使用AngularJS构建RIA前端架构

贺天卓

个人介绍

- **Worktile** 前端架构师
- **10**年码农
- 做过设计师、写过后端业务、画过产品原型
- **htz@worktile.com**

小调查

- 前端：?%
- 做RIA富应用的前端：?%
- 用过AngularJS的：?%

议题介绍

- 定义 RIA
- 了解 AngularJS
- 用好 AngularJS
- 更好的 AngularJS
- 一起来 AngularJS
- 其他相关资源

什么是RIA？难在哪？

RIA富客户端

- 富应用Rich Internet Application，具有高度互动性、丰富用户体验以及功能强大的客户端。
- 提供了比HTML更为丰富的界面表现元素，密集、响应速度快和图形丰富的页面元素与数据模型结合在一起，为用户提供好的使用体验。
- Gmail、GoogleDoc；Office365；Worktile……

难在哪里

- 界面：多区域、多模块、多层元素堆叠……
- 交互：拖拽、缩放、快捷键……
- 数据：展现形式、排序规则、合规性检查……

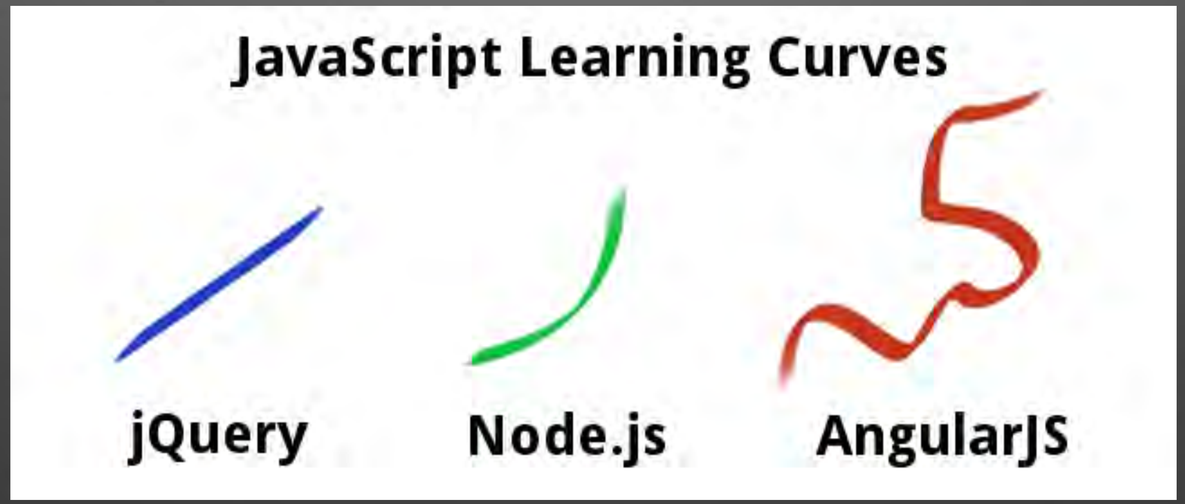
没有Angular之前

Hello Angular

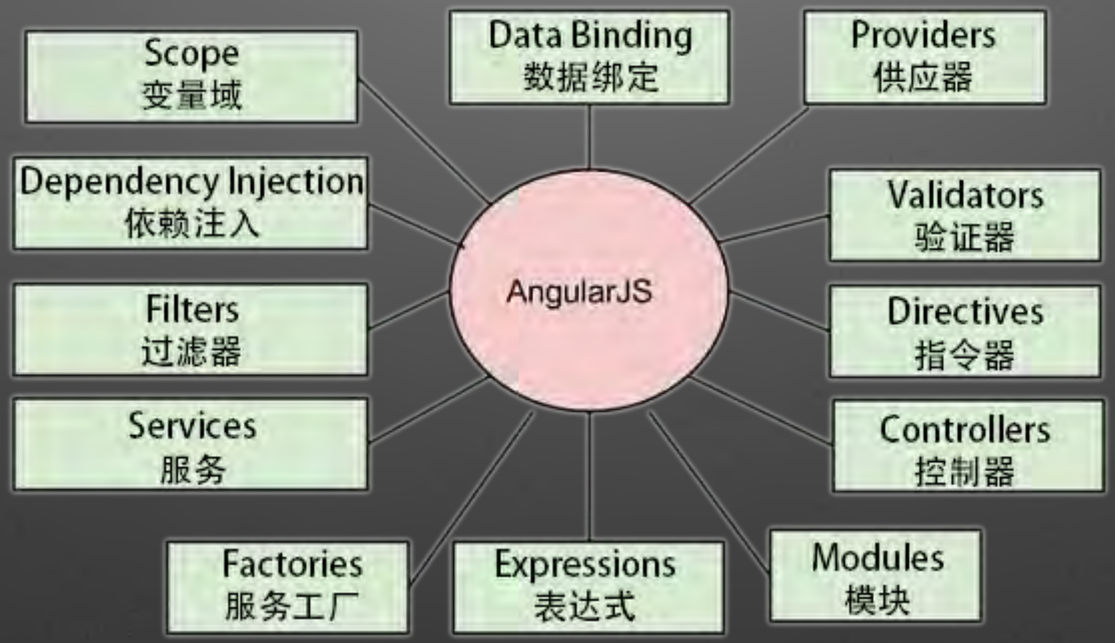
Hello Angular

```
<!DOCTYPE html>
<html ng-app>
<head>
  <script src="./angular@1.5.9/angular.min.js"></script>
</head>
<body>
  <input type="text" ng-model="yourName"/>
  <h1>Hello {{yourName || 'AngulR'}}!</h1>
</body>
</html>
```

