**钟万林项目代码**

**namespace MongoRepository
{
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
/// <summary>
/// Attribute used to annotate Enities with to override mongo collection name. By default, when this attribute
/// is not specified, the classname will be used.
/// </summary>
[AttributeUsage(AttributeTargets.Class, Inherited = true)]
public class CollectionName : Attribute
{
/// <summary>
/// Initializes a new instance of the CollectionName class attribute with the desired name.
/// </summary>
/// <param name="value">Name of the collection.</param>
public CollectionName(string value)
{
#if NET35
if (string.IsNullOrEmpty(value) || value.Trim().Length == 0)
#else
if (string.IsNullOrWhiteSpace(value))
#endif
throw new ArgumentException("Empty collectionname not allowed", "value");
this.Name = value;
}
/// <summary>
/// Gets the name of the collection.
/// </summary>
/// <value>The name of the collection.</value>
public virtual string Name { get; private set; }
}
}
namespace MongoRepository
{
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;
using System;
using System.Runtime.Serialization;
/// <summary>
/// Abstract Entity for all the BusinessEntities.
/// </summary>
[DataContract]
[Serializable]
[BsonIgnoreExtraElements(Inherited = true)]
public abstract class Entity : IEntity<string>
{
/// <summary>
/// Gets or sets the id for this object (the primary record for an entity).
/// </summary>
/// <value>The id for this object (the primary record for an entity).</value>
[DataMember]
[BsonRepresentation(BsonType.ObjectId)]
public virtual string Id { get; set; }
}
}
namespace MongoRepository
{
using MongoDB.Bson.Serialization.Attributes;
/// <summary>
/// Generic Entity interface.
/// </summary>
/// <typeparam name="TKey">The type used for the entity's Id.</typeparam>
public interface IEntity<TKey>
{
/// <summary>
/// Gets or sets the Id of the Entity.
/// </summary>
/// <value>Id of the Entity.</value>
[BsonId]
TKey Id { get; set; }
}
/// <summary>
/// "Default" Entity interface.
/// </summary>
/// <remarks>Entities are assumed to use strings for Id's.</remarks>
public interface IEntity : IEntity<string>
{
}
}
namespace MongoRepository
{
using MongoDB.Driver;
using System;
using System.Configuration;
/// <summary>
/// Internal miscellaneous utility functions.
/// </summary>
internal static class Util<U>
{
/// <summary>
/// The default key MongoRepository will look for in the App.config or Web.config file.
/// </summary>
private const string DefaultConnectionstringName = "MongoServerSettings";
/// <summary>
/// Retrieves the default connectionstring from the App.config or Web.config file.
/// </summary>
/// <returns>Returns the default connectionstring from the App.config or Web.config file.</returns>
public static string GetDefaultConnectionString()
{
return ConfigurationManager.ConnectionStrings[DefaultConnectionstringName].ConnectionString;
}
/// <summary>
/// Creates and returns a MongoDatabase from the specified url.
/// </summary>
/// <param name="url">The url to use to get the database from.</param>
/// <returns>Returns a MongoDatabase from the specified url.</returns>
private static MongoDatabase GetDatabaseFromUrl(MongoUrl url)
{
var client = new MongoClient(url);
var server = client.GetServer();
return server.GetDatabase(url.DatabaseName); // WriteConcern defaulted to Acknowledged
}
/// <summary>
/// Creates and returns a MongoCollection from the specified type and connectionstring.
/// </summary>
/// <typeparam name="T">The type to get the collection of.</typeparam>
/// <param name="connectionString">The connectionstring to use to get the collection from.</param>
/// <returns>Returns a MongoCollection from the specified type and connectionstring.</returns>
public static MongoCollection<T> GetCollectionFromConnectionString<T>(string connectionString)
where T : IEntity<U>
{
return Util<U>.GetCollectionFromConnectionString<T>(connectionString, GetCollectionName<T>());
}
/// <summary>
/// Creates and returns a MongoCollection from the specified type and connectionstring.
