中介者模式 - 世界政府

发表于 2022-09-16 | 更新于 2023-04-06 | 设计模式

| 字数总计: 2.7k | 阅读时长: 9 分钟 | 阅读量: 407 | 评论数: 1

配套视频课程已更新完毕，大家可通过以下两种方式观看视频讲解：

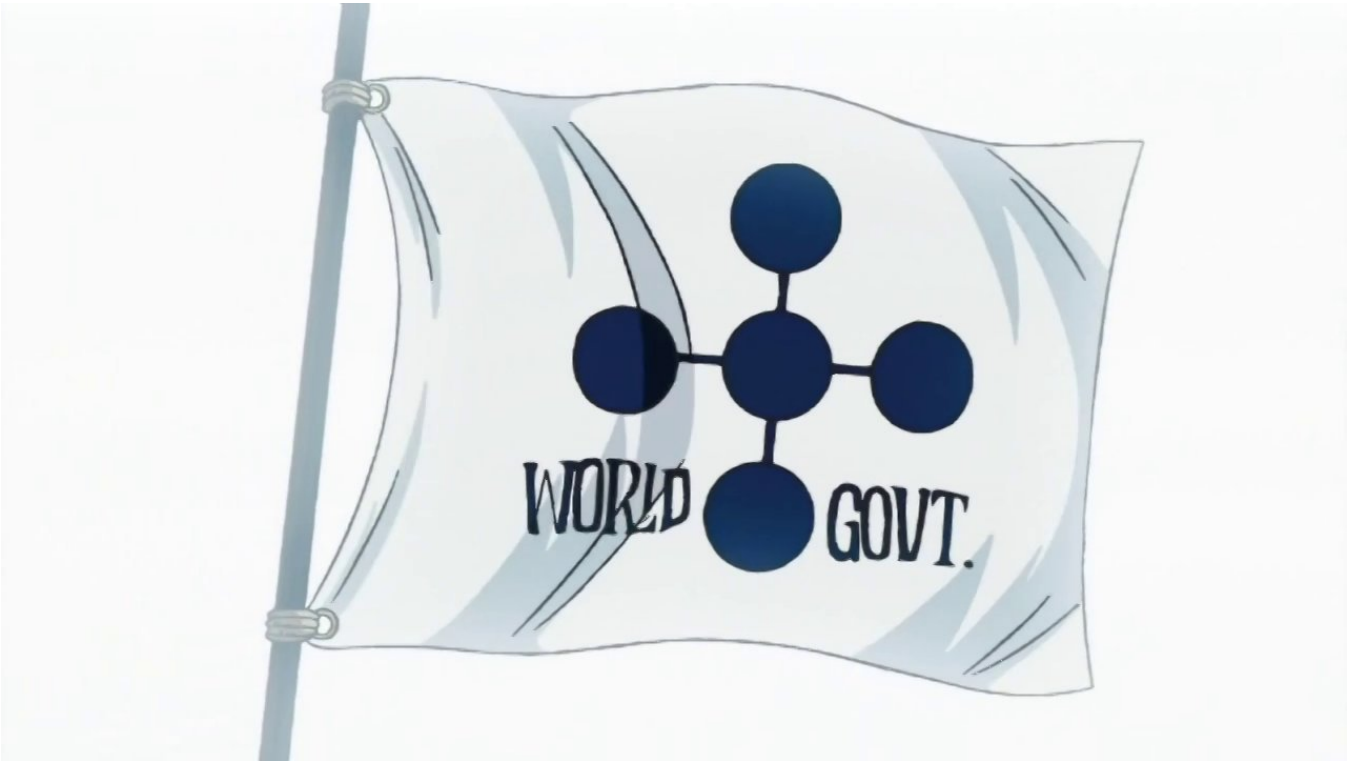
关注公众号： 爱编程的大丙 ，或者进入 大丙课堂 学习。

苏丙榅

合抱之木，生于毫末；九层之台，起于垒土；千里之行，始于足下。

1. 中介者

在海贼世界中，世界政府是拥有 170 个以上加盟国，依靠牢固的结盟而生成的最大规模的国际组织，以 “世界” 自居，支配整个世界，守卫掌管司法世界的秩序。世界政府以 “绝对正义” 的名义维护着世界的安全，为了铲除邪恶以及非法的事物可以不择手段，甚至杀害无辜的人，至于该事物是否 “正义” 则由政府说了算。



