**项目实战**

## [Tornado实战项目(伪JD商城)](https://www.cnblogs.com/kongqi816-boke/p/5883970.html)

**预备知识**

在之前tornado商城项目中，在开始之前需要引入一些项目设计知识，如接口，抽象方法抽象类，组合，程序设计原则等，个人理解项目的合理设计可增加其灵活性，

降低数据之间的耦合性，提高稳定性，下面介绍一些预备知识

**1、接口**

其实py中没有接口这个概念。要想实现接口的功能,可以通过主动抛出异常来实现

**接口作用：对派生类起到限制的作用**

|  |
| --- |
| #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  """  接口，python中的接口，通过在父类中主动抛出异常实现  接口的作用:起到了限制的作用  """    class IFoo:      def fun1(self):          pass          raise Exception("错误提示")    class Bar(IFoo):      def fun1(self):          #方法名必须和父类中的方法名相同，不然没办法正常执行，会抛出异常          print("子类中如果想要调用父类中的方法，子类中必须要有父类中的方法名")      def fun2(self):          print("test")    obj = Bar()  obj.fun2() |

**2.抽象方法抽象类**

抽象类，抽象方法是普通类和接口的综合，即可以继承也可以起到限制作用

由于python 本身没有抽象类、接口的概念，所以要实现这种功能得abc.py 这个类库，

具体实现方法如下 :

|  |
| --- |
| #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  """  抽象类，抽象方法  抽象类，抽象方法是普通类和接口的综合，即可以继承也可以起到限制作用  """    import abc  class Foo(metaclass=abc.ABCMeta):      def fun1(self):          print("fun1")        def fun2(self):          print("fun2")        @abc.abstractclassmethod      def fun3(self):          pass      class Bar(Foo):      def fun3(self):          print("子类必须有父类的抽象方法名，不然会抛出异常")      obj = Bar()  obj.fun1()  obj.fun2()  obj.fun3() |

**3.组合**

python中“多用组合少用继承”，因为继承的偶合性太强，可以把基类，当做参数传入派生类中，用于解偶

**§继承**

|  |
| --- |
| #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  #继承    class Animals:      def eat(self):          print(self.Name + " eat")      def drink(self):          print(self.Name + " drink")    class Person(Animals):      def \_\_init\_\_(self, name):          self.Name = name        def think(self):          print(self.Name + " think")  obj = Person("user1")  obj.drink()  obj.eat()  obj.think() |

**§组合**

|  |
| --- |
| class Animals:      def \_\_init\_\_(self,name):          self.Name = name        def eat(self):          print(self.Name + " eat")        def drink(self):          print(self.Name + " drink")    class Person:      def \_\_init\_\_(self, obj):          self.obj = obj        def eat(self):          self.obj.eat()        def think(self,name):          print(name + " think")    animals = Animals("animals")  obj = Person(animals)  obj.think("person")  obj.eat() |

**4.依赖注入**

像上一例中，如果有多层关系时，需要传入多个对象，为了解决这个问题就引入了依赖注入，

如上例在Person类实例化时自动传入**Animals对象**

|  |
| --- |
| class Foo:      def \_\_init\_\_(self):          self.name = 111          def fun(self)          print(self.name)    obj = Foo() #obj是Foo的实例化对象 |

在python中一切皆对象，Foo是通过type类创建的

|  |
| --- |
| #!/usr/bin/env python  # -\*- coding:utf-8 -\*-    class MyType(type):        def \_\_call\_\_(cls, \*args, \*\*kwargs):          obj = cls.\_\_new\_\_(cls, \*args, \*\*kwargs)          obj.\_\_init\_\_(\*args, \*\*kwargs)          return obj      class Foo(metaclass=MyType):        def \_\_init\_\_(self, name):          self.name = name        def f1(self):          print(self.name) |
| 解释器解释：     1.遇到 class Foo，执行type的\_\_init\_\_方法     1.Type的init的方法里做什么么呢？不知道         obj = Foo(123)     3.执行Type的 \_\_call\_\_方法         执行Foo类的 \_\_new\_\_方法         执行Foo类的 \_\_init\_\_ 方法 |

new 和 \_\_init()和\_\_metaclass\_\_:

* \_\_new\_\_函数是实例一个类所要调用的函数,每当我们调用obj = Foo()来实例一个类时,都是先调用\_\_new\_\_()
* 然后再调用\_\_init\_\_()函数初始化实例. \_\_init\_\_()在\_\_new\_\_()执行后执行,
* 类中还有一个属性 \_\_metaclass\_\_，其用来表示该类由 谁 来实例化创建，所以，我们可以为 \_\_metaclass\_\_ 设置一个type类的派生类，从而查看 类 创建的过程。

那么依赖注入的实现方法，自定义一个type方法，实例化类的时候指定由自定义的type方法创建，

具体实现方法如下:

|  |
| --- |
| #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  # 依赖注入应用  #DI  class Mapper:      \_\_mapper\_relation ={}        @staticmethod      def register(cls,value):          Mapper.\_\_mapper\_relation[cls] = value        @staticmethod      def exist(cls):          if cls in Mapper.\_\_mapper\_relation:              return True          return False        @staticmethod      def value(cls):          return Mapper.\_\_mapper\_relation[cls]      class MyType(type):      def \_\_call\_\_(self, \*args, \*\*kwargs):          obj = self.\_\_new\_\_(self, \*args, \*\*kwargs)          arg\_list = list(args)          if Mapper.exist(self):              value=Mapper.value(self)              arg\_list.append(value)          obj.\_\_init\_\_(\*arg\_list, \*\*kwargs)          return obj      #定义由谁来实例化  class Foo(metaclass=MyType):      def \_\_init\_\_(self,name):          self.name = name        def f1(self):          print(self.name)    class Bar(metaclass=MyType):      def \_\_init\_\_(self,name):          self.name = name        def f1(self):          print(self.name)    Mapper.register(Foo,"test1")  Mapper.register(Bar,"test12")  f=Foo()  print(f.name) |

**5.程序的设计原则**

**1. 单一责任原则(SRP)**

|  |
| --- |
| 一个对象只对一个元素负责    优点;  　　消除耦合，减小因需求变化引起代码僵化 |

**2.开放封闭原则(OCP)**

|  |
| --- |
| 例如装饰器，可以对独立的功能实现扩展，但对源码不能进行修改  对扩展开放，对修改封闭    　　　　优点：  　　　　　　按照OCP原则设计出来的系统，降低了程序各部分之间的耦合性，其适应性、灵活性、稳定性都比较好。当已有软件系统需要增加新的功能时，    　　　　　　不需要对作为系统基础的抽象层进行修改，只需要在原有基础上附加新的模块就能实现所需要添加的功能。增加的新模块对原有的模块完全没有影响或影响很小，    　　　　　　这样就无须为原有模块进行重新测试    　　　　如何实现 ？    　　　　　　在面向对象设计中，不允许更必的是系统的抽象层，面允许扩展的是系统的实现层，所以解决问题的关键是在于抽象化。    　　　　　　在面向对象编程中，通过抽象类及接口，规定具体类的特征作为抽象层，相对稳定，不需要做更改的从面可以满足“对修改关闭”的原则；而从抽象类导出的具体 类可以    　　　　　　改变系统 的行为，从而满足“对扩展开放的原则" |

**3.里氏替换原则(LSP)**

|  |
| --- |
| 子类可以替换父类，父类出现的地方都可以用子类替换  可以使用任何派生类（子类）替换基类    　　　　优点：  　　　　　　可以很容易的实现同一父类下各个子类的互换，而客户端可以毫不察觉 |

**4.接口分享原则(ISP)**

|  |
| --- |
| 对于接口进行分类避免一个接口的方法过多，避免”胖接口"  　　　　优点：  　　　　　　会使一个软件系统功能扩展时，修改的压力不会传到别的对象那里  　　　　如何实现 ?  　　　　　　得用委托分离接口  　　　　　　利用多继承分离接口 |

**5.依赖倒置原则(DIP)**

|  |
| --- |
| 高层模块不应该依赖低层模块，二者都应该依赖其抽象（理解为接口）；抽象不应该依赖细节；细节应该依赖抽象    隔离关系，使用接口或抽象类代指  高层次的模块不应该依赖于低层次的模块，而是，都应该依赖于抽象  　　　　优点：  　　　　　　使用传统过程化程序设计所创建的依赖关系，策略依赖于细节，这是糟糕的，因为策略受到细节改变的影响。  　　　　　　依赖倒置原则使细节和策略都依赖于抽象，抽象的稳定性决定了系统的稳定性 |