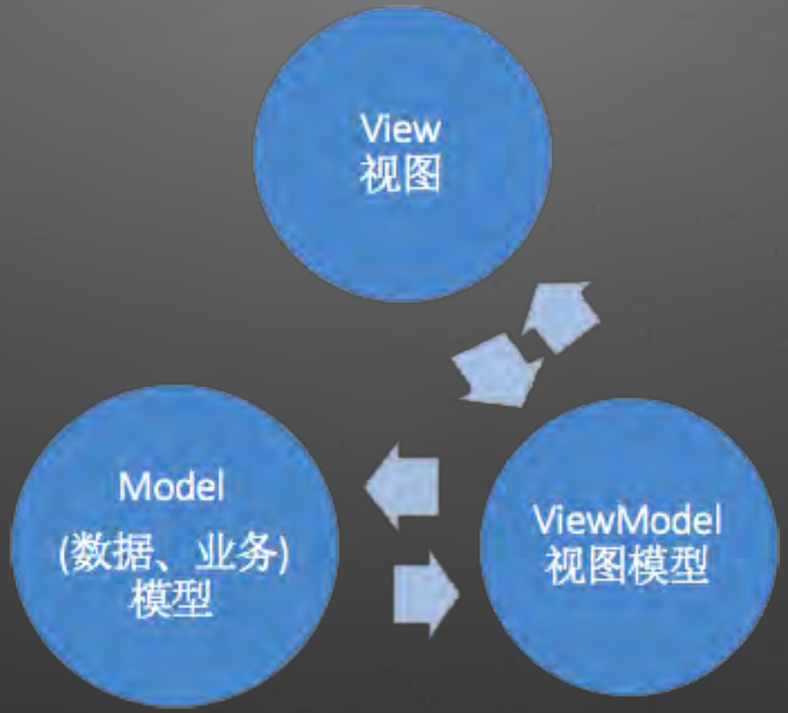
框架学习曲线



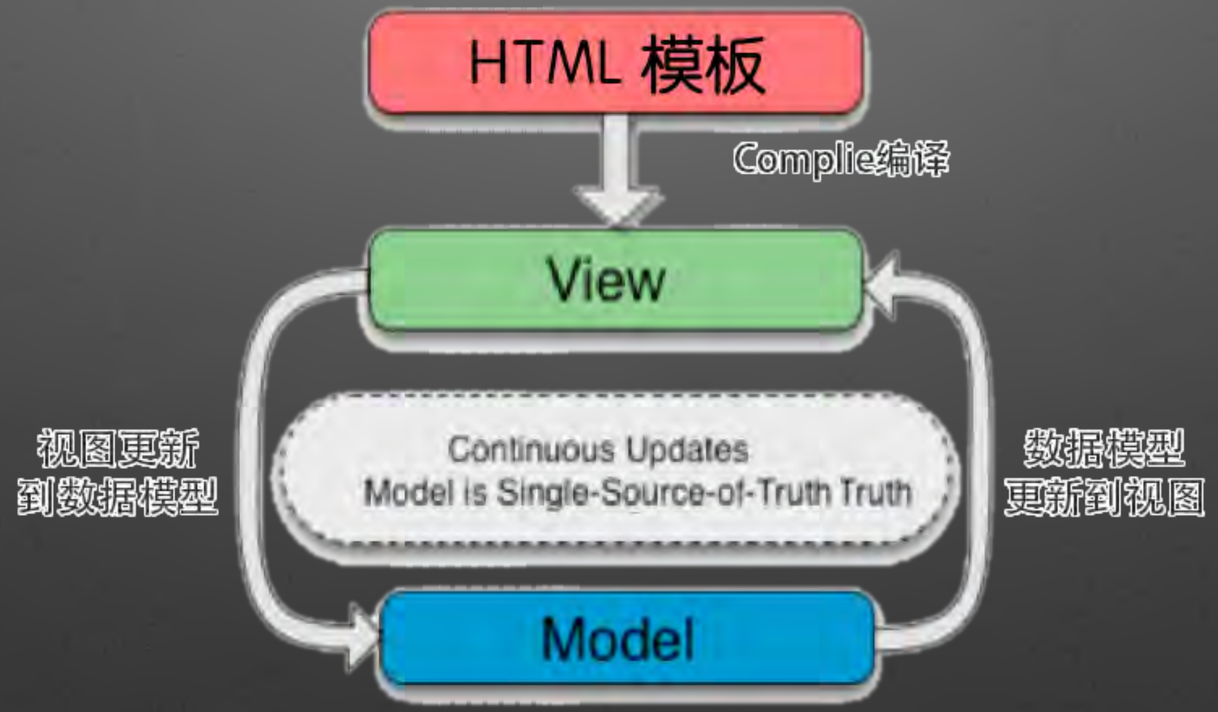