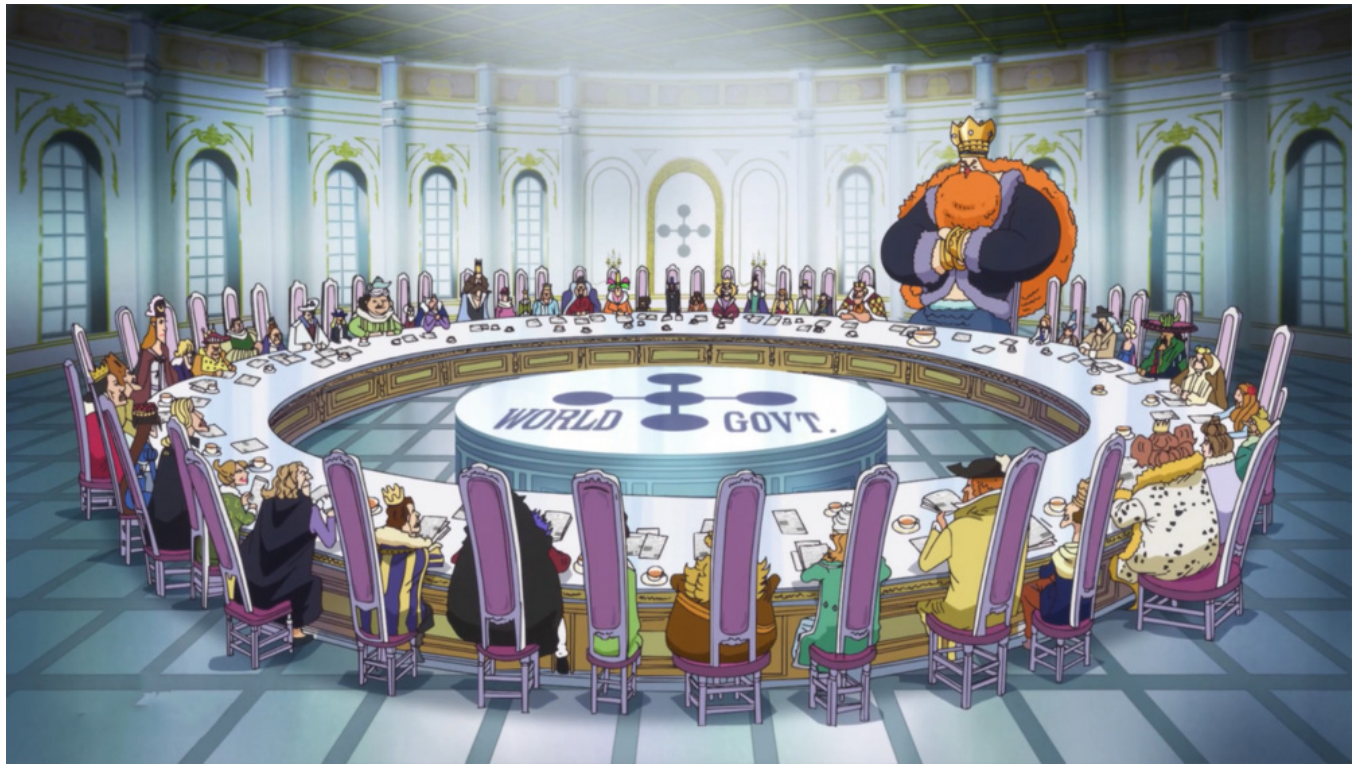
虽然在海贼的世界格局是非常混乱的，但是如果没有世界政府的存在它将更加混乱。以程序猿的视角来看，每个国家都是一个对象，一个国家需要和其它的很多国家都产生交集，这就难免产生冲