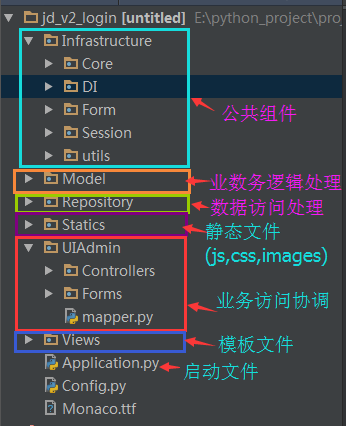
**6.依赖注入(DI)和控制反转原则(ICO)**

|  |
| --- |
| 使用钩子再原来执行流程中注入其他对象 |

**tornado项目设计实例**

 实例只包含登录，写此实例目的在于更好的理解及应用以上的内容

**1、目录规划**



 说明：

|  |
| --- |
| Infrastructure 目录：公共组件目录    Model:业务逻辑处理目录    Repository:　数据仓库及数据处理目录    Statics：静态文件目录如（css,js,images等)    UIAdmin: UI层    Views:模板文件目录    Application.py ： 服务启动文件 |

**2.业务访问流程**

|  |
| --- |
| 介绍完目录规划，那就来讲讲业务访问流程及数据走向    启动服务后，客户端访问URL，根据tornado路由找到相对的handler进行处理  找到handler后其相对方法(get/post/delete/put)中调用Model逻辑处理层方法进行处理并接收处理结果  Model逻辑处理层需      ①创建接口      ②建模      ③创建协调层  　　　创建完之后 ，由协调层（这里通用Services)调用数据层方法并接收处理结果返回给handler    　　4.数据处理层接收到Model调用后，处理数据并将数据返回给Model业务逻辑处理层    　　5.最终handler接收到最终结果，进行判断处理，并将处理结果返回给用户 |

**3、落实实施**

**1.启动文件，路由关系配置**

#!/usr/bin/env python

# -\*- coding:utf-8 -\*-

import tornado.ioloop

import tornado.web

from UIAdmin.Controllers import Account

from UIAdmin.Controllers import Region

from UIAdmin.Controllers import Customer

from UIAdmin.Controllers import Merchant

from UIAdmin import mapper

settings = {

'template\_path': 'Views',

'static\_path': 'Statics',

'static\_url\_prefix': '/statics/',

}

application = tornado.web.Application([

(r"/login", Account.LoginHandler),

(r"/check", Account.CheckCodeHandler),

],\*\*settings)

if \_\_name\_\_ == "\_\_main\_\_":

application.listen(8000)

tornado.ioloop.IOLoop.instance().start()

说明：

settings 中指定配置，如模板文件路径，静态文件路径等

application ：路由配置，那个路径由那个handler进行处理

**2.handler配置**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import io

from Infrastructure.Core.HttpRequest import BaseRequestHandler

from Infrastructure.utils import check\_code

from Model.User import UserService

class LoginHandler(BaseRequestHandler):

def get(self, \*args, \*\*kwargs):

self.render("Admin/Account/login.html")

def post(self, \*args, \*\*kwargs):

username = self.get\_argument("username",None)

email = self.get\_argument("email",None)

pwd = self.get\_argument("pwd",None)

code = self.get\_argument("checkcode",None)

service = UserService()

result = service.check\_login(user=username,email=email,pwd=pwd)

#obj封装了所有的用户信息，UserModel对象

if result and code.upper() == self.session["CheckCode"].upper():

self.session['username'] = result.username

self.redirect("/ProvinceManager.html")

else:

self.write("alert('error')")

handler中主要是针对数据访问方式的不同，给出不同的处理方法，并将结果返回给客户端

**3.Model 逻辑处理层**

|  |
| --- |
| 逻辑处理层中，着重看的有三点：      建模      接口      协调 |

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

#建模

from Infrastructure.DI.Meta import DIMetaClass

class VipType:

VIP\_TYPE = (

{'nid': 1, 'caption': '铜牌'},

{'nid': 2, 'caption': '银牌'},

{'nid': 3, 'caption': '金牌'},

{'nid': 4, 'caption': '铂金'},

)

def \_\_init\_\_(self, nid):

self.nid = nid

def get\_caption(self):

caption = None

for item in VipType.VIP\_TYPE:

if item['nid'] == self.nid:

caption = item['caption']

break

return caption

caption = property(get\_caption)

class UserType:

USER\_TYPE = (

{'nid': 1, 'caption': '用户'},

{'nid': 2, 'caption': '商户'},

{'nid': 3, 'caption': '管理员'},

)

def \_\_init\_\_(self, nid):

self.nid = nid

def get\_caption(self):

caption = None

for item in UserType.USER\_TYPE:

if item['nid'] == self.nid:

caption = item['caption']

break

return caption

caption = property(get\_caption)

class UserModel:

def \_\_init\_\_(self, nid, username,password, email, last\_login, user\_type\_obj, vip\_type\_obj):

self.nid = nid

self.username = username

self.email = email

self.password = password

self.last\_login = last\_login

self.user\_type\_obj = user\_type\_obj

self.vip\_type\_obj = vip\_type\_obj

接口  IUseRepository类：接口类，用于约束数据库访问类的方法

class IUserRepository:

def fetch\_one\_by\_user(self,user,pwd):

"""

根据用户名和密码获取对象

:param user:

:param pwd:

:return:

"""

def fetch\_one\_by\_email(self, user, pwd):

"""

根据邮箱和密码获取对象

:param user:

:param pwd:

:return:

"""

 协调  协调作用主要是调用数据处理层的方法，并将数据处理层处理后的结果返回给它的上一层的调度者

class UserService(metaclass=DIMetaClass):

def \_\_init\_\_(self, user\_repository):

"""

:param user\_repository: 数据仓库对象

"""

self.userRepository = user\_repository

def check\_login(self,user,email,pwd):

if user:

#数据仓库执行SQL后返回的字典

#{"nid":1,username:xxx,vip:2,usertype:1}

ret = self.userRepository.fetch\_one\_by\_user(user,pwd)

else:

ret = self.userRepository.fetch\_one\_by\_email(email,pwd)

return ret

**4.Repository数据处理层**

将处理后结果（usermodel对象)返回给上一层调度者(UserService)

#!/usr/bin/env python

# -\*- coding:utf-8 -\*-

#数据表创建

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy import Column

from sqlalchemy import Integer, Integer, CHAR, VARCHAR, ForeignKey, Index, DateTime, DECIMAL, TEXT

from sqlalchemy.orm import sessionmaker, relationship

from sqlalchemy import create\_engine

engine = create\_engine("mysql+pymysql://root:123@127.0.0.1:3306/ShoppingDb?charset=utf8", max\_overflow=5)

Base = declarative\_base()

class Province(Base):

"""

省

"""

\_\_tablename\_\_ = 'province'

nid = Column(Integer, primary\_key=True)

caption = Column(VARCHAR(16), index=True)

class City(Base):

"""

市

"""

\_\_tablename\_\_ = 'city'

nid = Column(Integer, primary\_key=True)

caption = Column(VARCHAR(16), index=True)

province\_id = Column(Integer, ForeignKey('province.nid'))

class County(Base):

"""

县（区）

"""

\_\_tablename\_\_ = 'county'

nid = Column(Integer, primary\_key=True)

caption = Column(VARCHAR(16), index=True)

city\_id = Column(Integer, ForeignKey('city.nid'))

class UserInfo(Base):

"""

用户信息

"""

\_\_tablename\_\_ = 'userinfo'

nid = Column(Integer, primary\_key=True)

USER\_TYPE = (

{'nid': 1, 'caption': '用户'},

{'nid': 2, 'caption': '商户'},

{'nid': 3, 'caption': '管理员'},

)

user\_type = Column(Integer)

VIP\_TYPE = (

{'nid': 1, 'caption': '铜牌'},

{'nid': 2, 'caption': '银牌'},

{'nid': 3, 'caption': '金牌'},

{'nid': 4, 'caption': '铂金'},

)

vip = Column(Integer)

username = Column(VARCHAR(32))

password = Column(VARCHAR(64))

email = Column(VARCHAR(64))

last\_login = Column(DateTime)

ctime = Column(DateTime)

\_\_table\_args\_\_ = (

Index('ix\_user\_pwd', 'username', 'password'),

Index('ix\_email\_pwd', 'email', 'password'),

)