AngularJS 的构成



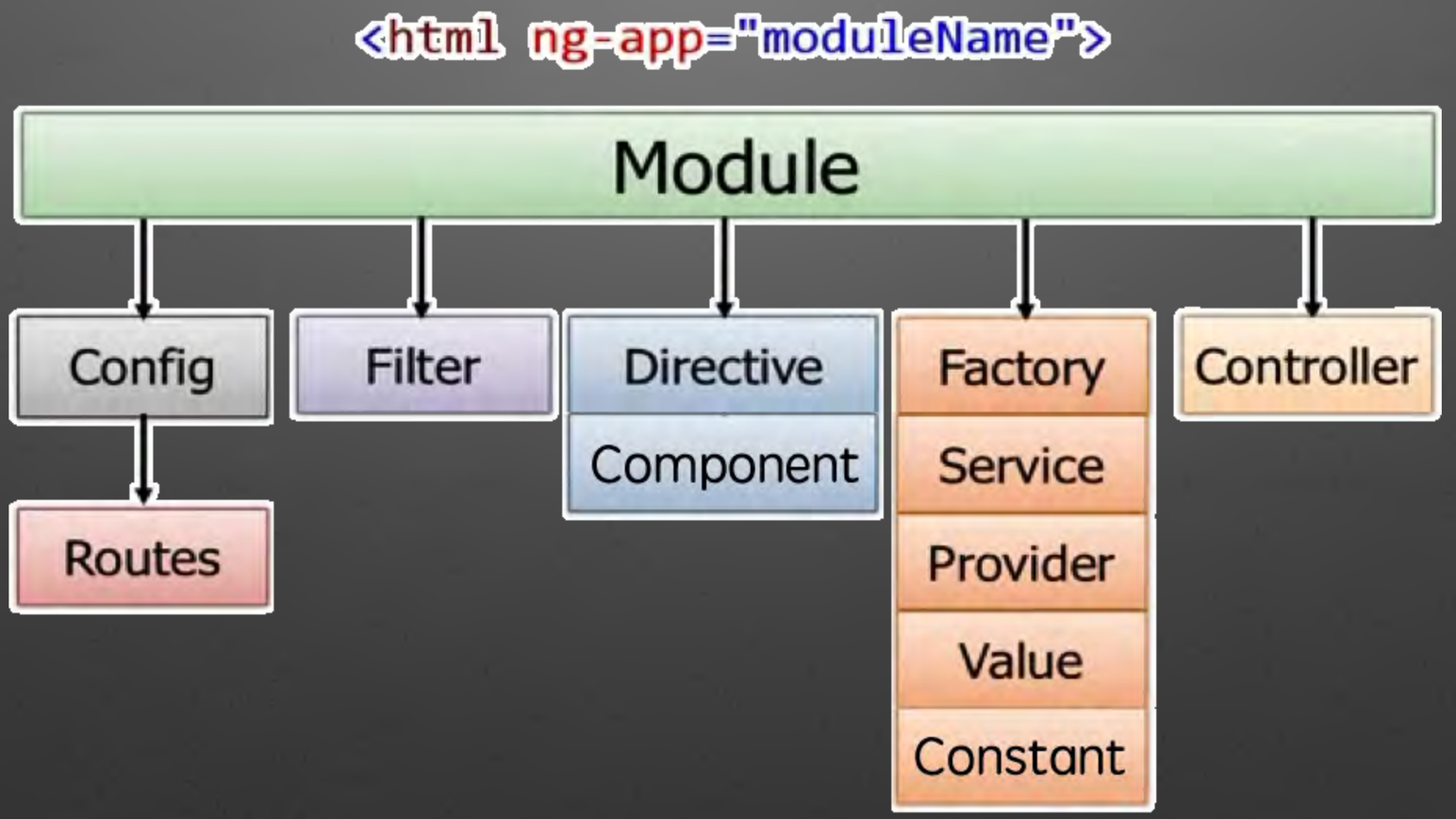
MV*(Whatever)