目录

突、掠夺和战争。如果有世界政府的存在，就可以在一定程度上避免各个国家之间的正面直接接触（对象解耦），还能起到一定的调节作用。

关于世界政府的这种组织形式，在设计模式中被称之为中介者模式。中介者模式可以减少对象之间混乱无序的依赖关系，从而使其耦合松散，限制对象之间的直接交互，迫使它们通过一个中介者对象进行合作。

如果不使用中介者模式，各个国家之间的关系就是一个网状结构，关系错综复杂，这样的系统也很难容易维护。有了中介者对象，可以将系统的网状结构变成以中介者为中心的放射形结构。每个具体的对象不再通过直接的联系与另一对象发生相互作用，而是通过中介者对象与另一个对象发生相互作用。



CMake 保姆级教程（下）
2023-03-15

CMake 保姆级教程（上）
2023-03-06

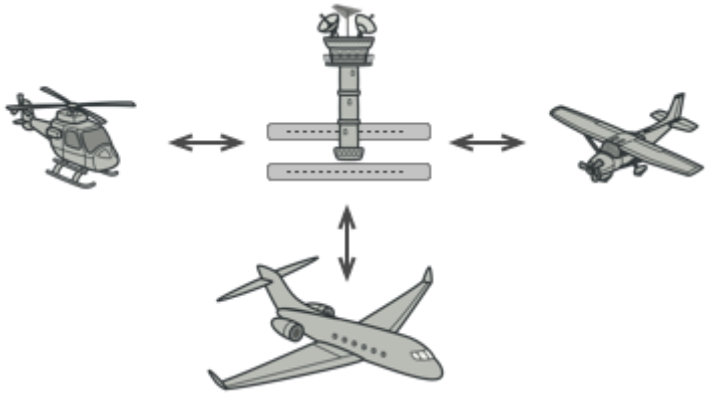
访问者模式 - 再见，香波地群岛
2022-09-22

模板方法模式 - 和平主义者
2022-09-21

状态模式 - 文斯莫克・山治
2022-09-20

中介者对象的设计，使得系统的结构体不会因为新对象的引入造成大量的修改工作。现实生活中关于中介者模式的应用也有很多，以下这些都是中介者：

- 联合国
- 各种中介机构：房产中介、婚恋中介、出国留学中介
- 十字路口指挥交通的信号灯或者交警
- 机场协调各架次飞机起降的塔台



2. 愿世界和平

2.1 国家

世界政府有 170 个以上加盟国，每个国家都有自己的诉求，所以必须得给这些国家提供一个抽象的基类，这个抽象类的定义如下：

C++

```cpp
// 抽象国家类
class Country
{
public:
    Country(){}
    // 发表声明
    virtual void declare(string msg, string country) = 0;
    virtual void setMessage(string msg) = 0;
    virtual string getName() = 0;
};
```

在这个基类中有三个纯虚函数，分别是：

- `declare()` ：各个国家可以通过这个函数发表不同的声明，强调立场，维护自身利益。
- `setMessage()` ：通过这个函数得到外界发布的关于自己国家的信息
- `getName()`：得到当前国家的名字

2.2 润滑剂

世界政府充当了一个中间人的角色，用于调解各个国家之间的关系。但是有一些国家并没有屈服于世界政府的淫威，加入这个组织，他们走到了世界政府的对立面加入到了路飞的老爸 蒙奇·D·龙 领导的革命军组织，试图推翻世界政府的统治。

不论是革命军还是世界政府他们都属于某一个组织，都可以充当中介者的角色，所以关于中介者类也可以定义出一个抽象的基类：

C++

```cpp
// 抽象的中介机构
class MediatorOrg
{
public:
    void addMember(Country* country);
    virtual void declare(string msg, Country* country, string name = string
    virtual ~MediatorOrg() {}
protected:
    map<string, Country*> m_countryMap;
};
```

这个中介者的抽象类中包含了一个受保护的 `map` 容器对象，`key` 值存储的是国家的名字，`value` 值存储的是国家的对象。 这样，和中介者对象关联的所有国家对象就可以全部被存储起来了。

