },
  ".foldfullmap": function(){
    fullmap.fold();
    $(this).addClass("folded");
  },
  ".foldfullmap.folded": function(){
    fullmap.unfold();
    $(this).removeClass("folded");
  },
}

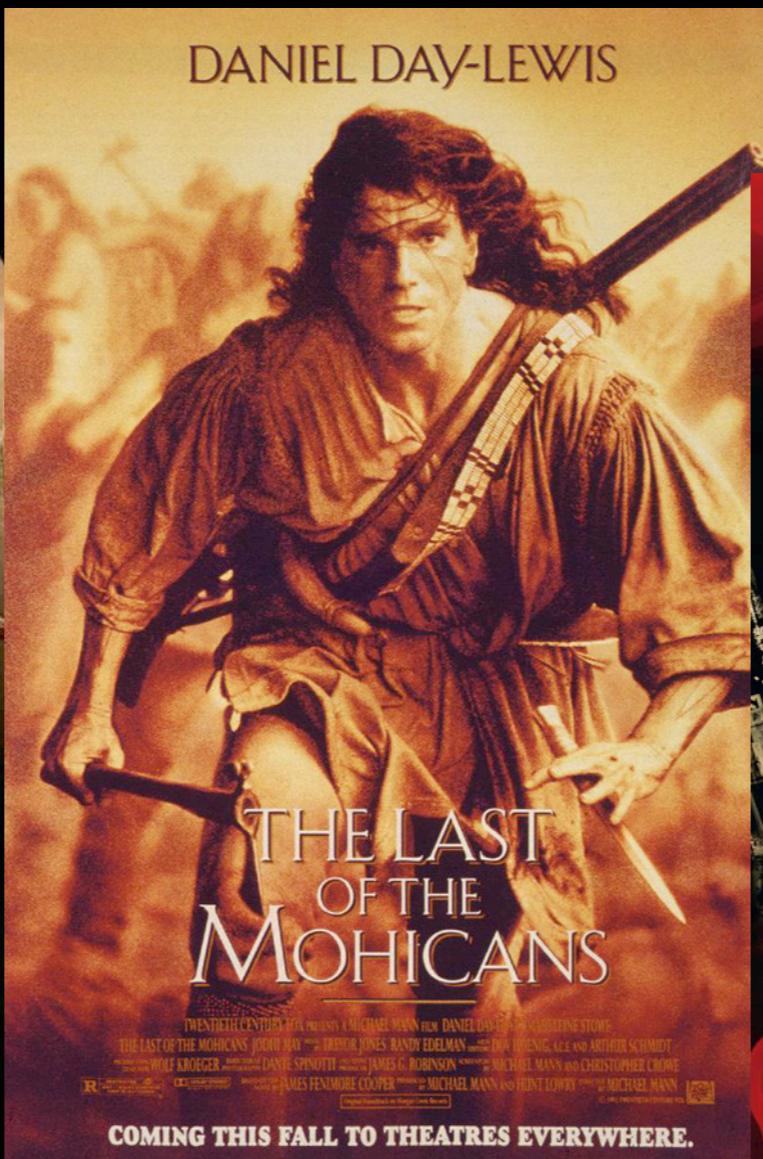
```

5. 视图渲染 (Render)

- 两种方式生成视图：1、模板配置，2、绘图命令
- 两种方式修改视图：1、重绘，2、DOM操作
- 三种API：DOM、SVG、Canvas

5. 视图渲染 (Render)

- 两种方式生成视图：1、模板配置，2、绘图命令
- 两种方式修改视图：1、重绘，2、DOM操作
- 三种API：DOM、SVG、Canvas
- 阿尔法城采用上述的四种方式加一种API



阿尔法城是一个纯HTML4应用

- template模块提供两种模板工具
- format (python风格的字符串格式化)
- convertTpl (micro-templateing, John Resig/Underscore)

```

soundRiver[0].innerHTML = cache.record.reduce([
    return p.concat(c);
}).map(function(o, i){
    return this.format(TPL_SOUNDBOX, {
        l: o[0],
        b: o[1],
        z: o[2],
        c: this.convertTpl("tplSound", sounds[i])
    });
}, viewkit).join("");

```