/// </summary>
/// <typeparam name="T">The type to get the collection of.</typeparam>
/// <param name="connectionString">The connectionstring to use to get the collection from.</param>
/// <param name="collectionName">The name of the collection to use.</param>
/// <returns>Returns a MongoCollection from the specified type and connectionstring.</returns>
public static MongoCollection<T> GetCollectionFromConnectionString<T>(string connectionString, string collectionName)
where T : IEntity<U>
{
return Util<U>.GetDatabaseFromUrl(new MongoUrl(connectionString))
.GetCollection<T>(collectionName);
}
/// <summary>
/// Creates and returns a MongoCollection from the specified type and url.
/// </summary>
/// <typeparam name="T">The type to get the collection of.</typeparam>
/// <param name="url">The url to use to get the collection from.</param>
/// <returns>Returns a MongoCollection from the specified type and url.</returns>
public static MongoCollection<T> GetCollectionFromUrl<T>(MongoUrl url)
where T : IEntity<U>
{
return Util<U>.GetCollectionFromUrl<T>(url, GetCollectionName<T>());
}
/// <summary>
/// Creates and returns a MongoCollection from the specified type and url.
/// </summary>
/// <typeparam name="T">The type to get the collection of.</typeparam>
/// <param name="url">The url to use to get the collection from.</param>
/// <param name="collectionName">The name of the collection to use.</param>
/// <returns>Returns a MongoCollection from the specified type and url.</returns>
public static MongoCollection<T> GetCollectionFromUrl<T>(MongoUrl url, string collectionName)
where T : IEntity<U>
{
return Util<U>.GetDatabaseFromUrl(url)
.GetCollection<T>(collectionName);
}
/// <summary>
/// Determines the collectionname for T and assures it is not empty
/// </summary>
/// <typeparam name="T">The type to determine the collectionname for.</typeparam>
/// <returns>Returns the collectionname for T.</returns>
private static string GetCollectionName<T>() where T : IEntity<U>
{
string collectionName;
if (typeof(T).BaseType.Equals(typeof(object)))
{
collectionName = GetCollectioNameFromInterface<T>();
}
else
{
collectionName = GetCollectionNameFromType(typeof(T));
}
if (string.IsNullOrEmpty(collectionName))
{
throw new ArgumentException("Collection name cannot be empty for this entity");
}
return collectionName;
}
/// <summary>
/// Determines the collectionname from the specified type.
/// </summary>
/// <typeparam name="T">The type to get the collectionname from.</typeparam>
/// <returns>Returns the collectionname from the specified type.</returns>
private static string GetCollectioNameFromInterface<T>()
{
string collectionname;
// Check to see if the object (inherited from Entity) has a CollectionName attribute
var att = Attribute.GetCustomAttribute(typeof(T), typeof(CollectionName));
if (att != null)
{
// It does! Return the value specified by the CollectionName attribute
collectionname = ((CollectionName)att).Name;
}
else
{
collectionname = typeof(T).Name;
}
return collectionname;
}
/// <summary>
/// Determines the collectionname from the specified type.
/// </summary>
/// <param name="entitytype">The type of the entity to get the collectionname from.</param>
/// <returns>Returns the collectionname from the specified type.</returns>
private static string GetCollectionNameFromType(Type entitytype)
{
string collectionname;
// Check to see if the object (inherited from Entity) has a CollectionName attribute
var att = Attribute.GetCustomAttribute(entitytype, typeof(CollectionName));
if (att != null)
{
// It does! Return the value specified by the CollectionName attribute
collectionname = ((CollectionName)att).Name;
}
else
{
if (typeof(Entity).IsAssignableFrom(entitytype))
{
// No attribute found, get the basetype
while (!entitytype.BaseType.Equals(typeof(Entity)))
{
entitytype = entitytype.BaseType;
}
}
collectionname = entitytype.Name;
}
return collectionname;
}
}
}
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("MongoRepository")]
[assembly: AssemblyDescription("Repository Implementation for MongoDB C# Driver")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Devcorner.nl")]
[assembly: AssemblyProduct("MongoRepository")]
[assembly: AssemblyCopyright("Copyright © 2011 - 2015 Rob Janssen (RobIII)")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]
// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("7e3d1784-b1f3-4b33-a303-7facc780917e")]
[assembly: AssemblyVersion("1.6.10.0")]
[assembly: AssemblyFileVersion("1.6.10.0")]
namespace MongoRepository
{
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using MongoDB.Bson;
using MongoDB.Driver;
/// <summary>
/// IRepository definition.
