ПРИЛОЖЕНИЕ А

(справочное)

Листинг кода класса TextFeedParser

public class TextFeedParser : IFeedParser

{

private static readonly Logger Logger = LogManager.GetCurrentClassLogger();

private readonly char \_delimiter;

public TextFeedParser(char delimiter)

{

\_delimiter = delimiter;

}

public IObservable<ProcessingResult<FeedItem>> Parse(TextReader reader, FeedItemScheme scheme)

{

if (reader == null)

{

throw new ArgumentNullException("reader");

}

if (scheme == null)

{

throw new ArgumentNullException("scheme");

}

return Observable.Create<ProcessingResult<FeedItem>>(

observer =>

{

Logger.Info("Starting '{0}'-delimited feed file parsing".f(\_delimiter));

ParseImpl(observer, reader, scheme);

Logger.Info("Feed file parsing finished.");

return Disposable.Empty;

});

}

private void ParseImpl(IObserver<ProcessingResult<FeedItem>> observer,

TextReader reader,

FeedItemScheme scheme)

{

var headerStr = ReadHeader(reader);

if (headerStr == null)

{

const string error = "Empty feed file. But expected header at least.";

OnFatalError(observer, error);

return;

}

var headerParsingResult = ParseHeader(headerStr, scheme);

if (!headerParsingResult.IsValid)

{

var error = String.Join(", ", headerParsingResult.Errors);

OnFatalError(observer, error);

return;

}

var header = headerParsingResult.Result;

ParseBody(observer, reader, scheme, header);

observer.OnCompleted();

}

private static string ReadHeader(TextReader reader)

{

string result;

do

{

result = reader.ReadLine();

} while (

result != null &&

String.IsNullOrWhiteSpace(result)

);

return result;

}

private ProcessingResult<ParsedHeader> ParseHeader(string headerStr, FeedItemScheme scheme)

{

Logger.Debug("Parsing feed file header.");

var fields = headerStr.Split(\_delimiter, StringSplitOptions.None)

.ToReadOnlyCollection();

var header = new ParsedHeader(fields);

var errors = ValidateHeader(header, scheme);

return ProcessingResult.Create(header, errors);

}

private static IEnumerable<string> ValidateHeader(ParsedHeader header, FeedItemScheme scheme)

{

var errors = new List<string>();

var expectedFields = scheme.FieldNames.ToHashSet();

var missingFields = expectedFields;

var excessFields = new List<string>();

foreach (var headerField in header.Fields)

{

if (!expectedFields.Contains(headerField))

{

excessFields.Add(headerField);

}

else

{

missingFields.Remove(headerField);

}

}

if (!missingFields.IsEmpty())

{

var errorBuilder = new StringBuilder("File header isn't fitting expected scheme.");

if (!missingFields.IsEmpty())

{

missingFields.Aggregate(

errorBuilder.Append(" Missing fields:"),

(acc, cur) => acc.AppendFormat("{0}, ", cur));

Logger.Debug(header.ToString());

}

var error = errorBuilder.ToString();

errors.Add(error);

}

return errors;

}

private void ParseBody(IObserver<ProcessingResult<FeedItem>> observer,

TextReader reader,

FeedItemScheme itemScheme,

ParsedHeader header)

{

Logger.Debug("Parsing feed file body.");

reader.AsEnumerable()

.WhereNot(String.IsNullOrWhiteSpace)

.ForEach(

line =>

{

Logger.Debug("Parsing next feed item.");

var lineParsingResult = ParseLine(line, header);

if (!lineParsingResult.IsValid)

{

var errors = lineParsingResult.Errors;

var parsedFeedItem = ProcessingResult.Create<FeedItem>(null, errors);

observer.OnNext(parsedFeedItem);

}

else

{

var parsedLine = lineParsingResult.Result;

var parsedFeedItem = CreateItem(parsedLine, itemScheme);

observer.OnNext(parsedFeedItem);

}

});

}

private ProcessingResult<ParsedLine> ParseLine(string line, ParsedHeader header)

{

var values = line.Split(\_delimiter, StringSplitOptions.None);

if (values.Length != header.FieldsCount)

{

var error =

(values.Length > header.FieldsCount

? ("Excess values in line '{0}'. " +

"Expected '{1}' values, but found '{2}' values.")

