# Product Detail页面显示逻辑

// 用来显示某个product详情 或者编辑购物车中product页面

ProductController.ProductDetails(int productId, int updatecartitemid = 0)

## FeatureedProducts

IPagedList<Product> featuredProducts = null;

featuredProducts = \_productService.SearchProducts(

categoryIds: new List<int> { category.Id },

storeId: \_storeContext.CurrentStore.Id,

visibleIndividuallyOnly: true,

featuredProducts: true);

hasFeaturedProductsCache = featuredProducts.TotalCount > 0;

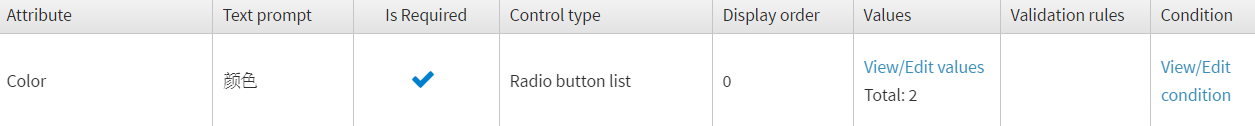
## 产品属性Product attributes

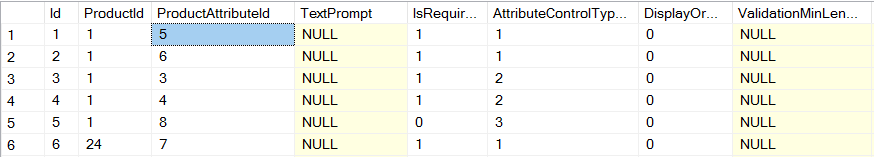
**dbo.ProductAttribute**： Custom TextColor 这些属性

**dbo.PredefinedProductAttributeValue 如果**属性有预设值， 在给product添加属性的时候会自动上这些属性到

Product\_ProductAttribute\_Mapping， ProductAttributeValue

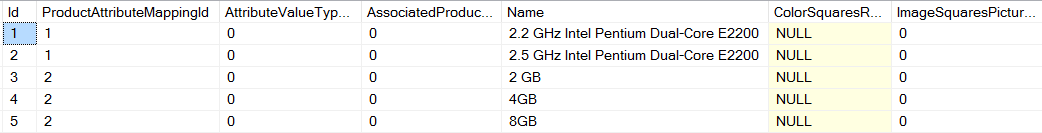
**dbo.Product\_ProductAttribute\_Mapping**： 保存product有哪些属性ProductId, AttributeId, 及属性的要求（IsRequired, AttributeControlType）





**dbo.ProductAttributeValue**: 产品属性对应的选项值

**AttributeControlType**不是TextBox MultilineTextbox Datepicker FileUpload时， 没有Value值



if (attribute.ShouldHaveValues())

{  
}

**ProductAttributeValue** 有属性AttributeValueTypeID， PriceAdjustment

public enum AttributeValueType{ Simple = 0, AssociatedToProduct = 10,}

如果是 Simple，价格增加值为PriceAdjustment

如果是 AssociatedToProduct， 价格增加值为AttributeValue **AssociatedProductId**属性对应铲平的价格

ProductAttributeValueModel 有属性 PriceAdjustment （string） 记录价格变动值

if (priceAdjustmentBase > decimal.Zero)

"+" + \_priceFormatter.FormatPrice(priceAdjustment, false, false);

else if (priceAdjustmentBase < decimal.Zero)

"-" + \_priceFormatter.FormatPrice(-priceAdjustment, false, false);

**得到产品的属性：**

IList<ProductAttributeMapping> productAttributeMapping =

\_productAttributeService.**GetProductAttributeMappingsByProductId**(product.Id);

public virtual IList<ProductAttributeMapping>

GetProductAttributeMappingsByProductId(int productId)

{

string key = string.Format(PRODUCTATTRIBUTEMAPPINGS\_ALL\_KEY, productId);

return \_cacheManager.Get(key, () =>

{

var query = from pam in \_productAttributeMappingRepository.Table

orderby pam.DisplayOrder

where pam.ProductId == productId

select pam;

var productAttributeMappings = query.ToList();

return productAttributeMappings;

});

}

如果该属性的AttributeControl需要值初始化， 在表ProductAttributeValue找

//AttributeControlType为TextBox,MultilineTextbox,Datepicker,FileUpload不需要值初始化

if (attribute.ShouldHaveValues())