双向绑定


HTML 模板
Complie编译
View
视图更新
到数据模型
Continuous Updates
Model is Single-Source-of-Truth Truth
数据模型
更新到视图
Model

Modules是容器

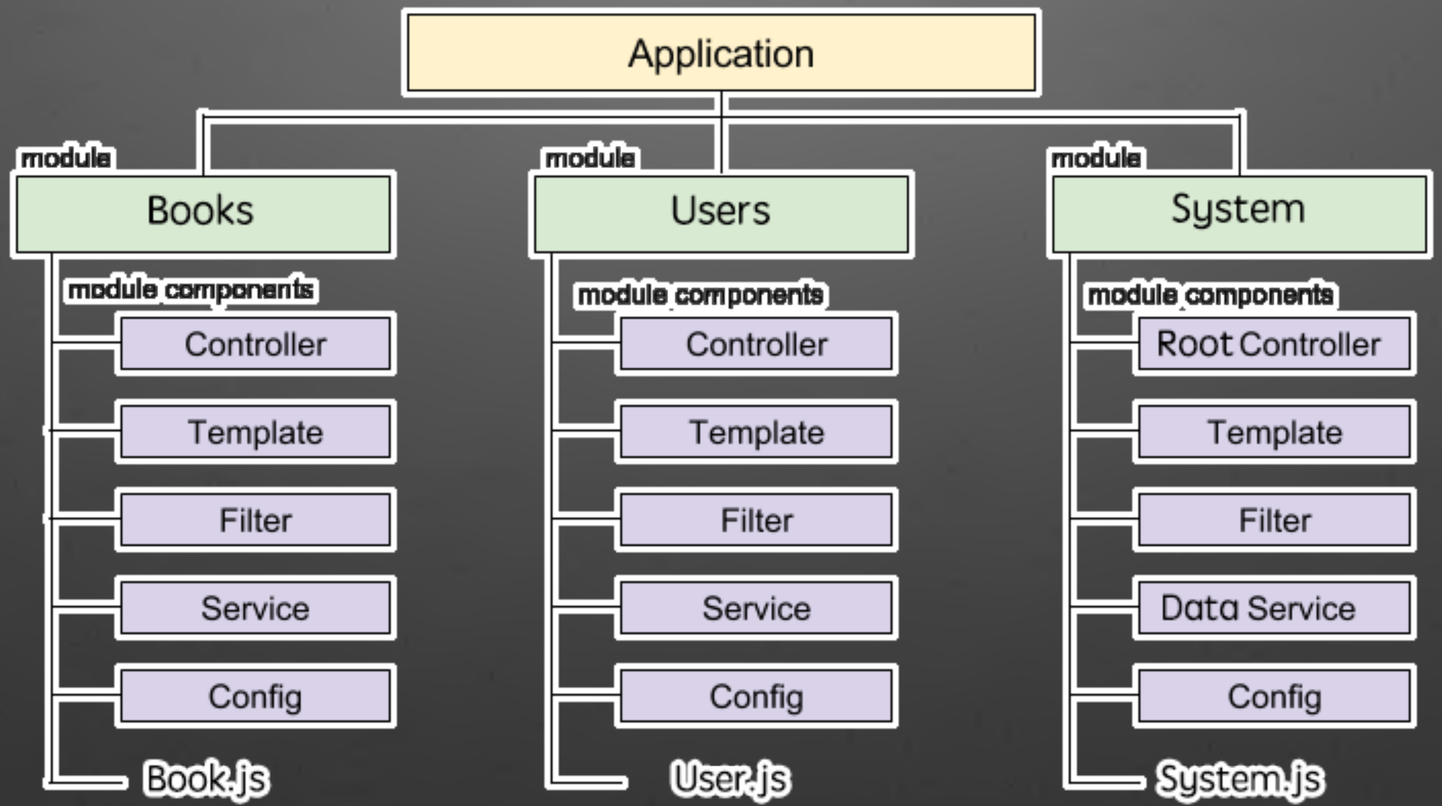

<html ng-app="moduleName">
Module
Config
Routes
Filter
Directive
Component
Factory
Service
Provider
Value
Constant
Controller

用好AngularJS

No jQuery方式

- DOM 不是数据结构
- 数据才是数据结构
- Angular 双向绑定的优势，业务控件基于 业务模型 编程，交互控件基于控件的有限状态机机制编程

Modules分解业务模块



Modules分解业务模块

- angular.module('Book', [])
  .constant('bookConstant',bookConstant)
  .config('bookConfig',bookConfig)
  .config(['$urlRouterProvider',function($urlRouterProvider){}])
  .service('bookDataService',bookDataService)
  .filter('bookStatus',bookStatusFilter)
  .directive('bookItem',bookItemDirective)
  .controller('bookHomeController', bookHomeController)

Module声明和引用

- 声明一个模块

```
angular.module('app', [])
  .controller('homeController', homeController);
```

- 在一个模块上附加代码

```
angular.module('app')
  .controller('indexController', indexController);
```

依赖注入三种声明的方式

- angular.controller(‘homeController’, [function (myService) {
