Vapula用户数据托管方案

Vapula通过信封（Envelope）和变量（Variable）实现用户数据托管。

信封的本质是变量的有序集合，其中提供变量的检索和动态增减功能。

信封的生成有两种途径，一种是在组件操作中生成，一种是用户自行生成。

当组件加载时，会为每个方法构造原型信封。

当用户调用组件方法时，调用器会复制原型信封，并托管在虚拟栈。

用户也可以自行编写符合规范的XML，构造用户自定义信封。

Vapula通过堆（和内存池）管理信封中托管的变量的数据。

变量是参数的数据实体，由信封托管。

每个变量包含数据类型、方向限定、数据长度等信息。

Vapula支持多种数据类型，数据长度根据数据类型不同有不同意义。

变量的数据可以是数组或单个实例，根据数据类型也有不同限制。

Vapula的方向限定有3种：仅输入、仅输出、输入或输出。

（表格）

通过这些基本类型，Vapula理论上可以操作任意数据类型。

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，作为非基值类型与object没有本质区别，

只是可以使用一些更具体的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，表示参数从一个变量传递到另一个变量，可以通过Deliver方法调用。

在Variable之间进行Deliver涉及潜在的内存复制，

不复制内存仅会传递数据的访问权，通过复制操作可以复制数据的拥有权。

Vapula不关心类型的实际数据构成，所以类型之间虽然执行了数据传递，

但是不能保证这个操作是有实际意义的。

为了提供更多灵活性，

Variable还提供地址操作、序列化操作

地址操作包括内存取地址和地址还原。

序列化操作提供内存压缩编码支持。

通过内存压缩编码可以将任意数据块压缩编码成base64，方便数据传输

地址操作通过Read、Write即可实现

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