#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

from Model.User import IUserRepository

from Model.User import UserModel

from Model.User import UserType

from Model.User import VipType

from Repository.Admin.DbConnection import DbConnection

class UserRepository(IUserRepository):

def \_\_init\_\_(self):

self.db\_conn = DbConnection()

def fetch\_one\_by\_email(self, email, password):

ret = None

cursor = self.db\_conn.connect()

sql = """select nid,username,password,email,last\_login,vip,user\_type from userinfo where email=%s and password=%s"""

cursor.execute(sql, (email, password))

db\_result = cursor.fetchone()

self.db\_conn.close()

print(type(db\_result), db\_result)

if db\_result:

ret = UserModel(nid=db\_result['nid'],

username=db\_result['username'],

password=db\_result['password'],

email=db\_result['email'],

last\_login=db\_result['last\_login'],

user\_type\_obj=UserType(nid=db\_result['user\_type']),

vip\_type\_obj=VipType(nid=db\_result['vip']),)

return ret

return db\_result

def fetch\_one\_by\_user(self, username, password):

ret = None

cursor = self.db\_conn.connect()

sql = """select nid,username,password,email,last\_login,vip,user\_type from userinfo where username=%s and password=%s"""

cursor.execute(sql, (username, password))

db\_result = cursor.fetchone()

self.db\_conn.close()

if db\_result:

#建模，将usermodel对象返回给上一层调用者，因为要向用户展示的user\_type不可能为1，2这些数据而应该是相对的caption

ret = UserModel(nid=db\_result['nid'],

username=db\_result['username'],

password=db\_result['password'],

email=db\_result['email'],

last\_login=db\_result['last\_login'],

user\_type\_obj=UserType(nid=db\_result['user\_type']),

vip\_type\_obj=VipType(nid=db\_result['vip']),)

return ret

return db\_result

**5.Handler最终处理**

接收到最终处理结果后判断，并返回数据给用户

说明：

　　有没有注意到UserService是怎么和数据处理层建立联系的？

　　这里我们用到了依赖注入，具体配置如下:

#!/usr/bin/env python

# -\*- coding:utf-8 -\*-

#依赖注入

class DIMapper:

\_\_mapper\_dict = {}

@staticmethod

def inject(cls, arg):

if cls not in DIMapper.\_\_mapper\_dict:

DIMapper.\_\_mapper\_dict[cls] = arg

@staticmethod

def get\_mappers():

return DIMapper.\_\_mapper\_dict

class DIMetaClass(type):

def \_\_call\_\_(cls, \*args, \*\*kwargs):

# 获取配置的对应的对象，携带进入

obj = cls.\_\_new\_\_(cls, \*args, \*\*kwargs)

mapper\_dict = DIMapper.get\_mappers()

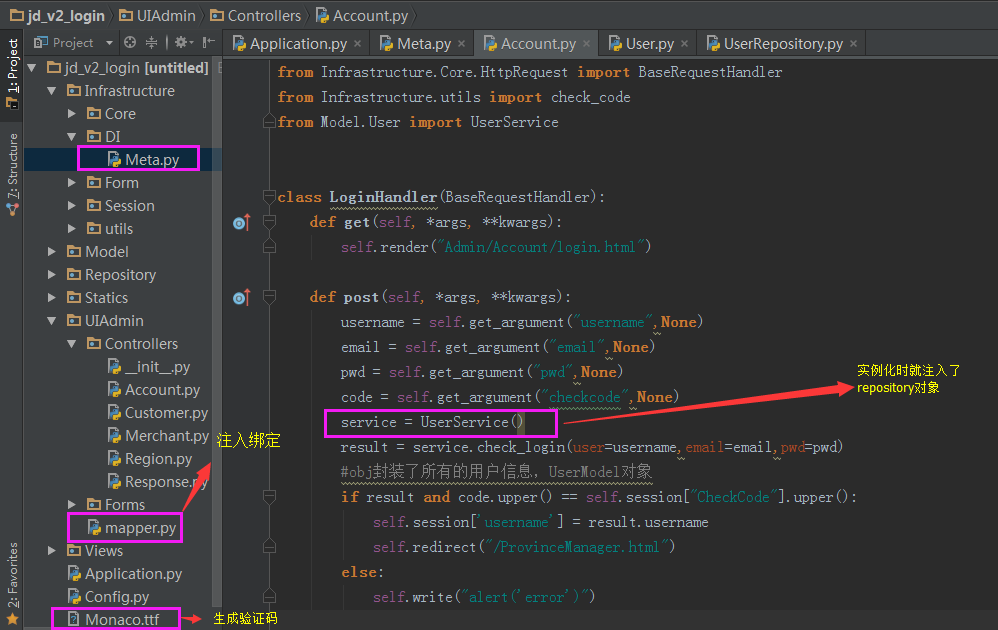
if cls in mapper\_dict:

cls.\_\_init\_\_(obj, mapper\_dict[cls])

else:

cls.\_\_init\_\_(obj, \*args, \*\*kwargs)

return obj



#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

# 依赖注入绑定

from Infrastructure.DI import Meta

from Model.User import UserService

from Repository.Admin.UserRepository import UserRepository

Meta.DIMapper.inject(UserService,UserRepository())

**6.静态文件代码**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<meta name="viewport" content="width=device-width" />

<meta http-equiv="X-UA-Compatible" content="IE=8" />

<title>购物商城</title>

<link href="/statics/Admin/Css/common.css" rel="stylesheet" />

<link href="/statics/Admin/Css/account.css" rel="stylesheet" />

</head>

<body>

<div class="account-container bg mt10">

<div class='header clearfix'>

<div>

<a href="/home/index">

<img src="/statics/Admin/Images/mll\_logo.gif">

</a>

</div>

</div>

</div>

<div class='account-container mt30'>

<div class='body clearfix pd10' style='position: relative;'>

<div class='logo left'>

<img style='height:350px;' src="/statics/Admin/Images/login\_logo.png" />

</div>

<div class='login left mt30'>

<form id='Form' action='/login' method='POST'>

<div class='group mt10'>

<label class='tip'><span class="red">\*</span>用户名：</label>

<input type='text' require='true' label='用户名' Field='string' range='4-40' name='username' />

<i class='i-name'></i>

</div>

<div class='group'>

<label class='tip'><span class="red">\*</span>密码：</label>

<input type='password' require='true' label='密码' min-len='6' name='pwd' />

<i class='i-pwd'></i>

</div>

<div class='group'>

<label class='tip'><span class="red">\*</span>验证码：</label>

<input type='text' require='true' label='验证码' style='width:80px;' name='checkcode' />

<a style='width:125px;display:inline-block;'><img class='checkcode' onclick='ChangeCode();' id='imgCode' src='/check' /></a>

</div>

<div class='group font12 mb0'>

<label class='tip'></label>

<label style='width:246px;display: inline-block;'>

<input id='protocol' name='protocol' type='checkbox' checked='checked' />

<span>自动登录</span>

<span class='ml10'><a href='#'>忘记密码？</a></span>

</label>

</div>

<div class='group mt0'>

<label class='tip'></label>

<input type='submit' class='submit' value='登 录' />

</div>

</form>

<div class='go-register'><a href='#'>免费注册 >> </a></div>

</div>

</div>

</div>

<div class='account-container mt20' style='text-align:center;color:#555;'>

© 2004-2015 www.xxxxx.com.cn All Rights Reserved. xxxxx 版权所有

</div>

<script src="/statics/Admin/js/jquery-1.8.2.min.js"></script>

<script src="/statics/Admin/js/treebiao.js"></script>

<script type="text/javascript">

$(function(){

$.login('#Form','');

});

function ChangeCode() {

var code = document.getElementById('imgCode');

code.src += '?';

}

</script>

</body>

</html>

.header{

padding:15px 0px;

}

.body{

border: 1px solid #d7d7d7;

padding: 40px;

padding-right: 0;

}

.body .logo{

width:50%;

}

.body .login{

width:50%;

color: #555;

}

.body .register{

width: 630px;

border-right: 1px dashed #e5e5e5;

color: #555;

}

.body .register .group,.body .login .group{

margin:15px 0px;

height:38px;

font-size:14px;

position:relative;

line-height:38px;

}

.body .register .group .tip,.body .login .group .tip{

width: 100px;

display: inline-block;

text-align: right;

font-size: 14px;

}

.body .register .group label .red,.body .login .group label .red{

margin:0 5px;

}

.body .register .group input[type='text'],.body .register .group input[type='password'],