: ("Lack of values in line '{0}'. " +

"Expected '{1}' values, but found '{2}' values.")

).f(line,

header.FieldsCount,

values.Length);

return ProcessingResult.Create<ParsedLine>(null, new[] {error});

}

var fields = header.Fields

.Zip(values,

(name, value) =>

{

Logger.Trace("'{0}' field parsed.".f(name));

return new ParsedField(name, value);

})

.ToReadOnlyCollection();

var parsedLine = new ParsedLine(fields);

return ProcessingResult.Create(parsedLine);

}

private static ProcessingResult<FeedItem> CreateItem(ParsedLine line, FeedItemScheme scheme)

{

var errors = new List<string>();

var item = line.Fields

.AggregateSafe(

seed: new FeedItem(scheme),

func: (acc, field) =>

{

return scheme.IsFieldNameDefined(field.Name)

? acc.WithFieldValue(field.Name, field.Value.Trim())

: acc;

},

onError: (cur, ex) =>

{

var error = ("Error while setting feed item '{0}' " +

"field value. {1}").f(cur.Name,

ex.Message);

errors.Add(error);

Logger.InfoException(error, ex);

});

return ProcessingResult.Create(item, errors);

}

private static void OnFatalError(IObserver<ProcessingResult<FeedItem>> observer, string message)

{

var error = "Feed file parsing error: {0}".f(message);

observer.OnError(new FeedParsingException(error));

}

}

ПРИЛОЖЕНИЕ Б

(справочное)

Листинг кода класса AdminController

[ValidateInput(false), Admin]

public class AdminController : BaseController, IUpdateModel

{

private readonly ISiteService \_siteService;

private readonly ISortableGridManager \_gridManager;

public AdminController(

IOrchardServices services,

IShapeFactory shapeFactory,

ISortableGridManager gridManager,

ISiteService siteService)

{

Services = services;

\_siteService = siteService;

\_gridManager = gridManager;

T = NullLocalizer.Instance;

Shape = shapeFactory;

}

private dynamic Shape { get; set; }

public IOrchardServices Services { get; set; }

public Localizer T { get; set; }

public ActionResult Index(PagerParameters pagerParameters, SortParameters sortParameters)

{

if (!Services.Authorizer.Authorize(DealerPermissions.List, T("Not authorized to view list dealers.")))

{

return new HttpUnauthorizedResult();

}

var pager = new Pager(\_siteService.GetSiteSettings(), pagerParameters);

var dealersQuery = Services.ContentManager.Query<DealerPart, DealerPartRecord>();

var pagerShape = Shape.Pager(pager)

.TotalItemCount(dealersQuery.Count());

var dealers = \_gridManager.Sort(dealersQuery, sortParameters).Slice(pager.GetStartIndex(), pager.PageSize).ToList();

var model = new DealersListingViewModel

{

Dealers = dealers.Select(x => new DealerEntry { Model = x }).ToList(),

Pager = pagerShape

};

AddRouteData("sortBy", sortParameters.SortBy);

AddRouteData("sortDirection", sortParameters.SortDirection);

pagerShape.RouteData(ControllerContext.RouteData.Values);

return View(model);

}

[HttpGet]

public ActionResult Create()

{

if (!Services.Authorizer.Authorize(DealerPermissions.Create, T("Not authorized to create dealer.")))

{

return new HttpUnauthorizedResult();

}

var dealer = Services.ContentManager.New<DealerPart>("Dealer");

dynamic model = Services.ContentManager.BuildEditor(dealer);

return View(model);

}

[HttpPost, ActionName("Create")]

public ActionResult CreatePOST()

{

if (!Services.Authorizer.Authorize(DealerPermissions.Create, T("Not authorized to create dealer.")))

{

return new HttpUnauthorizedResult();

}

var dealer = Services.ContentManager.New<DealerPart>("Dealer");

dynamic model = Services.ContentManager.UpdateEditor(dealer, this);

if (!IsDealerNameUnique(dealer))

{

ModelState.AddModelError(T("The Dealer Name field must be unique.").ToString());

}

if (!ModelState.IsValid)

{

Services.TransactionManager.Cancel();

return View((object)model);

}

Services.ContentManager.Create(dealer);

Services.Notifier.Information(T("Dealer was successfully created."));

return RedirectToAction("Index");

}

[HttpGet]

public ActionResult Edit(int id)

{

if (!Services.Authorizer.Authorize(DealerPermissions.Edit, T("Not authorized to edit dealer.")))

{

return new HttpUnauthorizedResult();

}

var item = Services.ContentManager.Get<DealerPart>(id);

if (item == null)

{

return HttpNotFound();

}

var model = Services.ContentManager.BuildEditor(item);

return View((object)model);

}

[HttpPost, ActionName("Edit")]

public ActionResult EditPOST(int id)

{

if (!Services.Authorizer.Authorize(DealerPermissions.Edit, T("Not authorized to edit dealer.")))

{

return new HttpUnauthorizedResult();

}

var dealer = Services.ContentManager.Get<DealerPart>(id);

if (dealer == null)

{

return HttpNotFound();

}

var model = Services.ContentManager.UpdateEditor(dealer, this);

if (!IsDealerNameUnique(dealer))

{

ModelState.AddModelError(T("The Dealer Name field must be unique.").ToString());

}

if (!ModelState.IsValid)

{

Services.TransactionManager.Cancel();

return View((object)model);

}

Services.Notifier.Information(T("Dealer was successfully updated."));

return RedirectToAction("Index");

}

public ActionResult Delete(int id)

{

if (!Services.Authorizer.Authorize(DealerPermissions.Delete, T("Not authorized to delete dealer.")))

{

return new HttpUnauthorizedResult();

}

var dealer = Services.ContentManager.Get(id);

if (dealer == null)

{

return HttpNotFound();

}

Services.ContentManager.Remove(dealer);

if (!ModelState.IsValid)

{

Services.TransactionManager.Cancel();

return RedirectToAction("Index");

}

Services.Notifier.Information(T("Dealer was successfully deleted."));

int pageSize;

int.TryParse(HttpContext.Request.QueryString["pageSize"], out pageSize);

var countPages = GetCountPage(pageSize, Services.ContentManager.Query<DealerPart, DealerPartRecord>().Count());

return RedirectAfterDelete("Index", countPages, new RouteValueDictionary());

}

bool IUpdateModel.TryUpdateModel<TModel>(TModel model, string prefix, string[] includeProperties,

string[] excludeProperties)

{

return TryUpdateModel(model, prefix, includeProperties, excludeProperties);

}

void IUpdateModel.AddModelError(string key, LocalizedString errorMessage)

{

ModelState.AddModelError(key, errorMessage.ToString());

}

private bool IsDealerNameUnique(DealerPart dealer)

{

return !Services.ContentManager.Query<DealerPart, DealerPartRecord>()

.Where(m => m.DealerName == dealer.DealerName && m.Id != dealer.Id)

.Any();

}

}

ПРИЛОЖЕНИЕ В

(справочное)

Листинг кода класса CarListController

[Themed]

public class CarListController : Controller

{

#region Constants

private readonly char separator = ',';

#endregion

#region Fields

private readonly IOrchardServices \_services;

private readonly ICarListService \_carListService;

private readonly ICarService \_carService;

private readonly ISiteService \_siteService;

private readonly ICarTypeService \_carTypeService;

private readonly ISpecialService \_specialService;

private readonly IVideoService \_videoService;

private readonly ICarListFilterProvider \_carListFilterProvider;

private readonly IDealerSettingsService \_dealerSettingsService;

#endregion

#region Properties

private dynamic Shape { get; set; }

protected ILogger Logger { get; set; }

public Localizer T { get; set; }

#endregion

#region Controller

public CarListController(IOrchardServices services, ICarListService carListService, ICarService carService,

