# 行为型模式-模板模式

通过一个简单的股票查询客户端来说明模板模式，通过自定义的火车票查询器来查询火车票价格需要以下几个步骤：登录、设置车次、查询、展示.

示例代码：

In [2]:
```
from abc import ABCMeta, abstractmethod

class ticket:
    __metaclass__ =  ABCMeta
    @abstractmethod
    def login(self, username, password):
        pass

    def set_code(self, value):
        self.code =value

    @abstractmethod
    def show(self):
        pass

class xiecheng_ticket(ticket):
    def login(self, username, password):
        if username == 'xiecheng' and password == 'abc123':
            print("Login xiecheng successfully!")
        else:
            raise ValueError('Login xiecheng failed!')

    def show(self):
        print("携程查询到了火车票%s" %self.code)

class feizhu_ticket(ticket):
    def login(self, username, password):
        if username == 'feizhu' and password == 'abc123':
            print("Login feizhu successfully!")
        else:
            raise ValueError("Login feizhu failed!")

    def show(self):
        print("飞猪查询到了火车票%s" %self.code)

if __name__ == '__main__':
    xiecheng = xiecheng_ticket()
    xiecheng.login("xiecheng", "abc123")
    xiecheng.set_code("T236")
    xiecheng.show()
    print("#" * 20)
    #xiecheng.login("xiecheng", 'xiecheng')
    print("#" * 20)
    feizhu = feizhu_ticket()
    feizhu.login("feizhu", "abc123")
    feizhu.set_code("T236")
    feizhu.show()
```

```
Login xiecheng successfully!
携程查询到了火车票T236
####################
####################
Login feizhu successfully!
飞猪查询到了火车票T236
```

如上代码每次操作，都会调用登录，设置代码 查询 展示这几步，有些繁琐，所以我们需将这几步过程封装成一个接口.
示例代码：

In [3]:
```
from abc import ABCMeta, abstractmethod
class ticket:
    __metaclass__ =  ABCMeta


    @abstractmethod
    def login(self, username, password):
        pass

    def set_code(self, value):
        self.code =value

    @abstractmethod
    def show(self):
        pass


class xiecheg_ticket(ticket):
    def login(self, username, password):
        if username == 'xiecheng' and password == 'abc123':
            print("Login xiecheng successfully!")
        else:
            raise ValueError('Login xiecheng failed!')

    def show(self):
        print("携程查询到了火车票%s" %self.code)

class feizhu_ticket(ticket):
    def login(self, username, password):
        if username == 'feizhu' and password == 'abc123':
            print("Login feizhu successfully!")
        else:
            raise ValueError("Login feizhu failed!")

    def show(self):
        print("飞猪查询到了火车票%s" %self.code)


class ticket_agent:
    def __init__(self, support, username, password, code):
        self.support = self.__get_support(support)
        self.username = username
        self.password = password
        self.code = code

    def __get_support(self, value):
        if value == 'xiecheng':
            return xiecheg_ticket()
        elif value == 'feizhu':
            return feizhu_ticket()
        else:
            raise TypeError("not found %s" %value)

    def operateShow(self):
        self.support.login(self.username, self.password)
        self.support.set_code(self.code)
        self.support.show()

if __name__ == '__main__':
    x = ticket_agent("xiecheng", "xiecheng", "abc123", "T236")
    x.operateShow()
    print('#' * 10)
    y = ticket_agent("feizhu", "feizhu", "abc123", 'D7046')
    y.operateShow()
```

```
Login xiecheng successfully!
携程查询到了火车票T236
##########
Login feizhu successfully!
飞猪查询到了火车票D7046
```

```
模板模式的优点:
    1)可变的部分可以充分扩展, 不变的步骤可以充分封装
    2)提取公共代码, 减少冗余代码, 便于维护.
    3)具体过程可以定制, 总体流程方便掌控.
```

In [ ]: