# Product price （ProductDetail页面）

如果有DisplayPrices权限，则 model.ProductPrice.HidePrices = false;

否则

model.ProductPrice.HidePrices = true;

model.ProductPrice.OldPrice = null;

model.ProductPrice.Price = null;

当product不是CustomerEntersPrice或CallForPrice时，会计算价格：

## 计算价格

\_taxService.GetProductPrice 计算出价格

数据库存储的价格货币表示currencySettings.PrimaryStoreCurrencyId (美元)

1. 先计算出按照数据库输入的价格（oldPriceBase, 有折扣的价格， 无折扣价格）
2. 再转换为当前workContext.WorkingCurrency价格

decimal taxRate;

decimal oldPrice**Base** = \_taxService.GetProductPrice(product, product.OldPrice, out taxRate);

decimal finalPriceWithoutDiscount**Base** = \_taxService.GetProductPrice(product, \_priceCalculationService.GetFinalPrice(product, \_workContext.CurrentCustomer, includeDiscounts: false), out taxRate);

decimal finalPriceWithDiscount**Base** = \_taxService.GetProductPrice(product, \_priceCalculationService.GetFinalPrice(product, \_workContext.CurrentCustomer, includeDiscounts: true), out taxRate);

decimal oldPrice = \_currencyService.ConvertFromPrimaryStoreCurrency(oldPriceBase, \_workContext.WorkingCurrency);

decimal finalPriceWithoutDiscount = \_currencyService.ConvertFromPrimaryStoreCurrency(finalPriceWithoutDiscountBase, \_workContext.WorkingCurrency);

decimal finalPriceWithDiscount = \_currencyService.ConvertFromPrimaryStoreCurrency(finalPriceWithDiscountBase, \_workContext.WorkingCurrency);

## taxService计算价格

如果当前价格已经包含了税（taxSettings.PricesIncludeTax 为True）， 则

WorkContext.TaxDisplayType 值 （IncludingTax 或 ExcludingTax）

如果taxSettings AllowCustomersToSelectTaxDisplayType 且 当前用户！=null

找genericAttributeService 中 Customer.TaxDisplayTypeId 属性的值

否则返回taxSettings.TaxDisplayType（默认为ExcludingTax）

//当前价格已经包含税了A value indicating whether price already includes tax

bool priceIncludesTax = **\_taxSettings.PricesIncludeTax**; // False

// 价格展示是 A value indicating whether calculated price should include tax

bool includingTax = \_workContext.TaxDisplayType == TaxDisplayType.IncludingTax;

int taxCategoryId = 0;

public virtual decimal GetProductPrice(Product product, int taxCategoryId,

decimal price, bool includingTax, Customer customer,

bool priceIncludesTax, out decimal taxRate)

{

//no need to calculate tax rate if passed "price" is 0

if (price == decimal.Zero)

{

taxRate = decimal.Zero;

return taxRate;

}

bool isTaxable;

GetTaxRate(product, taxCategoryId, customer, price, out taxRate, out isTaxable)

if (priceIncludesTax)

{

//"price" already includes tax

if (includingTax)

{

//we should calculate price WITH tax

if (!isTaxable)

{

//but our request is not taxable

//hence we should calculate price WITHOUT tax

price = CalculatePrice(price, taxRate, false);

}

}

else

{

//we should calculate price WITHOUT tax

price = CalculatePrice(price, taxRate, false);

}

}

else

{

//"price" doesn't include tax

if (includingTax)

{

//we should calculate price WITH tax

//do it only when price is taxable

if (isTaxable)

{

price = CalculatePrice(price, taxRate, true);

}

}

}

if (!isTaxable)

{

//we return 0% tax rate in case a request is not taxable

taxRate = decimal.Zero;

}

//allowed to support negative price adjustments

//if (price < decimal.Zero)

// price = decimal.Zero;

return price;  
}

### VAT(增值税) 计算

从Plugin中找到TaxProvider来计算税率是多少 （LoadActiveTaxProvider() ）

protected virtual void GetTaxRate(Product product, int taxCategoryId,

Customer customer, decimal price, out decimal taxRate, out bool isTaxable)