IShapeFactory shapeFactory,

ISiteService siteService, ICarTypeService carTypeService,

ISpecialService specialService, IVideoService videoService,

ICarListFilterProvider carListFilterProvider,

IDealerSettingsService dealerSettingsService)

{

\_dealerSettingsService = dealerSettingsService;

\_services = services;

\_carListService = carListService;

\_carService = carService;

\_siteService = siteService;

Logger = NullLogger.Instance;

Shape = shapeFactory;

T = NullLocalizer.Instance;

\_carTypeService = carTypeService;

\_specialService = specialService;

\_carListFilterProvider = carListFilterProvider;

\_videoService = videoService;

}

#endregion

#region Actions

[ValidateInput(false)]

public ActionResult Item(int carListId, CarListOptionsViewModel options, PagerParameters pageParameters)

{

var carListPart = \_services.ContentManager.Get(carListId, VersionOptions.Published).As<CarListPart>();

if (carListPart == null)

{

return HttpNotFound();

}

dynamic carList = \_services.ContentManager.BuildDisplay(carListPart);

var filterOptions = new CarFilterOptions

{

IsShowFilter = carListPart.IsShowFilter,

IsShowTrimFilter = carListPart.IsShowTrimFilter,

IsShowModelFilter = carListPart.IsShowModelFilter,

IsShowBodyStyleFilter = carListPart.IsShowBodyStyleFilter,

IsShowMakeFilter = carListPart.IsShowMakeFilter,

IsShowSortByFilter = carListPart.IsShowSortByFilter,

IsShowPriceFilter = carListPart.IsShowPriceFilter,

IsShowTypeFilter = carListPart.IsShowTypeFilter,

IsShowDealerFilter = carListPart.IsShowDealerFilter

};

var model = new CarIndexViewModel

{

CarListInfo = carList,

CarFilterOptions = filterOptions,

CarListId = carListId,

Options = options,

PageParams = pageParameters

};

return View(model);

}

public ActionResult Filter(int carListId, CarListOptionsViewModel options, PagerParameters pagerParameters)

{

var carListPart = \_services.ContentManager.Get(carListId, VersionOptions.Published).As<CarListPart>();

if (carListPart == null)

{

return HttpNotFound();

}

var carListName = carListPart.As<AutoroutePart>().Path;

var carFilter = \_carListFilterProvider.FindFilter(carListName);

var carRecordList = \_carService.GetRecordsWithoutSorting(options)

.Where(x => x.IsActive == true);

if (carFilter == null)

{

carRecordList = FilterCarsByListSettings(carRecordList, carListPart);

}

else

{

carRecordList = carFilter.Filter(carRecordList);

}

var filter = new CarFilter(carRecordList, \_carService, options);

var filterOptions = new CarFilterOptions

{

IsShowFilter = carListPart.IsShowFilter,

IsShowTrimFilter = carListPart.IsShowTrimFilter,

IsShowModelFilter = carListPart.IsShowModelFilter,

IsShowBodyStyleFilter = carListPart.IsShowBodyStyleFilter,

IsShowMakeFilter = carListPart.IsShowMakeFilter,

IsShowSortByFilter = carListPart.IsShowSortByFilter,

IsShowPriceFilter = carListPart.IsShowPriceFilter,

IsShowDealerFilter = carListPart.IsShowDealerFilter,

IsShowTypeFilter = carListPart.IsShowTypeFilter

};

var viewModel = new CarFilterViewModel

{

Filter = filter,

Options = filterOptions,

Page = pagerParameters.Page.HasValue?pagerParameters.Page.Value:1

};

return PartialView(viewModel);

}

private IQueryable<CarRecord> FilterCarsByListSettings(IQueryable<CarRecord> carRecordList, CarListPart carListPart)

{

if (carListPart.IsCertified.HasValue)

{

carRecordList = carListPart.IsCertified.Value

? carRecordList.Where(x => x.IsCertified.Value)

: carRecordList.Where(x => !x.IsCertified.Value);

}

if (carListPart.IsSpecial.HasValue)

{

carRecordList = carListPart.IsSpecial.Value

? carRecordList.Where(x => x.IsSpecial.Value)

