Vapula Envelope设计

Envelope是Method的参数信封

当Library被加载后，每个Method会创建一个Envelope原型

当需要调用Method时，

Invoker的构造和初始化过程中，会复制一个Envelope

并将该实用副本托管于Invoker的Stack中

Envelope用于Method的用户参数管理

Vapula通过堆（和内存池）管理参数数据

每个Variable包含参数类型、方向（输入、输出、输入或输出）、数据长度

Vapula的Envelope支持所有数据类型。

但是受制于不同语言、运行时对数据类型的定义的差别

Vapula做出以下限制：

除Object和String外，其他类型为基值类型

String是Object的一种特殊细分，作为非基值类型没有本质区别，只是可以使用更简单的API进行操作。

基值类型变量的数据作为一个数组存储

非基值类型变量的数据只能存储为一个独立个体，不能存储为数组

Vapula会识别写入和读取的类型是否是基值类型

根据判定结果，Vapula确定是否可以调用访问方法

同时，Vapula不关心数据的具体存储类型

对于基值类型，Vapula提供一些语法糖（模板函数），

可进行快速单值读取和写入，目的仅仅是简化操作

再次重申，Vapula不检测访问基值类型时使用的数据类型是否匹配

写入字符串时会自动复制并存储副本

投递操作：

投递是Vapula定义的一种行为，表示参数从一个信封的一个变量传递到另一个信封的另一个变量。用Deliver方法描述。

同型间投递通过精确数据传递实现，要求类型标识完全一致。

由于Vapula不关心非基值类型的实际类型信息，所以object类型之间虽然执行了精确数据复制，但是不能保证这个操作是有意义的。

不同类型间投递将类型分为3类：自由对象、字符串和基值。

转换通过字符串作为中介。

由于涉及复杂内存变换，所以内存操作策略由用户指定。

内存操作策略分为：引用传递、内存压缩编码、硬拷贝

默认采用引用传递，不推荐使用硬拷贝。

引用传递：

保证字符串、基值（将只有一个值）的字面量都是引用

内存压缩编码：

将值得对应内存压缩后，通过Base64编码

字符串存储Base64串

基值不接受这种方式，将用内存拷贝替代

硬拷贝

将内存块直接复制

一般获得Envelope后：

设置值：

Variable\* var = env -> GetVariable(1);

var -> Write<int>(12);

var -> Write (data, 1024, true);

var -> Write(str, true); //非基值类型可用