- `addMember()` ：添加国家对象到中介者对象中并保存起来。
- `declare()` ：中介者对象可以将某个国家的诉求转达给其他国家，这是中介者类中的数据中转函数。

世界政府

世界政府就是一个中介者实例，所以可以让世界政府类继承上面的中介者抽象类，关于这个类的定义如下：

头文件 Mediator.h

C++

```
// 世界政府
class Country;
class WorldGovt : public MediatorOrg
{
public:
    void declare(string msg, Country* country, string name = string()) overr
};
```

在头文件中只需要对 `Country 国家类` 进行声明即可，不要包含它的头文件，否则会造成两个类的头文件交叉互相包含。

源文件 Mediator.cpp

C++

```
#include <iostream>
#include "Mediator.h"
#include "Country.h"
using namespace std;

// 基类的成员添加函数
void MediatorOrg::addMember(Country* country)
{
    m_countryMap.insert(make_pair(country->getName(), country));
}

// 在子类中重写发表声明的函数
void WorldGovt::declare(string msg, Country* country, string name)
{
    if (m_countryMap.find(name) != m_countryMap.end())
```

```
    {
        string str = msg + "【来自: " + country→getName() + "】";
        m_countryMap[name]→setMessage(str);
    }
}
```

在世界政府类中，通过 `declare()` 会将一个国家的声明转发给另一个国家

- `msg`：发布声明的国家发布的信息
- `country`：发布声明的国家的对象
- `name`：中介者需要将声明转达给这个国家

在这个类的源文件中就可以包含国家类的头文件 `Country.h` 了，如果不包含头文件就无法识别出国家类提供的成员函数。

革命军

头文件 Mediator.h

C++

```
// 革命军
class GeMingArmy : public MediatorOrg
{
public:
    void declare(string msg, Country* country, string name = string()) overr
};
```

源文件 Mediator.cpp

C++

```cpp
// 在子类中重写发表声明的函数
void GeMingArmy::declare(string msg, Country* country, string name)
{
    string str = msg + "【来自: " + country→getName() + "】";
    for (const auto& item : m_countryMap)
    {
        if (item.second == country)
        {
            continue;
        }
        item.second→setMessage(str);
    }
}
```

在革命军类中，通过 `declare()` 会将一个国家的声明转发给其他的所有国家。

2.3 入伙

有了世界政府和革命军另个中介者组织之后，世界各国就可以站队了，所以哪个国家要加入哪个组织需要明确的指定出来，我们来修改一下国家类的基类：

C++

```cpp
// 抽象国家类
class Country
{
```

```
public:
    Country(MediatorOrg* mediator) : m_mediator(mediator) {}
    // 发表声明
    virtual void declare(string msg, string country) = 0;
    virtual void setMessage(string msg) = 0;
    virtual string getName() = 0;
    virtual ~Country() {}
protected:
    MediatorOrg* m_mediator = nullptr;
};
```

在创建国家对象的时候，需要指定出其认可的中介者对象，这样就能知道这个国家的立场了。

接下来就是基于这个抽象的国家类定义出一些可被实例化的子国家类：

C++

```
#pragma once
#include <string>
#include <iostream>
#include "Mediator.h"
using namespace std;

// 抽象国家类
class Country
{
public:
    Country(MediatorOrg* mediator) : m_mediator(mediator) {}
    // 发表声明
    virtual void declare(string msg, string country) = 0;
    virtual void setMessage(string msg) = 0;
```

```
    virtual string getName() = 0;
    virtual ~Country() {}
protected:
    MediatorOrg* m_mediator = nullptr;
};

// 阿拉巴斯坦
class Alabasta : public Country
{
public:
    using Country::Country;
    void declare(string msg, string country) override
    {
        m_mediator->declare(msg, this, country);
    }
    void setMessage(string msg) override
    {
        cout << "阿拉巴斯坦得到的消息: " << msg << endl;
    }
    string getName() override
    {
        return "阿拉巴斯坦";
    }
};

// 德雷斯罗萨
class Dressrosa : public Country
{
```