{

taxRate = decimal.Zero;

isTaxable = true;

//active tax provider

var activeTaxProvider = LoadActiveTaxProvider();

if (activeTaxProvider == null)

return;

//tax request

var calculateTaxRequest = CreateCalculateTaxRequest(product, taxCategoryId, customer, price);

//tax exempt

if (IsTaxExempt(product, calculateTaxRequest.Customer))

{

isTaxable = false;

}

//make EU VAT exempt validation (the European Union Value Added Tax)

if (isTaxable &&

\_taxSettings.EuVatEnabled &&

IsVatExempt(calculateTaxRequest.Address, calculateTaxRequest.Customer))

{

//VAT is not chargeable

isTaxable = false;

}

//get tax rate

var calculateTaxResult = activeTaxProvider.GetTaxRate(calculateTaxRequest);

if (calculateTaxResult.Success)

{

//ensure that tax is equal or greater than zero

if (calculateTaxResult.TaxRate < decimal.Zero)

calculateTaxResult.TaxRate = decimal.Zero;

taxRate = calculateTaxResult.TaxRate;

}

else

if (\_taxSettings.LogErrors)

{

foreach (var error in calculateTaxResult.Errors)

{

\_logger.Error(string.Format("{0} - {1}", activeTaxProvider.PluginDescriptor.FriendlyName, error), null, customer);

}

}

}

TaxService.CreateCalculateTaxRequest

VAT 可以基于BillingAddress， ShippingAddress， DefaultAddress计算

默认 taxSettings.**TaxBasedOn中为**BillingAddress

如果EU VAT Eupore Union Value Added Tax 启用：

//new EU VAT rules starting January 1st 2015

//find more info at http://ec.europa.eu/taxation\_customs/taxation/vat/how\_vat\_works/telecom/index\_en.htm#new\_rules

//EU VAT enabled?

if (\_taxSettings.EuVatEnabled)

{

//telecommunications, broadcasting and electronic services?

if (product != null && product.IsTelecommunicationsOrBroadcastingOrElectronicServices)

{

//January 1st 2015 passed?

if (DateTime.UtcNow > new DateTime(2015, 1, 1, 0, 0, 0, DateTimeKind.Utc))

{

//Europe Union consumer?

if (IsEuConsumer(customer))

{

//We must charge VAT in the EU country where the customer belongs (not where the business is based)

basedOn = TaxBasedOn.BillingAddress;

}

}

}

}

### 产品是否免税

当前用户是否有免税的角色

或者当前Product 的IsTaxExempt=true

public virtual bool IsTaxExempt(Product product, Customer customer)

{

if (customer != null)

{

if (customer.IsTaxExempt)

return true;

if (customer.CustomerRoles.Where(cr => cr.Active).Any(cr => cr.TaxExempt))

return true;

}

if (product == null)

{

return false;

}

if (product.IsTaxExempt)

{

return true;

}

return false;

}

### 根据税率计算产品价格 **TaxService.cs**

**taxSettings.PricesIncludeTax 为True -》 当前价格已经包含税** （默认为False）

当前价格已经包含税， 实际价格应该减去税 CalculatePrice(price, taxRate, false)

当前价格不包含税， 实际价格应该加上税 CalculatePrice(price, taxRate, true)

protected virtual decimal CalculatePrice(decimal price, decimal percent, bool increase)

{

if (percent == decimal.Zero)

return price;

decimal result;

if (increase)

{

result = price \* (1 + percent / 100);

}

else

{

result = price - (price) / (100 + percent) \* percent;

}

return result;

}

### PriceCalculationService计算产品价格，includeDiscounts参数可以计算有没有折扣下的价格：

public virtual decimal GetFinalPrice(Product product,

Customer customer,

decimal? overriddenProductPrice, // null

decimal additionalCharge, // 默认 decimal.Zero,

bool includeDiscounts, // 是否包含discount

int quantity, // 默认是1

DateTime? rentalStartDate, // 默认null

DateTime? rentalEndDate, // 默认null

out decimal discountAmount,

out List<Discount> appliedDiscounts)

