注册过程：

利用TypeInfo注册，注册过程中会新建TypeImpl结构，并依次赋值TypeImpl结构体字段，最后注册到全局的hash表中，键为qom类型名称（TypeInfo的name字段，例如：virtio-blk-pci），值就为TypeImpl结构体指针。

(1)类型初始化：由于对象使用懒加载模式，只有第一次初始化该类对象或者子类对象时，才会对该类型进行初始化。类型初始化时，调用class\_init函数完成类型内部的初始化。

例如：

virtio-blk-pci设备类型初始化顺序：

父-->子

Object没有class\_init(NULL) ->

DeviceClass的device\_class\_init->

【注册该类型的realize和unrealize函数，支持热插拔】

PCIDeviceClass的pci\_device\_class\_init->

【覆盖注册父类以及注册本类型的realize和unrealize函数，注册总线类型和设备属性】

VirtioPCIClass的virtio\_pci\_class\_init->

【覆盖注册父类以及本类型的realize函数，注册设备属性和厂商id等】

VirtioPCIClass的virtio\_blk\_pci\_class\_init（这里类型与父类一致，但覆写了class\_init）

【覆盖注册父类属性和realize函数，注册厂商和设备id】

(2)对象实例化：当类型递归初始化完毕之后，可以实例化非abstract的对象。对象的实例化也是一个递归过程，这是因为每一个子类类型都包含了父类的结构体，那么在实例化子类对象的时候，就需要从父类结构体字段到子类结构体字段以此初始化，从而完成对象的实例化。实例化的过程就是递归调用instance\_init的过程。

例如：

virtio-blk-pci设备对象的实例化过程：

父-->子

Object的object\_instance\_init->

【注册Object对象type属性的‘get'函数】

DeviceState的device\_initfn->

【注册对象的realized、hotpluggable、hotplugged属性的get和set函数】

PCIDevice没有instance\_init(NULL)->

VirtIOPCIProxy没有instance\_init(NULL)->

VirtIOBlkPCI的virtio\_blk\_pci\_instance\_init

【这里比较重要：1、实例化了virtio-blk-pci内部的virtio-blk设备对象（这个过程会递归调用virtio-blk类型的初始化和virtio-blk对象的实例化）。2、添加了virtio-blk-pci对象内部的iothread和bootindex属性】

**对于其中1中virtio-blk设备的实例化步骤如下：**

**（1）virtio-blk设备类型初始化:父->子，红色部分在vitio-blk-pci设备类型的初始化中已经初始化过了，这里不会再执行。**

Object没有class\_init(NULL) ->

DeviceClass的device\_class\_init->

【注册该类型的realize和unrealize函数，支持热插拔】

VirtioDeviceClass的virtio\_device\_class\_init->

【覆盖注册父类的realize和unrealize函数，覆盖注册父类的总线类型和设备属性】

VirtioDeviceClass的virtio\_blk\_class\_init（这里类型与父类一致，覆写了class\_init）

【注册类型的realize、unrealize、get\_config、set\_config、get\_feature、set\_feature、set\_status、save、load等函数】

**（2）virtio-blk设备对象的实例化：**

Object的object\_instance\_init->

【注册Object对象type属性的‘get'函数】

DeviceState的device\_initfn->

【注册对象的realized、hotpluggable、hotplugged属性的get和set函数】

VirtIODevice没有instance\_init（NULL）->

VirtIOBlock的virtio\_blk\_instance\_init

【添加了virtio-blk对象的iothreadhe bootindex属性】