exercise代码review及waltz4+dubbo框架知识点考查

考查流程：

1. 个人对练习项目的代码总结。

2、问答，内容主要涉及以下部分内容。

# 前端部分：

1、waltz4的生命周期过程，各过程的生命周期函数的应用场景。

一个标准的Waltz4模块由一个HTML模板，一个JS文件以及一个可选的的样式文件（CSS或者SCSS）组成，HTML作为视图(关注界面展示)，JS作为视图模型(关注逻辑和数据)。如果视图模型有定义以下函数，函数将在模块的生命周期过程中被执行。

Constructor：视图模型的构造函数先执行。

Created：视图模型初始化后，视图以及视图模型都连接到视图控制器。

Bind:通过视图控制器将视图与视图模型绑定。

Attached：虚拟节点与真实节点之间的转化并添加到dom树上。

AfterAttached:当所有模块都被加载完成的后续操作。

Detached:模块在离开的时候会从dom树下脱落。

Unbind:将视图模型与视图解绑。

2、waltz4的组件命名约定及原理。

Aurelia命名约定：自定义元素的视图模型通常与HTML“View”配对。在你的视图模型的名称后追加CustomElement或CustomAttribute来告诉框架某个类是自定义元素或自定义属性的视图模型。

JavaScript类的标准命名约定是InitCaps，意思是每个字母首字母大写; 然而，HTML是不区分大小写的，通常归一化为小写。为了缩小这个差距，Aurelia将采取类名称，去掉它CustomElement或CustomAttribute后缀，然后将类名转换为dash-case。

Aurelia也提供了覆盖命名约定的方式：通过在你的HTML行为的视图模型中使用@customAttribute或@customElement装饰器来明确命名你的HTML行为。你传递一个字符串到装饰器，这成为你的行为将使用的名称。Aurelia不会以任何方式改变这个字符串。

3、waltz规范的自定义事件。

绑定系统支持绑定DOM事件。一个DOM事件绑定将在DOM事件被触发时执行一段JS表达式。事件绑定声明有三个部分：

event.command="expression":

event: DOM事件名称，不带“on”前缀

command:Aurelia事件绑定命令:

trigger（引发，触发）: 将事件处理直接连接到元素上。当事件被触发时，处理函数将被执行。(load,unload,focus,blur等事件用trigger).在dom树的分支，会引起冒泡。

delegate（委托，代表）: 将一个单独的事件处理程序附加到Document上，当事件处于冒泡时期时，由Document回调合适的关联表达式。在dom树的顶点，不会引起冒泡。

capture: 将一个单独的事件处理程序附加到Doucument上，当事件处于捕捉时期时，由Document回调合适的关联表达式。

expression: 一个js表达式。使用$event属性去连接绑定表达式中的 DOM事件。

注意：事件委托是一种是为了减少事件订阅的策略。只有可以冒泡的事件才可以使用aurelia的delegate绑定。

如果你需要含参调用某个方法，需要创建一个对象，键是参数名，值是参数值，然后使用参数对象进行调用。参数对象的属性在call绑定表达式中将可以被数据绑定。

前端中call反射，对应着后台的invoke反射。

onsearch.call="onSearch($event)"

onreset.call="onReset($event)

click.delegate="onCreate()

4、如何自定义组件，slot使用方式。

自定义组件一般是一个html文件和一个js文件配套使用，有时候需添加一个scss或css样式以及validations校验规则。组件命名需遵循组件命名约定等。

Slot：插槽，也就是槽，是组件的一块HTML模板，这块模板显示不显示，以及怎样显示由父组件来决定。

5、waltz4提供的路由类的使用方式（如何跳转传参，回退）。

在frame-nav.js以及app.js等文件中预先定义好路由路径。在视图模型中引入aurelia-frame架构的路由类Router。在视图中定义好方法以及传入参数，在视图模型中使用Route类的方法进行跳转。

**this**.router.navigate('orderSource');

**this**.router.navigate('orderSource/create');

**this**.router.navigateBack();

**this**.router.navigateToRoute('orderSourceView', {  
 id: orderSource.id  
});

6、wz-form表单校验的底层原理，要求能够梳理对应源码，包括自定义表单校验规则。

wz-form表单校验规则的使用，通常需要预先在视图的form中绑定validations,并声明引用的form名称。建立视图对应名称的校验文件，在其中编写对应的校验规则。在视图模型中引入校验类，进行校验。

7、fetch拦截器做了什么，在常规endpoint无法适配的情况下，如何进行fetch请求（如文件，图片传输），fetch异常捕获。

fetch拦截器（interceptors）一般用于发起http请求之前或之后对请求进行统一的处理，如token实现的登录鉴权（每个请求带上token），统一处理404响应等等.