: carRecordList.Where(x => !x.IsSpecial.Value);

}

// filter cars by values for CarList TypeID

int listTypeId;

if (Int32.TryParse(carListPart.Type, out listTypeId))

{

IList<string> values = \_carTypeService.GetCarTypeValues(listTypeId);

carRecordList = carRecordList.Where(x => x.Type != null && values.Contains(x.Type.ToLower().Trim()));

}

if (!string.IsNullOrEmpty(carListPart.DealerIds))

{

var dealerIds = carListPart.DealerIds.Split(new[] {CarListPart.DefaultSplitter},

StringSplitOptions.RemoveEmptyEntries);

carRecordList = carRecordList

.Where(x => x.DealerId != null && dealerIds.Contains(x.DealerId));

}

return carRecordList;

}

public ActionResult CarForModelList(int carModelListId, PagerParameters pagerParameters)

{

var part = \_services.ContentManager.Get(carModelListId, VersionOptions.Published).As<CarModelListPart>();

if(part==null)

{

return HttpNotFound();

}

var options = new CarListOptionsViewModel {Model = part.Model};

var pager = new Pager(\_siteService.GetSiteSettings(), pagerParameters);

var carSettings = \_services.WorkContext.CurrentSite.As<CarSettingsPart>();

IQueryable<CarRecord> carRecordList = \_carService.GetActiveRecords(options);

IList<SpecialRecord> specials = \_specialService.GetAllLive();

int totalItemCount = carRecordList.Count();

IList<CarRecord> carRecords = carRecordList.Skip(pager.GetStartIndex()).Take(pager.PageSize).ToList();

\_videoService.CarRecordsWithModifyVideoCode(carRecords);

dynamic pagerShape = Shape.Pager(pager).TotalItemCount(totalItemCount);

var path = part.CarProjectionPart != null ? part.CarProjectionPart.As<AutoroutePart>().Path : string.Empty;

var dealers = \_dealerSettingsService.GetDealerSettingsParts();

var carItems = carRecords.Select(rec =>

{

var specialForCar = \_specialService.GetSpecialsForCar(rec, specials);

var dealerSettings =

dealers.FirstOrDefault(

dealer => dealer.DealerId.Split(separator)

.Contains(rec.DealerId));

return new CarItemInListViewModel(specialForCar.ToList(),

rec,

path,

carSettings,

\_carTypeService.GetCarTypeValues(CarTypes.Used),

\_carTypeService.GetCarTypeValues(CarTypes.New),

\_carTypeService.GetCarTypeValues(CarTypes.Cpo),

\_carTypeService.GetCarTypeValues(CarTypes.Demo),

dealerSettings);

});

var model = new CarFilterResult

{

CarItemInListViewModels = carItems,

Path = path,

CarSettingsPart = carSettings,

Pager = pagerShape

};

return PartialView("FilterResult", model);

}

public ActionResult SearchData(int carListId, PagerParameters pagerParameters, CarListOptionsViewModel options)

{

var carListPart = \_services.ContentManager.Get(carListId, VersionOptions.Published).As<CarListPart>();

var carListName = carListPart.As<AutoroutePart>().Path;

if (!Request.IsAjaxRequest())

{

return Redirect(String.Format("~\\{0}?page={1}", carListName, pagerParameters.Page.HasValue ? pagerParameters.Page : 1));

}

var pager = new Pager(\_siteService.GetSiteSettings(), pagerParameters);

var carSettings = \_services.WorkContext.CurrentSite.As<CarSettingsPart>();

if (carListPart == null)

{

return HttpNotFound();

}

CarProjectionPart carProjection = \_carListService.GetForCarListPart(carListPart);

string path = string.Empty;

if (carProjection != null)

{

path =

\_services.ContentManager

.Get<CarProjectionPart>(carProjection.Id, VersionOptions.Published)

.As<AutoroutePart>()

.Path;

}

var carFilter = \_carListFilterProvider.FindFilter(carListName);

var carRecordList = \_carService.GetActiveRecords(options);

if (carFilter == null)