```

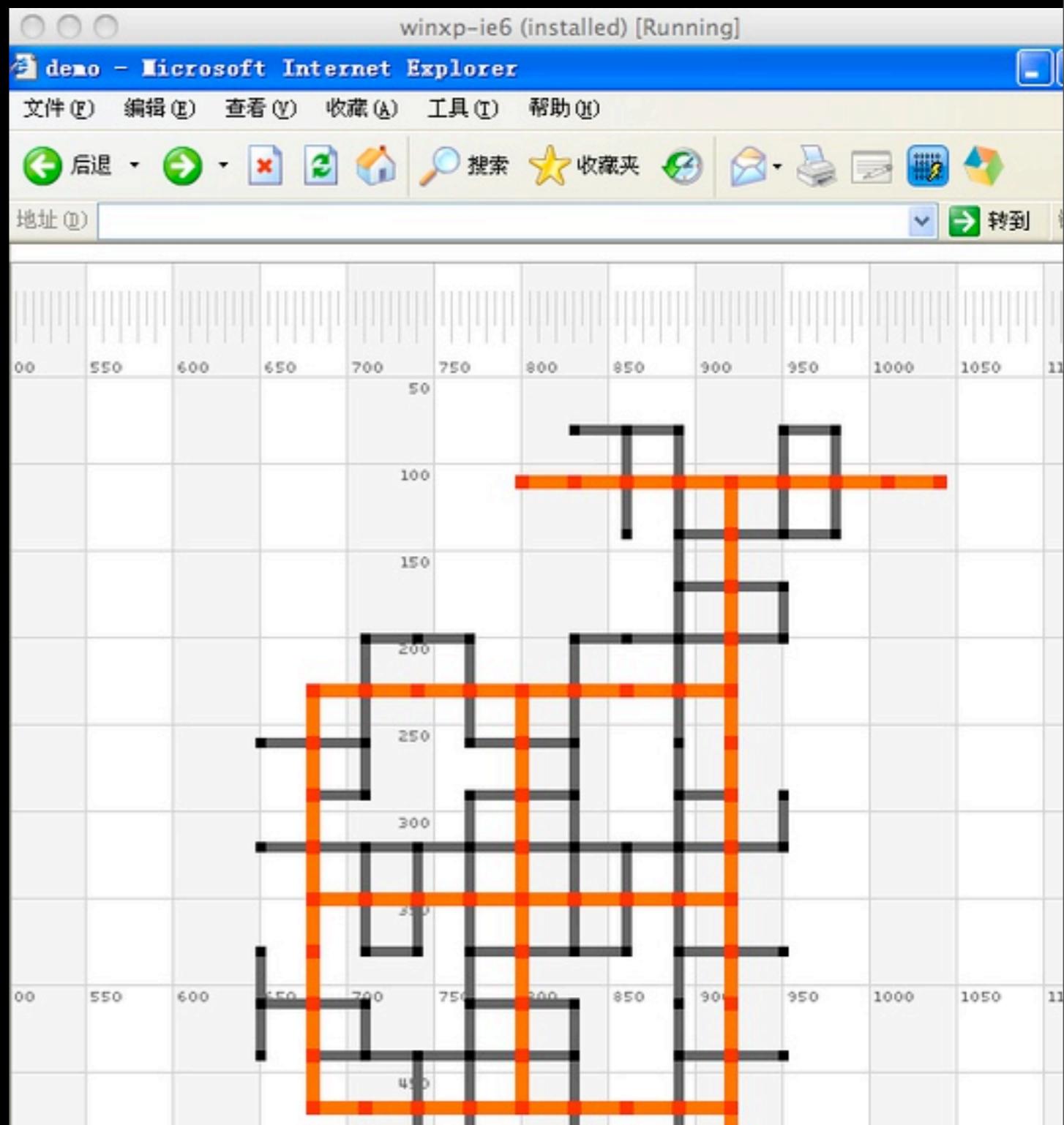
<script type="text/template" id="tplBase">
{> renderList.forEach(function(entity){ %}
    {> if (entity.kind < 4) { %}
        <div class="{%=entity.className%}" style="{%=entity.style%}">
            {> entity.rows && entity.rows.forEach(function(row){ %}
                <div class="row row{%=row.name%}" style="{%=row.style%}">...
            {> });
    {> if (focus.id === entity.id) { %}
        <div class="sound-river" id="sound_river_{%=focus.id.toString()%}"...
    {> } else { %}
        <div class="sound-emitter" id="sound_emitter_{%=focus.id%}_{%=par...
    {> } %}

```

- DomCanvas模块：命令式渲染，在有些需要计算layout的场合更直观。
- 用DOM模拟Canvas API
- 浏览器兼容性（但不是重点，ie6不值得支持）
- 具备SVG的优点，类似工具：raphael.js

