engine - 执行引擎

管理服务注册、流程执行、上下文信息等，是服务请求的实际执行层。每个服务器都会创建一个与其它服务器隔离的执行引擎，用于接管当前服务器的所有服务执行。引擎支持安全关闭与重启。

一、服务管理

1. 注册与注销

根据服务名、服务函数或对象来注册服务。服务名可使用通配参数，格式 `/order/request` 或 `/order/:type` 。其中 `mqc` ，`cron` 支持运行期间动态注册与注销服务，其它服务器只能在启动前注册服务。

2. 对象注册

注册的对象中包含有 `Handle` 结尾的函数，只要符合签名规则，都会注册为服务。实际服务名为：注册服务名+函数名去掉"handle"。

当函数名为 `GetHandle` ，`PostHandle` ，`PutHandle` ，`DeleteHandle` 则服务名不变，HTTP请求必须是对应的 `get` ，`post` ，`put` ，`delete` 才会被执行。此情况可用于RESTful服务注册。

当函数名为 `Handle` 也会注册为服务。

3. 函数签名

系统只支持一种格式的签名函数 func(ctx *hydra.Context)interface{}

ctx中包含有请求参数，环境参数，引擎参数，服务请求对象(request), 响应对象(response), 路由参数，日志对象，引擎组件等。

函数返回值为interface{}支持所有类型，如：struct,map,string,int,float,error.其中 `error` 为执行出错，其它都为执行成功。根据服务器的类型不同或明确指定的响应的格式，返回的结果，将被格式为 `json` ，`xml` 或翻译为指定的响应格式。

4. 函数检查

引擎初时化时会对注册的服务函数进行类型检查，有一个函数签名不符合要求的都会导致引擎创建失败。

二、执行流程

1. 引擎创建

引擎依赖"注册中心","配置中心","服务列表"，并初始化"注册服务",加载"基础组件"，包装"配置与注册中心"等。其中"初始化服务"的"函数检查"可能会导致引擎初始化失败。

2. 流程执行

引擎提供一个线程安全的业务执行函数 `handle` 。函数将创建“执行上下文" `context` ，并将执行期分为三步：`执行前`，`执行`，`执行后` 分别对应 `handling`，`handle`，`handled` 可通过勾子函数，在请求执行前与执行后执行业务处理。

3. 勾子函数

系统提供两个勾子函数：

1. handling,执行前。每个请求执行前执行，用于对请求进行预处理，例如：登录检查。
2. handled,执行后。每个请求执行后执行，用于对请求的资源回收处理，或后续业务处理。

三、基础组件

提供数据库、缓存、消息队列、远程调用的基础组件。可通过注册中心配置运行参数，使用时直接通过引擎提供的接口创建。

1. 生命周期

由引擎提供的接口进行组件创建，引擎销毁前首先关闭所有已创建的组件。

2. 组件版本

每个组件都有版本号，即配置信息版本号。当配置发生变更后，重新获取的组件将用最新的配置进行创建。原组件将在下一次引擎关闭前进行清理。

3. 组件隔离

组件依赖于引擎创建，每个引擎创建的组件都是全新的。