上面一共定义了四个字国家类分别是 `Alabasta`、`Dressrosa`、`Lulusia`、`Kamabaka`，这四个国家在发布声明的时候不是直接发布，而是将消息给到了他们所信任的中介者对象，通过这个中介者对象将消息发布给对应的其他国家，这样国家和国家直接就实现了解耦合，更利于程序的维护和扩展。

2.4 交流

最后就可以让上面的四个国家加入组织，通过中介者对象进行交流了：

C++

```
int main()
{
    // 世界政府
    WorldGovt* world = new WorldGovt;
    Alabasta* alaba = new Alabasta(world);
    Dressrosa* dresa = new Dressrosa(world);
    // 世界政府添加成员
    world→addMember(alaba);
    world→addMember(dresa);
    // 世界政府成员发声
    alaba→declare("德雷斯罗萨倒卖军火，搞得我国连年打仗，必须给个说法!!!", dresa-
    dresa→declare("天龙人都和我多弗朗明哥做生意，你算老几，呸!!!", alaba→getNam
    cout << "====================================" << endl;
    // 革命军
    GeMingArmy* geming = new GeMingArmy;
    Lulusia* lulu = new Lulusia(geming);
    Kamabaka* kama = new Kamabaka(geming);
    geming→addMember(lulu);
    geming→addMember(kama);
```

```
20      lulu->declare("我草，我的国家被伊姆毁灭了!!!", lulu->getName());
21
22      delete world;
23      delete alaba;
24      delete dresa;
25      delete geming;
26      delete lulu;
27      delete kama;
28      return 0;
29  }
```

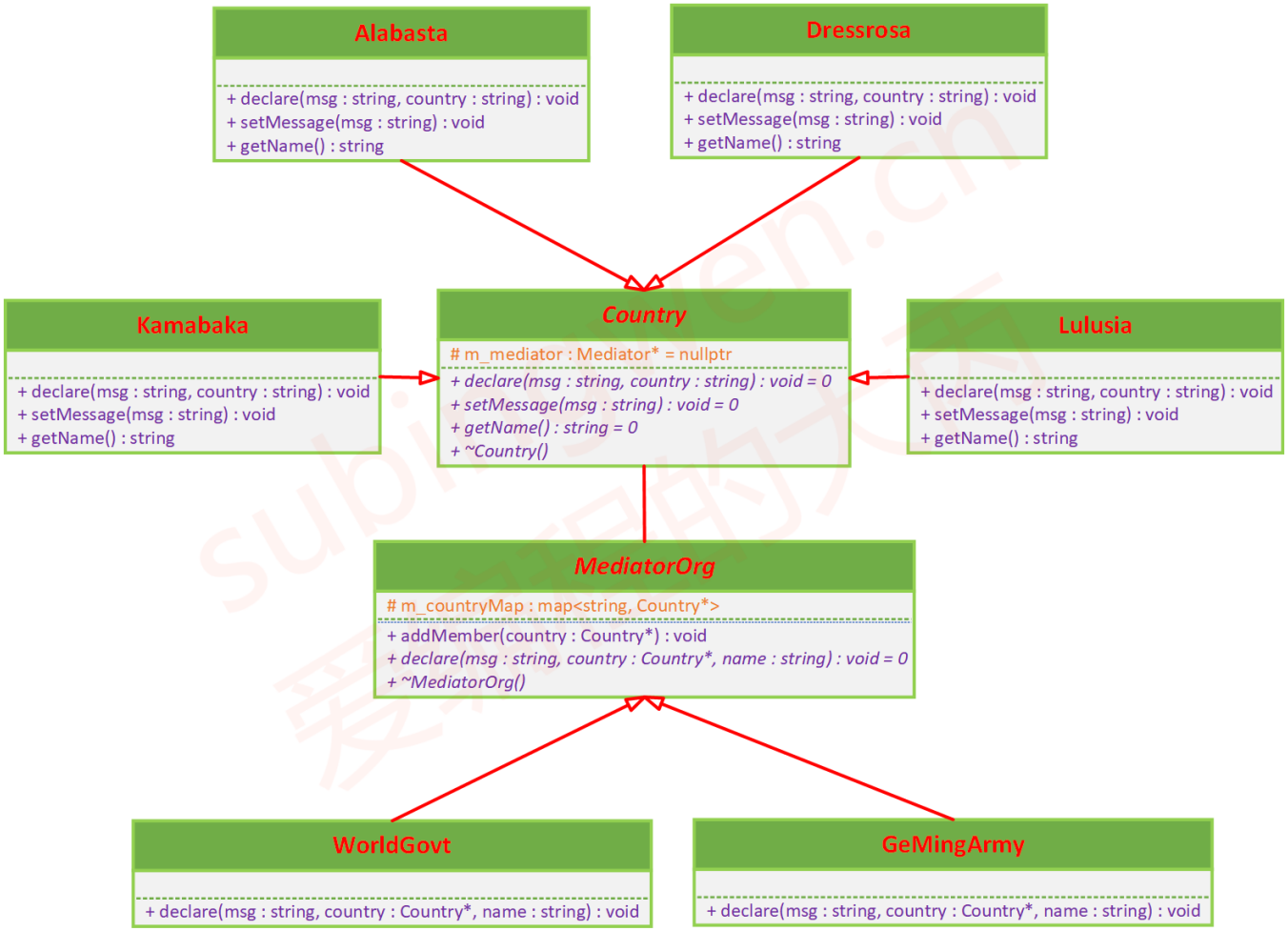
上面的四个国家有两个国家加入了世界政府，有两个国家加入了革命军，最后看一下他们交流的结果：