```
function DomCanvas(){...}

DomCanvas.prototype = {
    render: function(box){...},
    translate: function(x, y){...},
    fillRect: function(x, y, w, h){...}
}
```



6. 数据源 (DataSource)

- Model层的核心，托管所有数据，隐藏细节和来源，统一访问接口
- 可能需要涵盖网络通信、缓存、本地持久化存储、查询、过滤规则、格式转换、远程同步、实时状态更新、离线测试

```
model("node:view").get({ nid: parseInt(nbid) }, function(data){  
    node.render(nbid, data);  
});
```

```
model("node:all").get({}, function(json){  
    observer.enable("fullmap:loaded", [json]);  
});
```

- 阿尔法城的数据查询和接口配置

```
modelOptions = {  
  
    "node:view": {  
        remote: API_ROOT + "/node/{{nid}}/view?dest={{dest}}"  
        filter: function(opt, origin){...  
    },  
  
    "node:info": {  
        remote: API_ROOT + "/node/{{nid}}/info",  
        filter: function(opt, origin){...  
    },  
  
    "node:all": {  
        remote: API_ROOT + "/node/all_nodes",  
        filter: function(opt, origin){...  
    },  
};
```



异议！有了这些module，前端代码就能自动分离成M、V、C么！

不能，这还需要开发者的理解和支持！（客服语气XD）而且不同项目不同环境下的MVC，分层方式、层与层之间的关系，都可能有差别

阿尔法城的Model层

- 很薄的Model层，几乎不需要本地存储，更类似传统网页，基于URL的状态转移
- Model跟View之间没有事件绑定，不允许直接通信，只能由Controller来查询Model和接收消息
- Model只维护数据本身，不需要持有跟应用逻辑相关的关系。Model返回数据前有make步骤，过滤器会根据查询参数（Controller持有的状态）对原始数据做包装和修改
- 实例：阿尔法城视图的旋转

