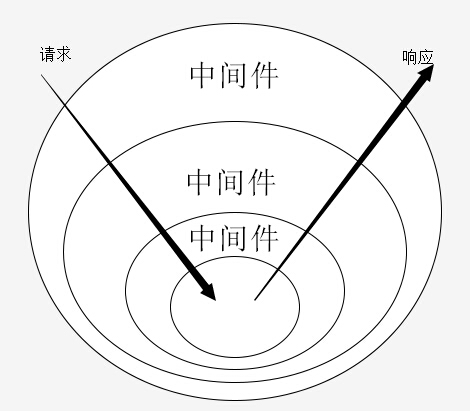
* 
* koa 最有特点的就是 U 型 middleware 的实现了，有点像面向切面编程 AOP 从上图可以看出，它顺序执行 middleware，到达最后一个时，又按队列再次顺序执行。
* 代码上理解就是这样子的：

|  |  |
| --- | --- |
| * 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10 * 11 * 12 | * .middleware1 { * // (1) do some stuff * .middleware2 { * // (2) do some other stuff * .middleware3 { * // (3) NO more middleware ! * // ctx.body = 'hello world' * } * // (4) do some other stuff later * } * // (5) do some stuff lastest and return * } |

* **Middleware 能做的事**
* 在请求到达前，做用户和权限的判断，做 RateLimit，限制 某个可用的 IP …
* 在请求处理后，可以记录服务器处理时长，对结果统一格式化 ….
* **Express 的 Middleware 是单向的，就没有它这么方便。**
* 1. Yield next 与Yield\* next的区别：
  + yield\* 是委托提取器。
  + 简单地说，yield 是你给什么它提取什么，但是 yield\* 会继续向下请求，直到没的提取为止。
* 1.问题描述：
  + 线下人工充值时，
    - 用户无权限，但是仍然正常充值了。
    - 充值以后，进行了页面重定向（即302）
* 2. 原因分析
  + 由于用户无权限，所以页面重定向是对的，但不应该正常充值。
  + 给人的感觉是先进行了接口调用，后进行的权限校验
* 3. 解决方案
  + 权限处理后未彻底退出，仍继续往下运行了
  + 已区分接口和UI，对应不同的处理方式