/// </summary>
/// <typeparam name="T">The type contained in the repository.</typeparam>
/// <typeparam name="TKey">The type used for the entity's Id.</typeparam>
public interface IRepository<T, TKey> : IQueryable<T>
where T : IEntity<TKey>
{
/// <summary>
/// Gets the Mongo collection (to perform advanced operations).
/// </summary>
/// <remarks>
/// One can argue that exposing this property (and with that, access to it's Database property for instance
/// (which is a "parent")) is not the responsibility of this class. Use of this property is highly discouraged;
/// for most purposes you can use the MongoRepositoryManager&lt;T&gt;
/// </remarks>
/// <value>The Mongo collection (to perform advanced operations).</value>
MongoCollection<T> Collection { get; }
/// <summary>
/// Returns the T by its given id.
/// </summary>
/// <param name="id">The value representing the ObjectId of the entity to retrieve.</param>
/// <returns>The Entity T.</returns>
T GetById(TKey id);
/// <summary>
/// Adds the new entity in the repository.
/// </summary>
/// <param name="entity">The entity to add.</param>
/// <returns>The added entity including its new ObjectId.</returns>
T Add(T entity);
/// <summary>
/// Adds the new entities in the repository.
/// </summary>
/// <param name="entities">The entities of type T.</param>
void Add(IEnumerable<T> entities);
/// <summary>
/// Adds the new object in the repository.
/// </summary>
/// <param name="object"></param>
void Add(object ojb);
/// <summary>
/// Adds the json in the repository.
/// </summary>
/// <param name="json"></param>
void AddFromJson(string json);
/// <summary>
/// Upserts an entity.
/// </summary>
/// <param name="entity">The entity.</param>
/// <returns>The updated entity.</returns>
T Update(T entity);
/// <summary>
/// Upserts the entities.
/// </summary>
/// <param name="entities">The entities to update.</param>
void Update(IEnumerable<T> entities);
/// <summary>
/// Deletes an entity from the repository by its id.
/// </summary>
/// <param name="id">The entity's id.</param>
void Delete(TKey id);
/// <summary>
/// Deletes the given entity.
/// </summary>
/// <param name="entity">The entity to delete.</param>
void Delete(T entity);
/// <summary>
/// Deletes the entities matching the predicate.
/// </summary>
/// <param name="predicate">The expression.</param>
void Delete(Expression<Func<T, bool>> predicate);
/// <summary>
/// Deletes all entities in the repository.
/// </summary>
void DeleteAll();
/// <summary>
/// Counts the total entities in the repository.
/// </summary>
/// <returns>Count of entities in the repository.</returns>
long Count();
/// <summary>
/// Checks if the entity exists for given predicate.
/// </summary>
/// <param name="predicate">The expression.</param>
/// <returns>True when an entity matching the predicate exists, false otherwise.</returns>
bool Exists(Expression<Func<T, bool>> predicate);
/// <summary>
/// Lets the server know that this thread is about to begin a series of related operations that must all occur
/// on the same connection. The return value of this method implements IDisposable and can be placed in a using
/// statement (in which case RequestDone will be called automatically when leaving the using statement).
/// </summary>
/// <returns>A helper object that implements IDisposable and calls RequestDone() from the Dispose method.</returns>
/// <remarks>
/// Sometimes a series of operations needs to be performed on the same connection in order to guarantee correct
/// results. This is rarely the case, and most of the time there is no need to call RequestStart/RequestDone.