8、插值语法与绑定语法的差异，值转换器的用途。

默认情况下，插值绑定的模式是to-view，表达式的结果总是被强制字符串。null或未定义的结果将导致空字符串。插值更易读，但是如果你需要双向绑定一个可编辑的元素。还是需要bind。绑定行为和值转换器之间的主要区别是绑定行为在整个生命周期中都可以完全访问绑定实例。

Aurelia的一个简单约定：以ValueConverter结尾的导出名称被假定为值转换器。 该协议使用导出名称来注册转换器，驼峰式，从最后剥离ValueConverter部分。

9、show和if的差别。

if：判断是否渲染，减轻服务器的压力，在需要时加载；

show：调整css 的display属性，可以使客户操作更加流畅

10、模态框如何实现。

在视图模型中引入waltz-webframe中的dialog类，在需要时使用即可。

*/\*\* type类型 可选值'alert', 'confirm', 'tooltip', 'loading' \*/  
/\*\* state状态 可选值'success', 'info', 'question', 'error', 'loading' \*/*

常见用例如下：

**this**.dialog.toolTip(error.message, 'error');

**this**.dialog.alert('请至少选择一条记录！');

**this**.dialog.confirm('确定取消？').then(closeResult => {  
 **if** (!closeResult.wasCancelled) {//点击确定  
 **this**.router.navigateBack();  
 }  
});

11、@mixin的用途。

mixin模式就是一些提供能够被一个或者一组子类简单继承功能的类,意在重用其功能。在面向对象的语言中，我们会通过接口继承的方式来实现功能的复用。但是在javascript中，我们没办法通过接口继承的方式，但是我们可以通过javascript特有的原型链属性，将功能引用复制到原型链上，达到功能的注入。

12、事件冒泡（wz-button当设置disabled属性时，如何阻止他的事件依然被触发）。

事件冒泡机制可以在它的子标签上的js代码末尾中加上这一行进行阻止。

event.stopPropagation();

13、aurelia中，计算属性的使用以及他与observable，bindable注解所标注的属性的区别。

首先需在在视图模型中引入aurelia-framework中的computedForm类，常用于批处理过程中计算。

@observable只跟踪属性值的变化，而不是值本身。这意味着，如果属性是数组，当数组添加，删除，编辑items时，值改变处理器不会被调用。

@bindable装饰器用视图模型的一个属性上时，Aurelia将允许使用Aurelia提供的所有绑定命令（bind，two-way，one-way，one-time，等）来通过HTML绑定到该属性

14.前端中js方法的for in 与 for of的区别；

For in 遍历的是数组，对象的属性，获取的是下标。

For of 遍历的是数组的值，对象的值。

15.前端js中，===与==的区别。

==是一般比较，比较的是值。

===是严格比较，不光比较值同时比较数据类型。

16.前端中class样式与style样式的区别。

style是HTML中直接定义样式，class可用于引用定义好的css类(class)，实现重用等。

另外，在JS中，style就是用style关键字来使用

document.getElementById("div1").style

class要使用className关键字来使用

上例中是document.getElementById("div2").className

17.js中typeof与instanceof的用法。

typeof用于判断一个数据的类型.

number,boolean,string,function,undefined,object

instanceof用于判断一个数据是否是某个对象的实例化。

# 后端部分：

1.dubbo服务注册xml文档解释，本地调试方式。

提供方，消费方，注册中心，运行中心，监控中心

服务容器负责启动,加载,运行服务提供者。

服务提供者在启动时,向注册中心注册自己提供的服务。

服务消费者在启动时,向注册中心订阅自己所需的服务。

注册中心返回服务提供者地址列表给消费者,如果有变更,注册中心将基于长连接推送变更数据给消费者。

服务消费者,从提供者地址列表中,基于软负载均衡算法,选一台提供者进行调用,如果调用失败,再选另一台调用。

服务消费者和提供者,在内存中累计调用次数和调用时间,定时每分钟发送一次统计数据到监控中心

2.当前框架如何实现job，服务启动过程主要经历了哪些流程。

提供者服务端启动过程：

1. 初始化动态配置；
2. 初始化缓存配置；
3. 初始化API权限；
4. 初始化搜索客户端；
5. 启动监视器；
6. 启动负载均衡器；
7. 初始化OSS配置；