阿尔法城的View层

- 图形渲染引擎 + UI组件集合（View widget/View对象）
- View对象不允许直接查询或修改数据，不依赖Model和Controller，只能抛出事件/消息（Martin Fowler的称呼：Passive View）
- View对象不含业务逻辑，但可以持有状态和上下文（跟传统web MVC不同），web前端跟桌面GUI一样具备持久的UI对象，DOM是原生View，View对象是DOM的高层封装

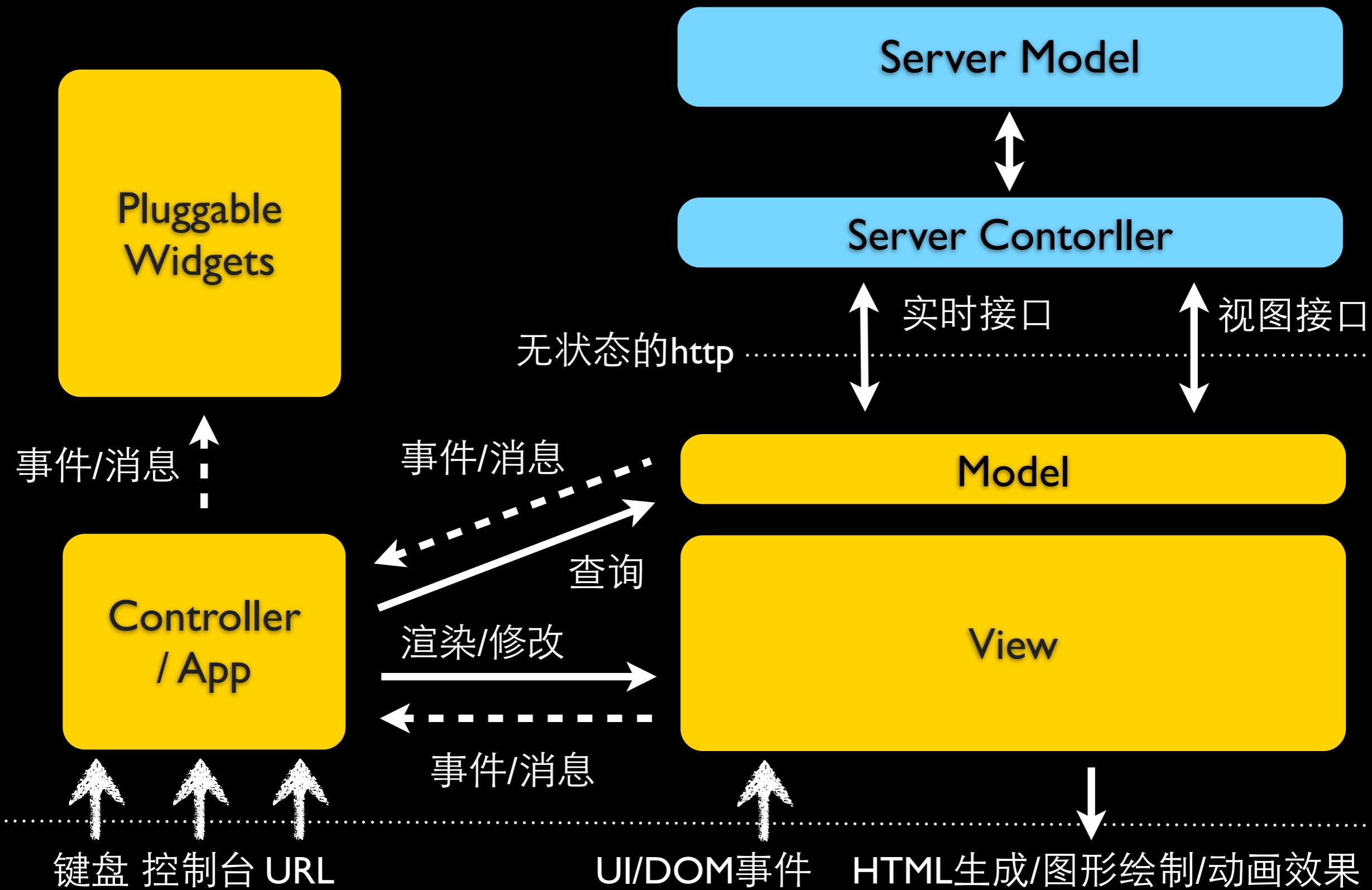
阿尔法城的View层

- 由View层负责处理来自UI的用户输入（而不是Controller），因为：
 - 浏览器UI事件的参数是DOM对象
 - 现代web应用的UI交互伴随大量过渡效果/副作用
 - 实例：进入小店、Google+删除circle
 - 全局事件委托像Cocoa一样连接UI元件和View对象，调用View指令前必须先把DOM状态转换成纯数据参数，保证在View层之外也能直接调用这些方法再现相同的功能

阿尔法城的Controller层

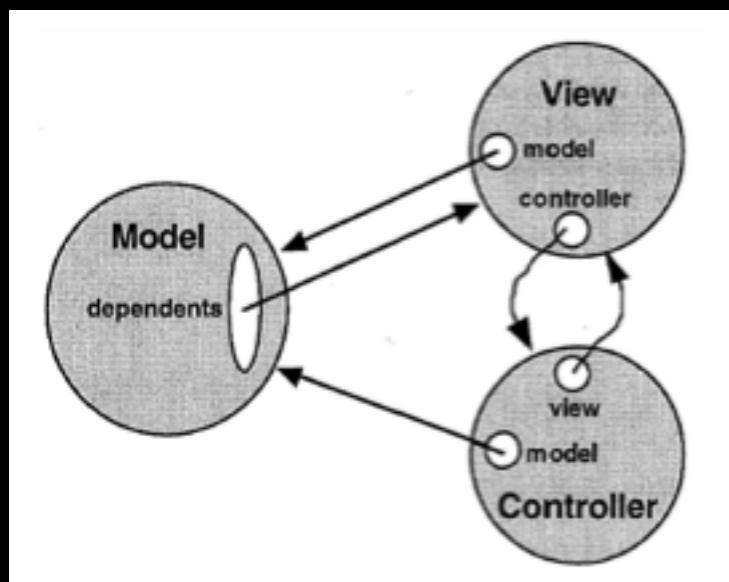
- 负责截获外部输入： 键盘操作， 控制台命令， URL routing， API调用（来自 widget、 自动化测试工具）