C++

```
1  德雷斯罗萨得到的消息：德雷斯罗萨倒卖军火，搞得我国连年打仗，必须给个说法!!!【来自：阿拉巴
2  阿拉巴斯坦得到的消息：天龙人都和我多弗朗明哥做生意，你算老几，呸!!!【来自：德雷斯罗萨】
3  ====================================
4  卡玛巴卡得到的消息：我草，我的国家被伊姆毁灭了!!!【来自：露露西亚】
```

3. 结构图

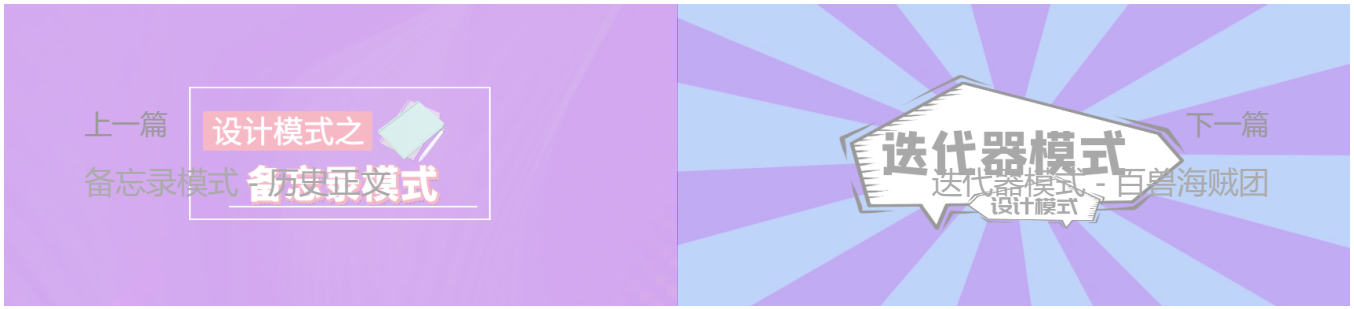
最后根据上面的例子把中介者模式对应的 UML 类图画一下（学会了中介者模式之后，应该先画 UML 类图再写程序。）



当一些对象和其他对象紧密耦合以致难以对其进行修改时；当组件因过于依赖其他组件而无法在不同应用中复用时；当为了能在不同情景下复用一些基本行为，导致需要被迫创建大量组件子类时，都可使用中介者模式。

设计模式

打赏



相关推荐

评论

昵称 邮箱 网址(http://)

来都来了, 说点什么吧...

提交

1 评论

Anonymous Chrome 86.0.4240.198 Windows 11

2023-03-18

回复

123

Powered By Valine

v1.5.1