{

//values

var attributeValues =

\_productAttributeService.**GetProductAttributeValues**(attribute.Id);

foreach (var attributeValue in attributeValues)

{

var valueModel = new ProductDetailsModel.ProductAttributeValueModel

{

Id = attributeValue.Id,

Name = attributeValue.GetLocalized(x => x.Name),

ColorSquaresRgb = attributeValue.ColorSquaresRgb, //used with "Color squares" attribute type

IsPreSelected = attributeValue.IsPreSelected

};

attributeModel.Values.Add(valueModel);

//display price if allowed

if (\_permissionService.Authorize(StandardPermissionProvider.DisplayPrices))

{

decimal taxRate;

decimal attributeValuePriceAdjustment =

\_priceCalculationService.GetProductAttributeValuePriceAdjustment(attributeValue);

decimal priceAdjustmentBase = \_taxService.GetProductPrice(product, attributeValuePriceAdjustment, out taxRate);

decimal priceAdjustment = \_currencyService.ConvertFromPrimaryStoreCurrency(priceAdjustmentBase, \_workContext.WorkingCurrency);

if (priceAdjustmentBase > decimal.Zero)

valueModel.PriceAdjustment = "+" + \_priceFormatter.FormatPrice(priceAdjustment, false, false);

else if (priceAdjustmentBase < decimal.Zero)

valueModel.PriceAdjustment = "-" + \_priceFormatter.FormatPrice(-priceAdjustment, false, false);

valueModel.PriceAdjustmentValue = priceAdjustment;

}

//"image square" picture (with with "image squares" attribute type only)

if (attributeValue.ImageSquaresPictureId > 0)

{

var productAttributeImageSquarePictureCacheKey = string.Format(ModelCacheEventConsumer.PRODUCTATTRIBUTE\_IMAGESQUARE\_PICTURE\_MODEL\_KEY,

attributeValue.ImageSquaresPictureId,

\_webHelper.IsCurrentConnectionSecured(),

\_storeContext.CurrentStore.Id);

valueModel.ImageSquaresPictureModel = \_cacheManager.Get(productAttributeImageSquarePictureCacheKey, () =>

{

var imageSquaresPicture = \_pictureService.GetPictureById(attributeValue.ImageSquaresPictureId);

if (imageSquaresPicture != null)

{

return new PictureModel

{

FullSizeImageUrl = \_pictureService.GetPictureUrl(imageSquaresPicture),

ImageUrl = \_pictureService.GetPictureUrl(imageSquaresPicture, \_mediaSettings.ImageSquarePictureSize)

};

}

return new PictureModel();

});

}

//picture of a product attribute value

valueModel.PictureId = attributeValue.PictureId;

}

}

public virtual IList<ProductAttributeValue> GetProductAttributeValues(int

productAttributeMappingId)

{

string key = string.Format(PRODUCTATTRIBUTEVALUES\_ALL\_KEY, productAttributeMappingId);

return \_cacheManager.Get(key, () =>

{

var query = from pav in \_productAttributeValueRepository.Table

orderby pav.DisplayOrder

where pav.ProductAttributeMappingId == productAttributeMappingId

select pav;

var productAttributeValues = query.ToList();

return productAttributeValues;

});

}

// 选择该属性的价格变动（比如绿色属性会多收5块钱）

// AttributeValueType 为Simple， 取PriceAdjustment 属性

// AttributeValueType 为AssociatedToProduct, 取AssociatedProductId属性对应产品的折// 扣后的最终价格

public virtual decimal GetProductAttributeValuePriceAdjustment(ProductAttributeValue value)

{

if (value == null)

throw new ArgumentNullException("value");

var adjustment = decimal.Zero;

switch (value.AttributeValueType)

{

case AttributeValueType.Simple:

{

//simple attribute

adjustment = value.PriceAdjustment;

}

break;

case AttributeValueType.AssociatedToProduct:

{

//bundled product

var associatedProduct = \_productService.GetProductById(value.AssociatedProductId);

if (associatedProduct != null)

{

adjustment = GetFinalPrice(associatedProduct, \_workContext.CurrentCustomer, includeDiscounts: true) \* value.Quantity;

}

}

break;

default:

break;

}

return adjustment;