- 持有View和Model的引用， 能直接调用它们的方法， 反之只能用事件来通信
- 维护一组简洁、 完整的顶层接口， 以App 对象的形式暴露给外部， 用于复合了多个MVC应用的页面， 或自动化测试

阿尔法城的MVC

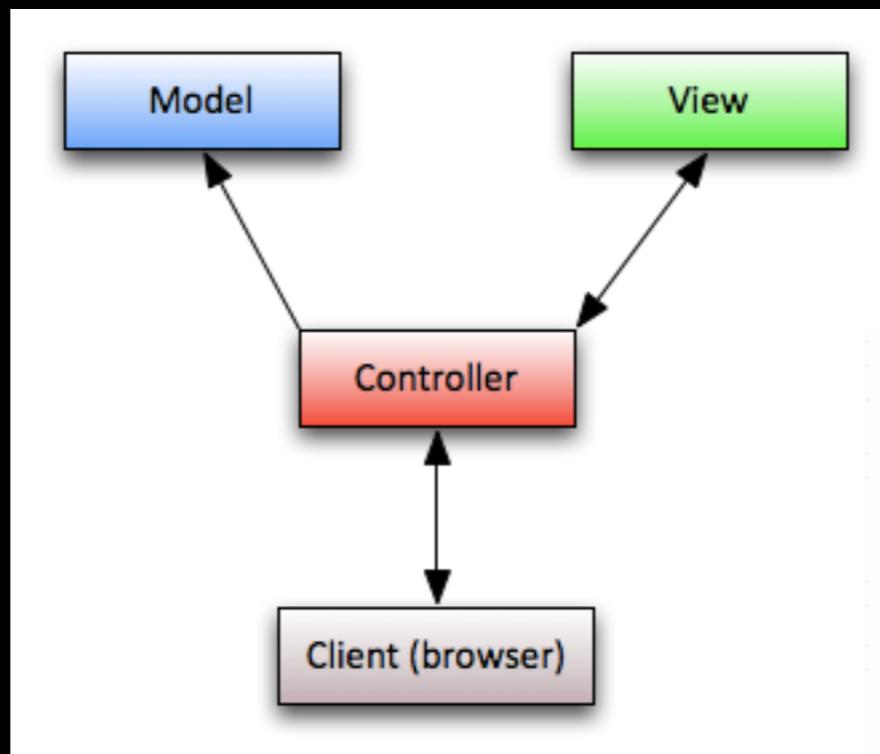


其他的MVC

Smalltalk-80



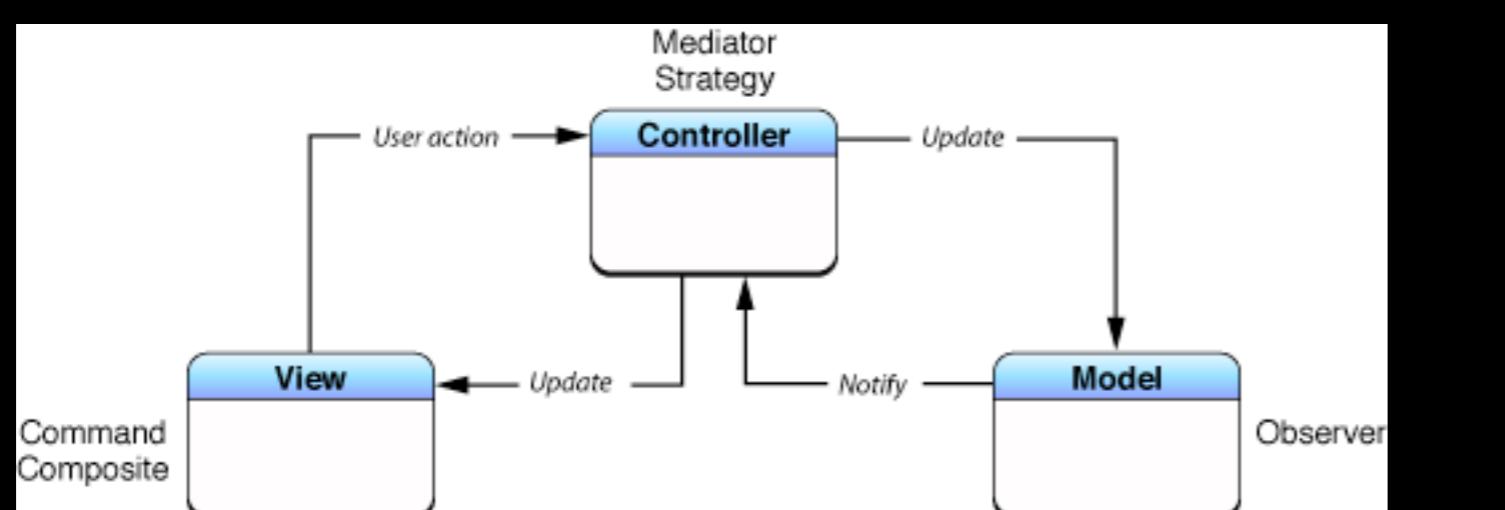
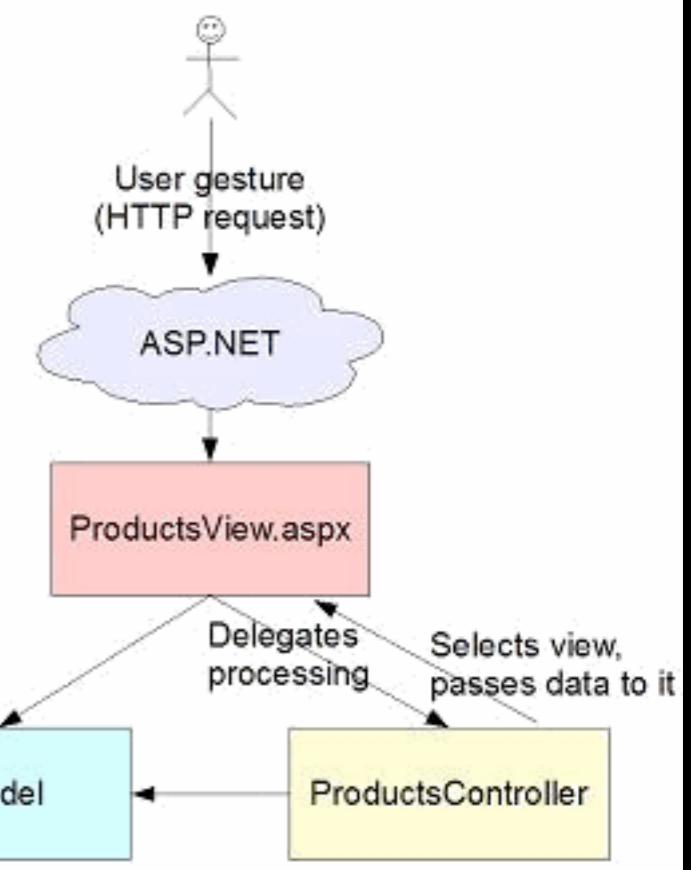
Cocoa



Ruby On Rails

ASP.NET

Model-View-Presenter



We need SMART Models, THIN
Controllers, and DUMB Views



MVC? It's no big deal



MVC? It's no big deal

Thanks

- 我的豆瓣主页：http://www.douban.com/people/Dexter_Yy/
- 我的blog：<http://www.limboy.com>
- 新版阿尔法城：<http://site.douban.com/118836/>（近期上线）
- 可用于MVC应用的module：<https://github.com/dexteryy/OzJS/tree/master/mod>（近期更新）