最近写代码很注意消除重复与抽象,在一个项目中遇到一个场景,需要定义几个不同类型的容器.略一思考决定使用模板做一个ItemTable.开始只定义了一个模板参数.非常顺利完成了90%的代码.

但是模板类型虽然都有名字与ID但是取得的方法却不尽相同.

于是使用了Traits来对ItemType的这一差异进行包装. 最终代码如下.

template<typename ItemType,typename ItemTraits>

class ItemTable

{

public:

typedef boost::shared\_ptr<ItemType> ItemPtr;

typedef typename std::map<std::string, ItemPtr>::iterator iterator;

void AddItem(ItemPtr apItem)

{

std::string lItemID = ItemTraits::GetItemID(apItem);

if(m\_ItemMap.count(lItemID))

{

std::string Exception\_info\_str = "Add ";

Exception\_info\_str +=ItemTraits::GetItemTypeName();

Exception\_info\_str += " Already Exist ItemID=";

Exception\_info\_str += lItemID;

throw std::exception(Exception\_info\_str.c\_str());

}

else

{

m\_ItemMap.insert(make\_pair(lItemID,apItem));

}

}

ItemPtr GetItem( const std::string& aItemID )

{

if(m\_ItemMap.count(aItemID))

{

return m\_ItemMap[aItemID];

}

else

{

return ItemPtr();

}

}

public:

iterator begin(){return m\_ItemMap.begin();};

iterator end(){return m\_ItemMap.end();};

private:

std::map<std::string, ItemPtr> m\_ItemMap;

};

结论是Traits可以将类型的不同接口的特性萃取到相同的接口之下 ,方便模板使用