  alert(“inference 推断式”);
}]);

- homeController.$inject = [‘myService’];
function homeController(myService) {
  alert(“annotation 注解方式”);
}

- angular.controller(['myService', function (myService) {
  alert(“inline 内联方式”);
}]);

- 不建议用第一种，不能使用 Uglifyjs进行代码压缩。

文件存放 按模块集中化、水平化

- ROOT
- └ book
- └ book.config.js
  book.service.js
  book.controllers.js
  book.directive.js
  book.item.less
  book.item.html
  book.detail.html

用 ui-router 将 MV* 关联起来

-
  <ui-view></ui-view>

- $stateProvider.state('book', {
  templateUrl: '/book/book.list.html',
  controller : 'bookListCtrl'
  })

Controller控制器只放置场景代码

- 一个控制器只负责一小块视图，不要写出万能控制器。
- 不要考虑重用，当有重用可能，考虑是否放到 **Service** 里。
- 不操作 **DOM**，**DOM** 应该由 **Directive** 驱动。
- 不写 数据模型格式化逻辑， 应该放到 **Service** 和 **Filter** 里。
- 这里没有数据过滤操作， 应该在模板上使用**$filter**服务
- 控制器之间很少互相调用的，但也有 **$emit** 冒泡和 **$broadcast** 广播机制

避免$scope嵌套混乱

- 界面所要用的 在 controller 中放到 $scope.vm 对象中
- 原则上每个 directive 都应该是 isolate scope
- 数据变化采用 cache层引用触发
- 业务变化采用 $emit & $broadcast 广播发射，具体 directive 监听自身关注的消息