}

## 产品属性值的保存productAttributeParser ？TODO

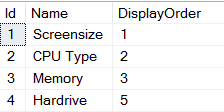
\_productAttributeParser.ParseValues(updatecartitem.AttributesXml, attribute.Id);

# Product specifications

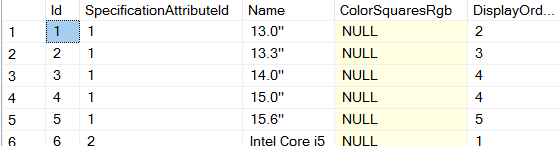
产品Attribute用来在购买的时候选择款式什么的

产品Specification用来显示产品，不能选择，只有展示功能

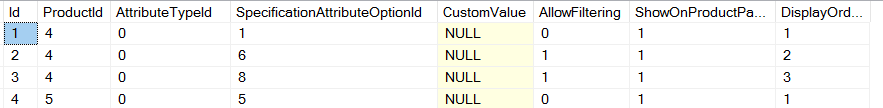
**SpecificationAttribute**



**SpecificationAttributeOption**



**Product\_SpecificationAttribute\_Mapping**



public enum SpecificationAttributeType

{

Option = 0,

CustomText = 10,

CustomHtmlText = 20,

Hyperlink = 30

}

1. Presentation\Nop.Web\Controllers\ProductController.cs -> PrepareProductDetailsPageModel

if (!isAssociatedProduct)

{

model.ProductSpecifications = this.PrepareProductSpecificationModel(\_workContext,

\_specificationAttributeService,\_cacheManager, product);

}

2. Presentation\Nop.Web\Extensions\ControllerExtensions.cs

specificationAttributeService.GetProductSpecificationAttributes(product.Id, 0, null, true) 先找到ProductId, ShowOnProductPage = true 按找DisplayOrder 排序的Product\_SpecificationAttribute\_Mapping

public static IList<ProductSpecificationModel> PrepareProductSpecificationModel(this Controller controller,

IWorkContext workContext,

ISpecificationAttributeService specificationAttributeService,

ICacheManager cacheManager,

Product product)

{

if (product == null)

throw new ArgumentNullException("product");

string cacheKey = string.Format(ModelCacheEventConsumer.PRODUCT\_SPECS\_MODEL\_KEY, product.Id, workContext.WorkingLanguage.Id);

return cacheManager.Get(cacheKey, () =>

specificationAttributeService.GetProductSpecificationAttributes(product.Id, 0, null, true)

.Select(psa =>

{

var m = new ProductSpecificationModel

{

SpecificationAttributeId = psa.SpecificationAttributeOption.SpecificationAttributeId,

SpecificationAttributeName = psa.SpecificationAttributeOption.SpecificationAttribute.GetLocalized(x => x.Name),

ColorSquaresRgb = psa.SpecificationAttributeOption.ColorSquaresRgb

};

switch (psa.AttributeType)

{

case SpecificationAttributeType.Option:

m.ValueRaw = HttpUtility.HtmlEncode(psa.SpecificationAttributeOption.GetLocalized(x => x.Name));

break;

case SpecificationAttributeType.CustomText:

m.ValueRaw = HttpUtility.HtmlEncode(psa.CustomValue);

break;

case SpecificationAttributeType.CustomHtmlText:

m.ValueRaw = psa.CustomValue;

break;

case SpecificationAttributeType.Hyperlink:

m.ValueRaw = string.Format("<a href='{0}' target='\_blank'>{0}</a>", psa.CustomValue);

break;

default:

break;

}

return m;

}).ToList()

);

}

## 产品的评论Product Review

如果 catalogSettings.ShowProductReviewsPerStore 为true （默认false）

只会选出当前store下的评论 （pr.StoreId == storeContext.CurrentStore.Id）来返回

ProductReviewOverviewModel { RatingSum， TotalReviews}

否则计算所有的评论

productReview.ProductId = product.Id;

productReview.AllowCustomerReviews = product.AllowCustomerReviews;

public static ProductReviewOverviewModel PrepareProductReviewOverviewModel(this Controller controller,

IStoreContext storeContext,

CatalogSettings catalogSettings,

ICacheManager cacheManager,

Product product)

{

ProductReviewOverviewModel productReview;

if (catalogSettings.ShowProductReviewsPerStore)