/// An example of when this might be necessary is when a series of Inserts are called in rapid succession with
/// SafeMode off, and you want to query that data in a consistent manner immediately thereafter (with SafeMode
/// off the writes can queue up at the server and might not be immediately visible to other connections). Using
/// RequestStart you can force a query to be on the same connection as the writes, so the query won't execute
/// until the server has caught up with the writes.
/// A thread can temporarily reserve a connection from the connection pool by using RequestStart and
/// RequestDone. You are free to use any other databases as well during the request. RequestStart increments a
/// counter (for this thread) and RequestDone decrements the counter. The connection that was reserved is not
/// actually returned to the connection pool until the count reaches zero again. This means that calls to
/// RequestStart/RequestDone can be nested and the right thing will happen.
/// </remarks>
//IDisposable RequestStart();
/// <summary>
/// Lets the server know that this thread is done with a series of related operations.
/// </summary>
/// <remarks>
/// Instead of calling this method it is better to put the return value of RequestStart in a using statement.
/// </remarks>
//void RequestDone();
}
/// <summary>
/// IRepository definition.
/// </summary>
/// <typeparam name="T">The type contained in the repository.</typeparam>
/// <remarks>Entities are assumed to use strings for Id's.</remarks>
public interface IRepository<T> : IQueryable<T>, IRepository<T, string>
where T : IEntity<string> { }
}
namespace MongoRepository
{
using MongoDB.Bson;
using MongoDB.Driver;
using MongoDB.Driver.Builders;
using MongoDB.Driver.Linq;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
/// <summary>
/// Deals with entities in MongoDb.
/// </summary>
/// <typeparam name="T">The type contained in the repository.</typeparam>
/// <typeparam name="TKey">The type used for the entity's Id.</typeparam>
public class MongoRepository<T, TKey> : IRepository<T, TKey>
where T : IEntity<TKey>
{
/// <summary>
/// MongoCollection field.
/// </summary>
protected internal MongoCollection<T> collection;
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// Uses the Default App/Web.Config connectionstrings to fetch the connectionString and Database name.
/// </summary>
/// <remarks>Default constructor defaults to "MongoServerSettings" key for connectionstring.</remarks>
public MongoRepository()
: this(Util<TKey>.GetDefaultConnectionString())
{
}
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
public MongoRepository(string connectionString)
{
this.collection = Util<TKey>.GetCollectionFromConnectionString<T>(connectionString);
}
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
/// <param name="collectionName">The name of the collection to use.</param>
public MongoRepository(string connectionString, string collectionName)
{
this.collection = Util<TKey>.GetCollectionFromConnectionString<T>(connectionString, collectionName);
}
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="url">Url to use for connecting to MongoDB.</param>
public MongoRepository(MongoUrl url)
{
this.collection = Util<TKey>.GetCollectionFromUrl<T>(url);
}
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="url">Url to use for connecting to MongoDB.</param>
/// <param name="collectionName">The name of the collection to use.</param>
public MongoRepository(MongoUrl url, string collectionName)
{
this.collection = Util<TKey>.GetCollectionFromUrl<T>(url, collectionName);
}
/// <summary>
/// Gets the Mongo collection (to perform advanced operations).
/// </summary>
/// <remarks>
/// One can argue that exposing this property (and with that, access to it's Database property for instance
/// (which is a "parent")) is not the responsibility of this class. Use of this property is highly discouraged;
/// for most purposes you can use the MongoRepositoryManager&lt;T&gt;
/// </remarks>
/// <value>The Mongo collection (to perform advanced operations).</value>
public MongoCollection<T> Collection
{
get { return this.collection; }
}
/// <summary>
/// Gets the name of the collection
/// </summary>
public string CollectionName
{
get { return this.collection.Name; }
}
/// <summary>
/// Returns the T by its given id.