Directive中的 link、compile、controller

- compile
- controller
- pre-link
  - (Children loop)
- post-link

Directive中 compile 和 link 的使用时机

- Compile
  - 1. 想在DOM渲染前对它进行变形，并且不需要scope参数
  - 2. 想在所有相同directive里共享某些方法，这时应该定义在compile里，性能会比较好
  - 3. 返回值就是link的function，这时就是共同使用的时候
- Link
  - 1. 对特定的元素注册事件
  - 2. 需要用到scope参数来实现DOM元素的一些行为

用 Directive restrict=E封装元素控件

- <datepicker name='表达式'></datepicker>

- ```
  angular.module('calendar', [])
    .directive('datepicker', datepickerDirective);

  datepickerDirective.$inject = ['momentjs'];
  function datepickerDirective(momentjs) {
    return {
      restrict   : 'E',
      scope      : { datetime: '=' },
      template   : ' {{datetime}} ',
      controller : 'datepickerCtrl'
    };
  }
  ```
  ```

用 Directive restrict=A封装交互组件

- <div datepicker name='表达式'>

- ```
  angular.module('calendar', [])
    .directive('datepicker', datepickerDirective);

  datepickerDirective.$inject = ['momentjs'];
  function datepickerDirective(momentjs) {
    return {
      restrict   : 'E',
      scope      : { datetime: '=' },
      controller : 'datepickerCtrl'
    };
  }
  ```
  ```

总结：分而治之、以序容易

- 路由层：采用 $routeProvider 或者 angular-ui-router 配置
- 视图层：采用 {{表达式}} 绑定数据、$filter 格式化数据显示
- 数据通讯层：$http，angular-resource 封装了Ajax/REST
- 组件：编写 directive 达到交互组件和元素控件重用
- 中间层：编写 service 实现 数据缓存层、业务逻辑层、消息总线层……

更快的 AngularJS

动画也是有开销的

- 局部开启动画：$animateProvider.classNameFilter(/wt-animate/);
- 使用更好的CSS3 translate 3D

DOM 操作是昂贵的

- ng-repeat：指定 track by 主键，复用 DOM 元素。
- ng-If 优于 ng-show
- 考虑在 大量repeat元素父级绑定事件，做事件代理
- directive中跟scope数据无关的操作放在compile阶段，它只执行一次。

双向绑定的困局

- 更小的范围绑定

```
<p>你好，<span ng-bind='vm.text'></span></p>
<p>你好，{{vm.text}}</p>
```

- 一次绑定渲染，一切不需要动态更新的字段，特别是主键

```
{{::bind_once}}
```

浏览器的颤动问题

- _.debounce防颤器：resize、keydown、keyup
- _.throttle节流阀：scroll、 mousemove
- 防止 ng-repeat 重绘重排：先将元素 hide，最后一个元素再show
- 慎用 ng-mouseenter 和 ng-mouseleave

$destory时记得回收资源

- scope.$on('$destory', function() { $timeout.cancel(timer); Socket.removeListener('chatMessage'); });

小步快跑：尽快的执行$digest

- 每次 $digest 过程中，filter 会执行至少两次
- 每次 $digest 过程中，watch 会执行1~2次
- 保证 $watch 和 $filter 没有阻塞，单次执行应该足够快
- 避免 $watch 中操作 DOM，因为它很耗时
- 避免深度 $watch

不要乱动：尽少的触发$digest

- 设定好 directive 的 priority优先级
  将DOM 生成的directive前置
  性能损耗大bind事件和触发$digest cycle的后置

- 不牵涉到数据变更的 $timeout 配置为跳过$digest

前端团队协作

- 可视化的协作管理
  - **DemoCode**：系统中所有的组件和例子代码、前端规范……
  - 看板式的 **API** 文档
  - 流程工具化：代码检查自动化测试

其他资源

- AngularJS官方教学 https://docs.angularjs.org/tutorial
- AngularUI https://angular-ui.github.io
- NgNice控件案例 http://www.ngnice.com/showcase/
- Angular编码规范 https://github.com/johnpapa/angular-styleguide/blob/master/i18n/zh-CN.md

Thank you

Q&A

QQ：2010666
邮箱：htz@worktile.com