.body .login .group input[type='text'],.body .login .group input[type='password']{

width:210px;

height:32px;

padding:0 30px 0 4px;

border: 1px solid #cccccc;

}

.body .register .group i,.body .login .group i{

position: absolute;

left: 330px;

}

.body .register .group .i-name,.body .login .group .i-name{

background: url(../Images/i\_name.jpg) no-repeat scroll 0 0 transparent;

height: 16px;

top: 10px;

width: 16px;

}

.body .register .group .i-pwd,.body .login .group .i-pwd{

background: url(../Images/i\_pwd.jpg) no-repeat scroll 0 0 transparent;

height: 19px;

top: 10px;

width: 14px;

}

.body .register .group .i-phone,.body .register .login .i-phone{

background: url(../Images/i\_phone.jpg) no-repeat scroll 0 0 transparent;

height: 21px;

top: 10px;

width: 14px;

}

.body .register .group .input-error{

font-size:12px;

color: #e4393c;

display: inline-block;

line-height: 32px;

height: 32px;

width: 260px;

padding: 0 5px;

background: #FFEBEB;

border: 1px solid #ffbdbe;

}

.body .login .group .input-error{

font-size:10px;

position: absolute;

color: #e4393c;

background: #FFEBEB;

border: 1px solid #ffbdbe;

display: block;

z-index: 10;

height: 15px;

width: 244px;

line-height: 15px;

left: 104px;

}

.body .register .group .checkcode,.body .login .group .checkcode{

position:absolute;

margin:-20px 0 0 5px;

}

.body .register .group .submit,.body .login .group .submit{

background-color: #e4393c;

padding:8px 20px;

width:246px;

color: white;

text-align: center;

border:1px solid #e4393c;

}

.body .more{

padding:20px;

}

.body .login .go-register{

position: absolute;

right:0px;

bottom:0px;

}

.body .login .go-register a{

line-height: 32px;

text-align: center;

font-size: 14px;

background: #7cbe56;

width: 115px;

height: 32px;

display: block;

color: #FFF;

}

.pg-footer{

margin:20px 0;

color: #555;

}

/\*公共开始\*/

body {

margin: 0 auto;

font-family: Arial;

\_font-family: 宋体,Arial;

font-size: 12px;

}

body, dl, dt, dd, ul, ol, li, h1, h2, h3, h4, h5, h6, pre, code, form, fieldset, legend, input, button, textarea, p, blockquote, th, td, figure, div {

margin: 0;

padding: 0;

}

ol, ul, li {

list-style: none;

}

a{

cursor:pointer;

text-decoration:none;

}

/\*a:hover{

color: #F60 !important;

text-decoration: underline;

}\*/

img{

border:none;

border-width:0px;

}

table{

border-collapse: collapse;

border-spacing: 0;

}

.red{

color: #c00 !important;

}

.m8{

margin:8px;

}

.mg20{

margin:20px;

}

.mt0{

margin-top:0px !important;

}

.mt10{

margin-top:10px;

}

.mt20{

margin-top:20px;

}

.mt30{

margin-top:30px !important;

}

.mr5{

margin-right:5px;

}

.ml5{

margin-left:5px;

}

.ml10{

margin-left:10px;

}

.mb0{

margin-bottom:0px !important;

}

.mb20{

margin-bottom:20px;

}

.mb10{

margin-bottom:10px;

}

.pd10{

padding:10px !important;

}

.pt18{

padding-top:18px;

}

.pt20{

padding-top:20px;

}

.pb20{

padding-bottom:20px;

}

.nbr{

border-right:0px;

}

.font12{

font-size:12px !important;

}

.font13{

font-size:13px !important;

}

.font14{

font-size:14px;

}

.font16{

font-size:16px;

}

.bold{

font-weight:bold;

}

.left{

float:left;

}

.right{

float:right;

}

.hide{

display:none;

}

.show{

display:table;

}

.clearfix{

clear:both;

}

.clearfix:after {

content: ".";

display: block;

height: 0;

clear: both;

visibility: hidden;

}

\* html .clearfix {zoom: 1;}

.container{

width:1190px;

margin-left:auto;

margin-right:auto;

}

.narrow{

width:980px !important;

margin-left:auto;

margin-right:auto;

}

.account-container{

width:980px;

margin-left:auto;

margin-right:auto;