计算价格时

如果overriddenProductPrice有值, price按overriddenProductPrice的值计算

如果当前时间在specialPrice时间之间， price按specialPrice的值计算

如果product有TierPrices， 找出Math.Min(price, tierPrice.Value)

//additional charge

price = price + additionalCharge;

// rental products

// 如果产品IsRental的, 如果按天计算（RentalPricePeriod.Days ）

// var totalDaysToRent = Math.Max((endDate - startDate).TotalDays, 1);

// int configuredPeriodDays = product.RentalPriceLength;

// totalPeriods =

// Convert.ToInt32(Math.Ceiling(totalDaysToRent/configuredPeriodDays));

// 价格乘以多少个RentalPriceLength就是最终价格

if (product.IsRental)

if (rentalStartDate.HasValue && rentalEndDate.HasValue)

price=price\*product.GetRentalPeriods(rentalStartDate.Value, rentalEndDate.Value);

// 计算出折扣有多少 =》 最终价格就是再减去折扣

if (includeDiscounts)

{

//discount

List<Discount> tmpAppliedDiscounts;

decimal tmpDiscountAmount = GetDiscountAmount(product, customer, price, out tmpAppliedDiscounts);

price = price - tmpDiscountAmount;

// 。。。  
}

if (price < decimal.Zero)

price = decimal.Zero;

result.Price = price;

## 货币转换： 把价格转换为当前store的currency

**可以根据Rate为1的货币为基数转换为其他货币的价格**

**如美元的Rate为1**

**如人民币的Rate为6.1100**

**表示1美元等于6.1100人民币**

**Name Rate**

**en-US 1.0000**

**ja-jp 109.2700**

**zh-cn 6.1100**

**比如把X日元转换为人民币， （必须以美元为基础根据Rate来换算）**

**先把日元转换为美元 （**ConvertToPrimaryExchangeRateCurrency**） X / 109.2700**

**再把美元转换为人民币 （**ConvertFromPrimaryExchangeRateCurrency）X \* **6.1100**

Libraries\Nop.Services\Directory\CurrencyService.cs

约定（默认情况）：

**currencySettings.PrimaryStoreCurrencyId 默认值为1**

**Currency表中Id为1的Currency是“US Dollar”， Rate为1.0000 货币都以美元为基础转换**

public virtual decimal ConvertFromPrimaryStoreCurrency(decimal amount, Currency targetCurrencyCode)

{

var primaryStoreCurrency = GetCurrencyById(\_currencySettings.PrimaryStoreCurrencyId);

var result = **ConvertCurrency**(amount, primaryStoreCurrency, targetCurrencyCode);

return result;

}

public virtual decimal ConvertCurrency(decimal amount, Currency sourceCurrencyCode, Currency targetCurrencyCode)

{

if (sourceCurrencyCode == null)

throw new ArgumentNullException("sourceCurrencyCode");

if (targetCurrencyCode == null)

throw new ArgumentNullException("targetCurrencyCode");

decimal result = amount;

if (sourceCurrencyCode.Id == targetCurrencyCode.Id)

return result;

if (result != decimal.Zero && sourceCurrencyCode.Id != targetCurrencyCode.Id)

{

result = ConvertToPrimaryExchangeRateCurrency(result, sourceCurrencyCode);

result = ConvertFromPrimaryExchangeRateCurrency(result, targetCurrencyCode);

}

return result;

}

## 格式话价格priceFormatter

\_priceFormatter.FormatPrice(minimumCustomerEnteredPrice, false, false)

Libraries\Nop.Services\Catalog\PriceFormatter.cs

public virtual string FormatPrice(decimal price, bool showCurrency, bool showTax)

{

bool priceIncludesTax = \_workContext.TaxDisplayType == TaxDisplayType.IncludingTax;

return FormatPrice(price, showCurrency, \_workContext.WorkingCurrency, \_workContext.WorkingLanguage, priceIncludesTax, showTax);

}

public virtual string FormatPrice(decimal price, bool showCurrency,

Currency targetCurrency, Language language, bool priceIncludesTax, bool showTax)

{

//we should round it no matter of "ShoppingCartSettings.RoundPricesDuringCalculation" setting

price = RoundingHelper.RoundPrice(price);

string currencyString = GetCurrencyString(price, showCurrency, targetCurrency);

if (showTax)