{

string cacheKey = string.Format(ModelCacheEventConsumer.PRODUCT\_REVIEWS\_MODEL\_KEY, product.Id, storeContext.CurrentStore.Id);

productReview = cacheManager.Get(cacheKey, () =>

{

return new ProductReviewOverviewModel

{

RatingSum = product.ProductReviews

.Where(pr => pr.IsApproved && pr.StoreId == storeContext.CurrentStore.Id)

.Sum(pr => pr.Rating),

TotalReviews = product

.ProductReviews

.Count(pr => pr.IsApproved && pr.StoreId == storeContext.CurrentStore.Id)

};

});

}

else

{

productReview = new ProductReviewOverviewModel()

{

RatingSum = product.ApprovedRatingSum,

TotalReviews = product.ApprovedTotalReviews

};

}

if (productReview != null)

{

productReview.ProductId = product.Id;

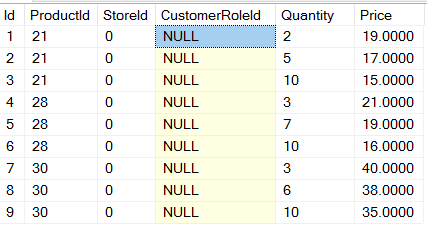
productReview.AllowCustomerReviews = product.AllowCustomerReviews;

}

return productReview;

}

## Tier prices 买多点的产品价格（意思相当于买的多了批发价）



if (product.HasTierPrices && \_permissionService.Authorize(StandardPermissionProvider.DisplayPrices))

{

model.TierPrices = product.TierPrices

.OrderBy(x => x.Quantity)

.ToList()

.FilterByStore(\_storeContext.CurrentStore.Id)

.FilterForCustomer(\_workContext.CurrentCustomer)

.RemoveDuplicatedQuantities()

.Select(tierPrice =>

{

var m = new ProductDetailsModel.TierPriceModel

{

Quantity = tierPrice.Quantity,

};

decimal taxRate;

decimal priceBase = \_taxService.GetProductPrice(product, \_priceCalculationService.GetFinalPrice(product, \_workContext.CurrentCustomer, decimal.Zero, \_catalogSettings.DisplayTierPricesWithDiscounts, tierPrice.Quantity), out taxRate);

decimal price = \_currencyService.ConvertFromPrimaryStoreCurrency(priceBase, \_workContext.WorkingCurrency);

m.Price = \_priceFormatter.FormatPrice(price, false, false);

return m;

})

.ToList();

}

## Associated products 关联产品

Product 有属性GroupedProductId （ProductType == ProductType.GroupedProduct）

找到该GroupedProductId = 该ProductId的产品

注意： PrepareProductDetailsPageModel第三个参数isAssociatedProduct为true， 这样就不会递归调用了（只调用当前proudct的属性， 不需要其他product的详细属性）

if (product.ProductType == ProductType.GroupedProduct)

{

//ensure no circular references

if (!isAssociatedProduct)

{

var associatedProducts = \_productService.GetAssociatedProducts(product.Id, \_storeContext.CurrentStore.Id);

foreach (var associatedProduct in associatedProducts)

model.AssociatedProducts.Add(PrepareProductDetailsPageModel(associatedProduct, null, true));

}

}

# Category 下的Product

//products

IList<int> alreadyFilteredSpecOptionIds = model.PagingFilteringContext.SpecificationFilter.GetAlreadyFilteredSpecOptionIds(\_webHelper);

IList<int> filterableSpecificationAttributeOptionIds;

var products = \_productService.SearchProducts(out filterableSpecificationAttributeOptionIds,

true,

categoryIds: categoryIds,

storeId: \_storeContext.CurrentStore.Id,

visibleIndividuallyOnly: true,

featuredProducts: \_catalogSettings.IncludeFeaturedProductsInNormalLists ? null : (bool?)false,

priceMin: minPriceConverted,

priceMax: maxPriceConverted,

filteredSpecs: alreadyFilteredSpecOptionIds,

orderBy: (ProductSortingEnum)command.OrderBy,

pageIndex: command.PageNumber - 1,

pageSize: command.PageSize);

model.Products = PrepareProductOverviewModels(products).ToList();

//specs

model.PagingFilteringContext.SpecificationFilter.PrepareSpecsFilters(alreadyFilteredSpecOptionIds,

filterableSpecificationAttributeOptionIds != null ? filterableSpecificationAttributeOptionIds.ToArray() : null,

\_specificationAttributeService,

\_webHelper,

\_workContext,

\_cacheManager);