{

carRecordList = FilterCarsByListSettings(carRecordList, carListPart);

}

else

{

carRecordList = carFilter.Filter(carRecordList);

}

int totalItemCount = carRecordList.Count();

IList<CarRecord> carRecords;

if (options.SortBy == SortByFilter.ColorStyleAtoZ)

{

carRecords = carRecordList.ToList();

carRecords = SortWithEmptyLast(carRecords);

carRecords = carRecords.Skip(pager.GetStartIndex()).Take(pager.PageSize).ToList();

}

else

{

carRecords = carRecordList.Skip(pager.GetStartIndex()).Take(pager.PageSize).ToList();

}

\_videoService.CarRecordsWithModifyVideoCode(carRecords); //update video from video manager todo refacoring

dynamic pagerShape = Shape.Pager(pager).TotalItemCount(totalItemCount);

// maintain previous route data when generating page links

var routeData = new RouteData();

routeData.Values.Add("Stock", options.Stock);

routeData.Values.Add("Model", options.Model);

routeData.Values.Add("Trim", options.Trim);

routeData.Values.Add("Year", options.Year);

routeData.Values.Add("BodyStyle", options.BodyStyle);

routeData.Values.Add("MSRP", options.MSRP);

routeData.Values.Add("SortBy", options.SortBy);

pagerShape.RouteData(routeData);

IList<SpecialRecord> specials = \_specialService.GetAllLive();

var dealers = \_dealerSettingsService.GetDealerSettingsParts();

var carItems = carRecords.Select(rec =>

{

var specialForCar = \_specialService.GetSpecialsForCar(rec, specials);

var dealerSettings =

dealers.FirstOrDefault(

dealer => dealer.DealerId.Split(separator).

Contains(rec.DealerId));

return new CarItemInListViewModel(specialForCar.ToList(),

rec,

path,

carSettings,

\_carTypeService.GetCarTypeValues(CarTypes.Used),

\_carTypeService.GetCarTypeValues(CarTypes.New),

\_carTypeService.GetCarTypeValues(CarTypes.Cpo),

\_carTypeService.GetCarTypeValues(CarTypes.Demo),

dealerSettings);

});

var model = new CarFilterResult

{

CarItemInListViewModels = carItems,

Path = path,

CarSettingsPart = carSettings,

Pager = pagerShape

};

return PartialView("FilterResult", model);

}

#endregion

#region Helper

private IList<CarRecord> SortWithEmptyLast(IEnumerable<CarRecord> carRecords)

{

return carRecords.OrderBy(x => string.IsNullOrEmpty(x.ExteriorColor)).ToList();

}

#endregion

}

ПРИЛОЖЕНИЕ Г

(справочное)

Листинг кода класса CarsListService

public class CarsListService : ICarListService

{

private readonly IContentManager \_contentManager;

private readonly IRepository<CarListPartRecord> \_repository;

private readonly ICarTypeService \_carTypeService;

public CarsListService(IContentManager contentManager , IRepository<CarListPartRecord> repository, ICarTypeService carTypeService)

{

\_contentManager = contentManager;

\_repository = repository;

\_carTypeService = carTypeService;

}

public CarListPartRecord CarRecordById(int carListId)

{

return \_repository.Get(carListId);

}

public CarListPart Get(string path)

{

return

\_contentManager.Query<AutoroutePart, AutoroutePartRecord>().Where(r => r.DisplayAlias == path).ForPart<CarListPart>().Slice(0, 1).FirstOrDefault();

}

public CarListPart Get(int id)

{

return

\_contentManager.Query<AutoroutePart, AutoroutePartRecord>().Where(r => r.Id == id).ForPart<CarListPart>().Slice(0, 1).FirstOrDefault();

}

public ContentItem Get(int id, VersionOptions versionOptions)

{

var carListPart = \_contentManager.Get<CarContainerPart>(id, versionOptions);

return carListPart == null ? null : carListPart.ContentItem;

}

public IEnumerable<CarListPart> Get()

{

return Get(VersionOptions.Published);

