UniqueService

我们可以通过 skynet.newservice 启动一个 lua 编写的服务。同一段脚本可以启动多份，每个有不同的地址。**地址是区分不同服务的唯一标识。**

但有时，整个系统中解决一类事务只需要一个服务，在系统启动时，它便启动好，而其它服务需要知道它的地址以便于使用它。这个时候，使用 skynet.uniqueservice 是更好的选择。**即 UniqueService 用于启动唯一服务**

skynet.uniqueservice 和 skynet.newservice 的输入参数相同，都可以以一个脚本名称找到一段 lua 脚本并启动它，返回这个服务的地址。但和 newservice 不同，每个名字的脚本在同一个 skynet 节点只会启动一次。如果已有同名服务启动或启动中，后调用的人获得的是前一次启动的服务的地址。

它很大程度上取代了具名服务（不再推荐使用的早期特性）的功能。很多 skynet 库都附带有一个独立服务，你可以在库的初始化时，写上类似的语句：

```
local SERVICE

skynet.init(function()

  SERVICE = skynet.uniqueservice "foobar"end)
```

这个范例会注册一个初始化函数去初始化 SERVICE 变量。而你的库函数就可以使用 SERVICE 这个地址来访问对应的唯一的 foobar 服务了。

uniqueservice 采用的是惰性初始化的策略。整个系统中第一次调用时，服务才会被启动起来。有时，你并不希望做惰性初始化，而在 skynet 启动脚本里明确把必须的服务初始化好（这个初始化过程可能比较漫长，惰性初始化会导致不必要的运行时延迟）。那么，如果你明确知道服务已经启动好，可以使用 skynet.queryservice 来查询已有服务。如果这个服务不存在，这个 api 会一直阻塞到它启动好为止。

uniquesevice 采用惰性初始化策略，第一次调用服务才启动。

阻塞 api queryservice 查询已有服务。

默认情况下，uniqueservice 是不跨节点的。也就是说，不同节点上调用 uniqueservice 即使服务脚本名相同，服务也会独立启动起来。如果你需要整个网络有唯一的服务，那么可以在调用 uniqueservice 的参数前加一个 true ，表示这是一个全局服务。**即 uniqueservice 不跨节点，需要独立设置为全局服务。**

对应的，查询服务 queryservice 也支持第一个参数为 true 的情况。这种全局服务，queryservice 更加有用。往往你需要明确知道一个全局服务部署在哪个节点上，以便于合理的架构。你可以在你设计的节点上的启动脚本中调

用 skynet.uniqueservice(true, "foobar") 将服务启动后，然后再在其它使用它的地方调用 skynet.queryservice(true, "foobar") 。

和 DataCenter 不同，uniqueservice 是一个专用于服务管理的模块。它在服务地址管理上做了特别的优化。因为对于同一个名字，只允许启动一次，且不准更换。所以，在实现上，我们可以在每个节点缓存查询过的结果，而不必每次都去中心节点查询。

uniqueservice 在每个节点上缓存查询过结果，可以不必每次都去中心节点查询。