/// </summary>
/// <param name="id">The Id of the entity to retrieve.</param>
/// <returns>The Entity T.</returns>
public virtual T GetById(TKey id)
{
if (typeof(T).IsSubclassOf(typeof(Entity)))
{
return this.GetById(new ObjectId(id as string));
}
return this.collection.FindOneByIdAs<T>(BsonValue.Create(id));
}
/// <summary>
/// Returns the T by its given id.
/// </summary>
/// <param name="id">The Id of the entity to retrieve.</param>
/// <returns>The Entity T.</returns>
public virtual T GetById(ObjectId id)
{
return this.collection.FindOneByIdAs<T>(id);
}
/// <summary>
/// Adds the new entity in the repository.
/// </summary>
/// <param name="entity">The entity T.</param>
/// <returns>The added entity including its new ObjectId.</returns>
public virtual T Add(T entity)
{
this.collection.Insert<T>(entity);
return entity;
}
/// <summary>
/// Adds the new entities in the repository.
/// </summary>
/// <param name="entities">The entities of type T.</param>
public virtual void Add(IEnumerable<T> entities)
{
this.collection.InsertBatch<T>(entities);
}
/// <summary>
/// Adds the new object in the repository.
/// </summary>
/// <param name="object"></param>
public virtual void Add(object obj)
{
if (obj == null) throw new ArgumentNullException("obj is null");
BsonDocument bs = BsonDocument.Create(obj);
collection.Insert(bs);
}
/// <summary>
/// Adds the json in the repository.
/// </summary>
/// <param name="json"></param>
public virtual void AddFromJson(string json)
{
BsonDocument bs;
if (!BsonDocument.TryParse(json, out bs)) return;
collection.Insert(bs);
}
/// <summary>
/// Upserts an entity.
/// </summary>
/// <param name="entity">The entity.</param>
/// <returns>The updated entity.</returns>
public virtual T Update(T entity)
{
this.collection.Save<T>(entity);
return entity;
}
/// <summary>
/// Upserts the entities.
/// </summary>
/// <param name="entities">The entities to update.</param>
public virtual void Update(IEnumerable<T> entities)
{
foreach (T entity in entities)
{
this.collection.Save<T>(entity);
}
}
/// <summary>
/// Deletes an entity from the repository by its id.
/// </summary>
/// <param name="id">The entity's id.</param>
public virtual void Delete(TKey id)
{
if (typeof(T).IsSubclassOf(typeof(Entity)))
{
this.collection.Remove(Query.EQ("\_id", new ObjectId(id as string)));
}
else
{
this.collection.Remove(Query.EQ("\_id", BsonValue.Create(id)));
}
}
/// <summary>
/// Deletes an entity from the repository by its ObjectId.
/// </summary>
/// <param name="id">The ObjectId of the entity.</param>
public virtual void Delete(ObjectId id)
{
this.collection.Remove(Query.EQ("\_id", id));
}
/// <summary>
/// Deletes the given entity.
/// </summary>
/// <param name="entity">The entity to delete.</param>
public virtual void Delete(T entity)
{
this.Delete(entity.Id);
}
/// <summary>
/// Deletes the entities matching the predicate.
/// </summary>
/// <param name="predicate">The expression.</param>
public virtual void Delete(Expression<Func<T, bool>> predicate)
{
foreach (T entity in this.collection.AsQueryable<T>().Where(predicate))
{
this.Delete(entity.Id);
}
}
/// <summary>
/// Deletes all entities in the repository.
/// </summary>
public virtual void DeleteAll()
{
this.collection.RemoveAll();
}
/// <summary>
/// Counts the total entities in the repository.
/// </summary>
/// <returns>Count of entities in the collection.</returns>
public virtual long Count()
{
return this.collection.Count();
}
/// <summary>
/// Checks if the entity exists for given predicate.