{

//show tax suffix

string formatStr;

if (priceIncludesTax)

{

formatStr = \_localizationService.GetResource("Products.InclTaxSuffix", language.Id, false);

if (String.IsNullOrEmpty(formatStr))

formatStr = "{0} incl tax";

}

else

{

formatStr = \_localizationService.GetResource("Products.ExclTaxSuffix", language.Id, false);

if (String.IsNullOrEmpty(formatStr))

formatStr = "{0} excl tax";

}

return string.Format(formatStr, currencyString);

}

return currencyString;

}

# 折扣

DiscountType 有

AssignedToOrderTotal

AssignedToSkus

AssignedToCategories

AssignedToCategories

AssignedToManufacturers

AssignedToShipping

AssignedToOrderSubTotal

DiscountLimitationType 有： （从discountUsageHistory来看当前discount使用多少次）

Unlimited 无限次数使用

NTimesOnly 使用n次

NTimesPerCustomer 每个用户使用n次

折扣IsCumulative表示是否可以累积

## 验证Discount是否合法

可以使用的Discount： DiscountService.cs

ValidateDiscount(Discount discount, Customer customer)

//couponCodeToValidate=customer.GetAttribute<string>(SystemCustomerAttributeNames.// DiscountCouponCode, \_genericAttributeService);

ValidateDiscount(Discount discount, Customer customer, string couponCodeToValidate)

验证返回结果为 DiscountValidationResult { bool IsValid ， string UserError }

验证过程

// 1. check coupon code

if (discount.RequiresCouponCode)

{

if (String.IsNullOrEmpty(discount.CouponCode))

return result;

if (!discount.CouponCode.Equals(couponCodeToValidate, StringComparison.InvariantCultureIgnoreCase))

return result;

}

// 2. 当购物车里有gift cards的时候AssignedToOrderSubTotal， AssignedToOrderTotal折扣// 不能使用否则就可以用折扣来购买礼品卡 Otherwise, this customer can purchase gift // cards with discount and get more than paid ("free money").

if (discount.DiscountType == DiscountType.AssignedToOrderSubTotal ||

discount.DiscountType == DiscountType.AssignedToOrderTotal)

var hasGiftCards = cart.Any(x => x.Product.IsGiftCard);

if (hasGiftCards)

{

result.UserError = T("ShoppingCart.Discount.CannotBeUsedWithGiftCards");

return result;

}

//3. check date range

当前时间在 discount.StartDateUtc， discount.EndDateUtc之间

//4. discount limitation 验证使用次数限制

// GetAllDiscountUsageHistory第二个参数为customerid, 第三个参数为orderId

switch (discount.DiscountLimitation)

{

case DiscountLimitationType.NTimesOnly:

{

var usedTimes = GetAllDiscountUsageHistory(discount.Id, null, null, 0, 1).TotalCount;

if (usedTimes >= discount.LimitationTimes)

return result;

}

break;

case DiscountLimitationType.NTimesPerCustomer:

{

if (customer.IsRegistered())

{

var usedTimes = GetAllDiscountUsageHistory(discount.Id, customer.Id, null, 0, 1).TotalCount;

if (usedTimes >= discount.LimitationTimes)

{

result.UserError = \_localizationService.GetResource("ShoppingCart.Discount.CannotBeUsedAnymore");

return result;

}

}

}

break;

case DiscountLimitationType.Unlimited:

default:

break;

}

// 5. discount requirements

// 每个Discount对应多个DiscountRequirement， DiscountRequirementRuleSystemName属性为// 插件的systemname

DiscountRequirement {

public int DiscountId { get; set; }

public string DiscountRequirementRuleSystemName { get; set; }

}

如果能找到对应DiscountRequirementRuleSystemName的插件， 调用插件的CheckRequirement放发来验证

foreach (var req in requirements)