}

public string GetCarListPathByType(CarTypes type) {

var carType = \_carTypeService.Get(type.ToString());

if(carType != null) {

var autoRoute = \_contentManager.Query<CarListPart, CarListPartRecord>(VersionOptions.Published)

.Where(cl => cl.Type == carType.Id.ToString())

.List<AutoroutePart>().FirstOrDefault();

return autoRoute != null ? autoRoute.Path : null;

}

return null;

}

public IEnumerable<CarListPart> Get(VersionOptions versionOptions)

{

var carLists = \_contentManager.Query<CarListPart, CarListPartRecord>(versionOptions)

.Join<TitlePartRecord>()

.OrderBy(br => br.Title)

.List();

return carLists;

}

public void Delete(ContentItem carList)

{

\_contentManager.Remove(carList);

}

public CarProjectionPart GetForCarListPart(CarListPart carListPart)

{

CarProjectionPart res;

res = carListPart == null

? null

: \_contentManager.Query<CarProjectionPart, CarProjectionPartRecord>().Where(

proj => proj.CarListPartId == carListPart.Id).Slice(0, 1).SingleOrDefault();

return res;

}

public CarProjectionPart GetForCarListPart(int carListRecordId)

{

CarProjectionPart res;

res = \_contentManager.Query<CarProjectionPart, CarProjectionPartRecord>().Where( proj => proj.CarListPartId == carListRecordId).Slice(0, 1).SingleOrDefault();

return res;

}

public ContentItem GetCarProjectionPart(int carProjectionId, VersionOptions versionOptions)

{

var carProjectionPart = \_contentManager.Get<CarProjectionPart>(carProjectionId, versionOptions);

return carProjectionPart == null ? null : carProjectionPart.ContentItem;

}

public ContentItem GetCarProjectionPart(string carProjectionGuid)

{

ContentItem result = null;

if(!string.IsNullOrEmpty(carProjectionGuid))

{

var carProjectionPart = \_contentManager

.Query<CarProjectionPart, CarProjectionPartRecord>()

.Where(cp => cp.CarProjectionGuid == carProjectionGuid)

.Slice(0, 1)

.SingleOrDefault();

result = carProjectionPart == null ? null : carProjectionPart.ContentItem;

}

return result;

}

public string GetPathByCarRecord(CarRecord carRecord)

{

CarListPart carProjectionlist = null;

CarProjectionPart carProjectionPart = null;

List<CarTypeRecord> carTypes = null;

if (carRecord != null)

{

carTypes = \_carTypeService.GetTypesByValue(carRecord.Type).ToList();

}

if (carTypes != null)

{

carProjectionlist = \_contentManager

.Query<CarListPart, CarListPartRecord>().List()

.Where(cl => carTypes.Select(ct => ct.Id.ToString()).Contains(cl.Type))

.FirstOrDefault(cl => !cl.IsCertified.HasValue || cl.IsCertified.Value == false);

}

if (carProjectionlist != null)

{

carProjectionPart = this.GetForCarListPart(carProjectionlist);

}

if (carProjectionPart == null)

{

var carLists =

\_contentManager.Query<CarListPart, CarListPartRecord>().List().FirstOrDefault(

cl => cl.CarProjectionPartField != null);

carProjectionPart = this.GetForCarListPart(carLists);

}

if (carProjectionPart != null)

{

return carProjectionPart.As<AutoroutePart>().Path;

}

return string.Empty;

}

public CarProjectionPart GetCarListProjection(string path)

{

return \_contentManager.Query<AutoroutePart, AutoroutePartRecord>().Where(r => r.DisplayAlias == path).ForPart<CarProjectionPart>().Slice(0, 1).FirstOrDefault();

}

public CarListPart GetCarListPart(int id)

{

return

\_contentManager.Query<AutoroutePart, AutoroutePartRecord>().Where(r => r.Id == id).ForPart<CarListPart>().Slice(0, 1).FirstOrDefault();

}

}

ПРИЛОЖЕНИЕ Д

(обязательное)

Спецификация

ПРИЛОЖЕНИЕ Е

(обязательное)

Ведомость документов