}

.group-box-1 .title{

height: 33px;

line-height: 33px;

border: 1px solid #DDD;

background: #f5f5f5;

padding-top: 0;

padding-left: 0;

}

.group-box-1 .title .title-font{

display: inline-block;

font-size: 14px;

font-family: 'Microsoft Yahei','SimHei';

font-weight: bold;

color: #333;

padding-left: 10px;

}

.group-box-1 .body {

border: 1px solid #e4e4e4;

border-top: none;

}

.tab-menu-box1 {

border: 1px solid #ddd;

margin-bottom: 20px;

}

.tab-menu-box1 .menu {

line-height: 33px;

height: 33px;

background-color: #f5f5f5;

}

.tab-menu-box1 .content {

min-height: 100px;

border-top: 1px solid #ddd;

background-color: white;

}

.tab-menu-box1 .menu ul {

padding: 0;

margin: 0;

list-style: none;

/\*position: absolute;\*/

}

.tab-menu-box1 .menu ul li {

position: relative;

float: left;

font-size: 14px;

font-family: 'Microsoft Yahei','SimHei';

text-align: center;

font-size: 14px;

font-weight: bold;

border-right: 1px solid #ddd;

padding: 0 18px;

cursor: pointer;

}

.tab-menu-box1 .menu ul li:hover {

color: #c9033b;

}

.tab-menu-box1 .menu .more {

float: right;

font-size: 12px;

padding-right: 10px;

font-family: "宋体";

color: #666;

text-decoration: none;

}

.tab-menu-box1 .menu a:hover {

color: #f60 !important;

text-decoration: underline;

}

.tab-menu-box1 .menu .current {

margin-top: -1px;

color: #c9033b;

background: #fff;

height: 33px;

border-top: 2px solid #c9033b;

z-index: 10;

}

.tab-menu-box-2 .float-title {

display: none;

top: 0px;

position: fixed;

z-index: 50;

}

.tab-menu-box-2 .title {

width: 890px;

border-bottom: 2px solid #b20101;

border-left: 1px solid #e1e1e1;

clear: both;

height: 32px;

}

.tab-menu-box-2 .title a {

float: left;

width: 107px;

height: 31px;

line-height: 31px;

font-size: 14px;

font-weight: bold;

text-align: center;

border-top: 1px solid #e1e1e1;

border-right: 1px solid #e1e1e1;

background: url(../images/bg4.png?3) 0 -308px repeat-x;

text-decoration: none;

color: #333;

cursor: pointer;

}

.tab-menu-box-2 .title a:hover {

background-position: -26px -271px;

text-decoration: none;

color: #fff;

}

.tab-menu-box-2 .content {

min-height: 100px;

background-color: white;

}

.tab-menu-box3 {

border: 1px solid #ddd;

}

.tab-menu-box3 .menu {

line-height: 33px;

height: 33px;

background-color: #f5f5f5;

}

.tab-menu-box3 .content {

height: 214px;

border-top: 1px solid #ddd;

background-color: white;

}

.tab-menu-box3 .menu ul {

padding: 0;

margin: 0;

list-style: none;

/\*position: absolute;\*/

}

.tab-menu-box3 .menu ul li {

position: relative;

float: left;

font-size: 14px;

font-family: 'Microsoft Yahei','SimHei';

text-align: center;

font-size: 14px;

width:50%;

cursor: pointer;

}

.tab-menu-box3 .menu ul li:hover {

color: #c9033b;

}

.tab-menu-box3 .menu .more {

float: right;

font-size: 12px;

padding-right: 10px;

font-family: "宋体";

color: #666;

text-decoration: none;

}

.tab-menu-box3 .menu a:hover {

color: #f60 !important;

text-decoration: underline;

font-weight: bold;

}

.tab-menu-box3 .menu .current {

margin-top: -1px;

color: #c9033b;

background: #fff;

height: 33px;

border-top: 2px solid #c9033b;

z-index: 10;

font-weight: bold;

}

.quantity-bg{

height:20px;

width: 77px;

border: 1px solid #999;

}

.quantity-bg .minus,.quantity-bg .plus{

height:20px;

width:20px;

line-height:20px;

text-align:center;

vertical-align:middle;

}

.quantity-bg input{

height:20px;

width:35px;

border:0px;

border-left:1px solid #999;

border-right:1px solid #999;

}

/\*公共结束\*/

后续还有更新版本...