{

//load a plugin

var requirementRulePlugin = LoadDiscountRequirementRuleBySystemName(r

if (requirementRulePlugin == null)

continue;

if (!\_pluginFinder.AuthenticateStore(requirementRulePlugin.PluginDesc

continue;

var ruleRequest = new DiscountRequirementValidationRequest

{

DiscountRequirementId = req.Id,

Customer = customer,

Store = \_storeContext.CurrentStore

};

var ruleResult = requirementRulePlugin.CheckRequirement(ruleRequest);

if (!ruleResult.IsValid)

{

result.UserError = ruleResult.UserError;

return result;

}

}

// 如果以上步骤没有问题， 返回该Discount可以使用

result.IsValid = true;

return result;

## 由Discount得到折扣的值（可以便宜多少钱）

// Libraries\Nop.Services\Discounts\DiscountExtensions.cs

折扣分为使用百分比和固定折扣

使用百分比时要验证折扣是否小于最大折扣MaximumDiscountAmount

public static decimal GetDiscountAmount(this Discount discount, decimal amount)

{

if (discount == null)

throw new ArgumentNullException("discount");

//calculate discount amount

decimal result;

if (discount.**UsePercentage**)

result = (decimal)((((float)amount) \* ((float)discount.DiscountPercentage)) / 100f);

else

result = discount.DiscountAmount;

//validate maximum disocunt amount

if (discount.UsePercentage &&

discount.MaximumDiscountAmount.HasValue &&

result > discount.MaximumDiscountAmount.Value)

result = discount.MaximumDiscountAmount.Value;

if (result < decimal.Zero)

result = decimal.Zero;

return result;

}

## 找到product的最大折扣

**PriceCalculationSErvice.cs**

找到product所有的折扣， 并返回最大折扣值

GetAllowedDiscounts， GetPreferredDiscount是两个重要的方法

/// <returns>Discount amount</returns>

protected virtual decimal GetDiscountAmount(Product product,

Customer customer,

decimal productPriceWithoutDiscount,

out List<Discount> appliedDiscounts)

{

if (product == null)

throw new ArgumentNullException("product");

appliedDiscounts = null;

decimal appliedDiscountAmount = decimal.Zero;

//we don't apply discounts to products with price entered by a customer

if (product.CustomerEntersPrice)

return appliedDiscountAmount;

//discounts are disabled

if (\_catalogSettings.**IgnoreDiscounts**)

return appliedDiscountAmount;

var allowedDiscounts = GetAllowedDiscounts(product, customer);

//no discounts

if (!allowedDiscounts.Any())

return appliedDiscountAmount;

/// Get preferred discount (with maximum discount value)

appliedDiscounts = allowedDiscounts.GetPreferredDiscount(productPriceWithoutDiscount, out appliedDiscountAmount);

return appliedDiscountAmount;

}

### 找product所有的折扣： GetAllowedDiscounts

Nop.Services\Catalog\PriceCalculationService.cs

IList<Discount> GetAllowedDiscounts(Product product, Customer customer)

**返回**

该Product下所有Valid的折扣

+该Product 所在Category下Valid的折扣 （GetAllowedDiscountsAppliedToCategories）

+该Product 所在Manufacture下Valid的折扣 (GetAllowedDiscountsAppliedToManufacturers)

Product下所有Valid的折扣

foreach (var discount in product.AppliedDiscounts)

{

if (\_discountService.ValidateDiscount(discount, customer).IsValid &&

discount.DiscountType == DiscountType.**AssignedToSkus** &&

!allowedDiscounts.ContainsDiscount(discount))

allowedDiscounts.Add(discount);

}

Product所在Manufacture Valid的折扣

GetAllowedDiscountsAppliedToManufacturers

foreach(var discount in \_discountService.GetAllDiscounts(

DiscountType.AssignedToManufacturers))

{

// 返回所有AssignedToManufacturers 的Discount的Id

var appliedToManufacturerIds = \_cacheManager.Get(cacheKey,

() => discount.AppliedToManufacturers.Select(x => x.Id).ToList());

if (appliedToManufacturerIds.Any())

{

// 找到product在的manufacturerIds

var manufacturerIds = \_manufacturerService

.GetProductManufacturersByProductId(product.Id)

.Select(x => x.ManufacturerId)

.ToList());

foreach (var id in manufacturerIds)

{

if (appliedToManufacturerIds.Contains(id))