消费者服务端启动过程：

（1）初始化动态配置；

（2）初始化缓存配置；

（3）初始化OSS配置；

3.es索引查询的使用方式。

已安装ElasticSearch工具。

4.框架如何实现按租户分库。

在global数据库中的cf\_config中配置相关信息，依据承租人id和当前应用名称配置数据源以及数据库名称等信息。

5.EventHandler的处理流程，及框架事务的实现。

框架dao层操作采用aop编程，有消费服务访问需要请求数据库时都先经过DataSourceAspect切面，获取当前分库。

框架事务加在dao层，若没有事务则新建事务，若发生异常则回滚。

@Transactional(value = "pluto-core.txManager", propagation = Propagation.*REQUIRED*, rollbackFor = Exception.class)

6.框架对于jpa懒加载的操作。

@OneToMany(cascade = CascadeType.*ALL*, fetch = FetchType.*LAZY*, mappedBy = "tran")

使用懒加载来调用关联数据，必须保证主查询session的没有被关闭。

7.框架封装的查询语句如何使用（主要是query的使用）。

EntityQuery q = new EntityQuery(name).select("o").from(PTaxPayer.class.getName(), "o")  
 .where("o.tenantId = :tenantId").p("tenantId", tenantId);

Query query = em.createQuery("from " + PTran.class.getName()  
 + " o left join fetch o.details where o.tenantId = :tenantId and o.xid = :xid");

8.如何实现按租户配置，如何添加新配置。

在global数据库中的cf\_config中配置相关信息，依据承租人id和当前应用名称配置数据源以及数据库名称等信息。

9.框架对redis缓存及oss的使用。

系统初始化服务时，从配置库中取出相应的配置信息。

启用了一个缓存工厂和一个oss媒体客户机。

OSS是一个综合的业务运营和管理平台。

10.api调用权限的实现。

在数据库对商户的权限进行相应配置。保存在数据库中。前端路由刷新时访问后台ac\_permission表中存储的权限信息。获取权限后保存localStorage本地存储中，后续使用即从该localStorage中取出进行使用。

11.版本升级如何升级数据库表（新建表，插入表字段，修改索引）

一些不影响之前表间关系的数据库操作。

# 前后端交互：

1.跨域的概念及当前框架跨域的实现。

**跨域**由于浏览器同源策略，凡是发送请求url的协议、域名、端口三者之间任意一与当前页面地址不同即为跨域。**当前框架通过配置跨域调用白名单实现跨域功能。**

2.单点登录的实现。

sso需要一个独立的认证中心，只有认证中心能接受用户的用户名密码等安全信息，其他系统不提供登录入口，只接受认证中心的间接授权。间接授权通过令牌实现，sso认证中心验证用户的用户名密码没问题，创建授权令牌，在接下来的跳转过程中，授权令牌作为参数发送给各个子系统，子系统拿到令牌，即得到了授权，可以借此创建局部会话，局部会话登录方式与单系统的登录方式相同。

本系统主要通过设置TokenFilter过滤器实现，系统在首次登陆时往cookie中放置一个特定的token签名。后续每次访问请求均需经过该过滤器，若未登陆或token已过期则重定向到登陆页面。

3.traceid的实现。

使用一个traceId跟踪请求的全部路径.

当发起请求来时生成一个traceId放在request请求头里，定义trace过滤器,当请求到来时，若请求头存在trace则将其放入traceContext，并打印出其请求的URL。

4.前后端数据包装，异常包装的方式。

数据包装，依照数据库中定义好的列名来。符合标准，见文知意即可

继承运行时异常。打印出相应的错误信息。

5.如何实现token颁发及过期强制前端跳转登录。

登陆时使用JWT生成token,将其存储在cookie中。定义一个token过滤器，请求来时判断token是否已失效，若失效则重定向到登陆页面，请用户重新登陆。

工具：

1、mkdown语法（掌握如何使用）

markDown语法掌握。