{

if (\_discountService.ValidateDiscount(discount, customer).IsValid &&

discount.DiscountType == DiscountType.**AssignedToManufacturers** &&

!allowedDiscounts.ContainsDiscount(discount))

allowedDiscounts.Add(discount);

}

}

}

}

### 得到折扣最大的折扣（便宜最多的）GetPreferredDiscount

先选出折扣最大的Discount （不可累积的折扣和可以累积的折扣单个折扣集合）

把可以累积（IsCumulative）的折扣值全部加起来

如果可累积折扣总值大于折扣单个使用的值， 返回可以累积折扣的集合

否则返回折扣最大Discount集合

/// <summary>

/// Get preferred discount (with maximum discount value)

/// </summary>

/// <param name="discounts">A list of discounts to check</param>

/// <param name="amount">Amount (initial value)</param>

/// <param name="discountAmount">Discount amount</param>

/// <returns>Preferred discount</returns>

public static List<Discount> GetPreferredDiscount(this IList<Discount> discounts,

decimal amount, out decimal discountAmount)

{

if (discounts == null)

throw new ArgumentNullException("discounts");

var result = new List<Discount>();

discountAmount = decimal.Zero;

if (!discounts.Any())

return result;

//first we check simple discounts

foreach (var discount in discounts)

{

decimal currentDiscountValue = discount.GetDiscountAmount(amount);

if (currentDiscountValue > discountAmount)

{

discountAmount = currentDiscountValue;

result.Clear();

result.Add(discount);

}

}

//now let's check cumulative discounts

//right now we calculate discount values based on the original amount value

//please keep it in mind if you're going to use discounts with "percentage"

var cumulativeDiscounts = discounts.Where(x => x.IsCumulative).OrderBy(x => x.Name).ToList();

if (cumulativeDiscounts.Count > 1)

{

var cumulativeDiscountAmount = cumulativeDiscounts.Sum(d => d.GetDiscountAmount(amount));

if (cumulativeDiscountAmount > discountAmount)

{

discountAmount = cumulativeDiscountAmount;

result.Clear();

result.AddRange(cumulativeDiscounts);

}

}

return result;

}

## 价格显示

decimal minimumCustomerEnteredPrice =

\_currencyService.**ConvertFromPrimaryStoreCurrency**(product.MinimumCustomerEnteredPrice, \_workContext.WorkingCurrency);

decimal maximumCustomerEnteredPrice =

\_currencyService.**ConvertFromPrimaryStoreCurrency**(product.MaximumCustomerEnteredPrice, \_workContext.WorkingCurrency);

/// currencySettings.PrimaryStoreCurrencyId安装后值是1, en-US的currency

public virtual decimal **ConvertFromPrimaryStoreCurrency**(decimal amount, Currency targetCurrencyCode)

{

var primaryStoreCurrency = GetCurrencyById(\_currencySettings.PrimaryStoreCurrencyId);

var result = **ConvertCurrency**(amount, primaryStoreCurrency, targetCurrencyCode);

return result;

}

/// <summary>

/// Converts currency source和target不一样再实行转换

/// 因为currency表保存的Rate都是相对于PrimaryCurrency的， 所以需要先转换为primary的值然后转换为target

/// ConvertToPrimaryExchangeRateCurrency 先转换为PrimaryCurrency

/// ConvertFromPrimaryExchangeRateCurrency PrimaryCurrency转换为target

/// </summary>

public virtual decimal ConvertCurrency(decimal amount, Currency sourceCurrencyCode, Currency targetCurrencyCode)

{

if (sourceCurrencyCode == null)

throw new ArgumentNullException("sourceCurrencyCode");

if (targetCurrencyCode == null)

throw new ArgumentNullException("targetCurrencyCode");

decimal result = amount;

if (sourceCurrencyCode.Id == targetCurrencyCode.Id)

return result;

if (result != decimal.Zero && sourceCurrencyCode.Id != targetCurrencyCode.Id)

{

result = ConvertToPrimaryExchangeRateCurrency(result, sourceCurrencyCode);

result = ConvertFromPrimaryExchangeRateCurrency(result, targetCurrencyCode);

}

return result;

}