/// </summary>
/// <param name="predicate">The expression.</param>
/// <returns>True when an entity matching the predicate exists, false otherwise.</returns>
public virtual bool Exists(Expression<Func<T, bool>> predicate)
{
return this.collection.AsQueryable<T>().Any(predicate);
}
/// <summary>
/// Lets the server know that this thread is about to begin a series of related operations that must all occur
/// on the same connection. The return value of this method implements IDisposable and can be placed in a using
/// statement (in which case RequestDone will be called automatically when leaving the using statement).
/// </summary>
/// <returns>A helper object that implements IDisposable and calls RequestDone() from the Dispose method.</returns>
/// <remarks>
/// <para>
/// Sometimes a series of operations needs to be performed on the same connection in order to guarantee correct
/// results. This is rarely the case, and most of the time there is no need to call RequestStart/RequestDone.
/// An example of when this might be necessary is when a series of Inserts are called in rapid succession with
/// SafeMode off, and you want to query that data in a consistent manner immediately thereafter (with SafeMode
/// off the writes can queue up at the server and might not be immediately visible to other connections). Using
/// RequestStart you can force a query to be on the same connection as the writes, so the query won't execute
/// until the server has caught up with the writes.
/// </para>
/// <para>
/// A thread can temporarily reserve a connection from the connection pool by using RequestStart and
/// RequestDone. You are free to use any other databases as well during the request. RequestStart increments a
/// counter (for this thread) and RequestDone decrements the counter. The connection that was reserved is not
/// actually returned to the connection pool until the count reaches zero again. This means that calls to
/// RequestStart/RequestDone can be nested and the right thing will happen.
/// </para>
/// <para>
/// Use the connectionstring to specify the readpreference; add "readPreference=X" where X is one of the following
/// values: primary, primaryPreferred, secondary, secondaryPreferred, nearest.
/// See http://docs.mongodb.org/manual/applications/replication/#read-preference
/// </para>
/// </remarks>
public virtual IDisposable RequestStart()
{
return null;//this.collection.Database.RequestStart();
}
/// <summary>
/// Lets the server know that this thread is done with a series of related operations.
/// </summary>
/// <remarks>
/// Instead of calling this method it is better to put the return value of RequestStart in a using statement.
/// </remarks>
public virtual void RequestDone()
{
//this.collection.Database.RequestDone();
}
#region IQueryable<T>
/// <summary>
/// Returns an enumerator that iterates through a collection.
/// </summary>
/// <returns>An IEnumerator&lt;T&gt; object that can be used to iterate through the collection.</returns>
public virtual IEnumerator<T> GetEnumerator()
{
return this.collection.AsQueryable<T>().GetEnumerator();
}
/// <summary>
/// Returns an enumerator that iterates through a collection.
/// </summary>
/// <returns>An IEnumerator object that can be used to iterate through the collection.</returns>
System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()
{
return this.collection.AsQueryable<T>().GetEnumerator();
}
/// <summary>
/// Gets the type of the element(s) that are returned when the expression tree associated with this instance of IQueryable is executed.
/// </summary>
public virtual Type ElementType
{
get { return this.collection.AsQueryable<T>().ElementType; }
}
/// <summary>
/// Gets the expression tree that is associated with the instance of IQueryable.
/// </summary>
public virtual Expression Expression
{
get { return this.collection.AsQueryable<T>().Expression; }
}
/// <summary>
/// Gets the query provider that is associated with this data source.
/// </summary>
public virtual IQueryProvider Provider
{
get { return this.collection.AsQueryable<T>().Provider; }
}
#endregion
}
/// <summary>
/// Deals with entities in MongoDb.
/// </summary>
/// <typeparam name="T">The type contained in the repository.</typeparam>
/// <remarks>Entities are assumed to use strings for Id's.</remarks>
public class MongoRepository<T> : MongoRepository<T, string>, IRepository<T>
where T : IEntity<string>
{
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// Uses the Default App/Web.Config connectionstrings to fetch the connectionString and Database name.
/// </summary>
/// <remarks>Default constructor defaults to "MongoServerSettings" key for connectionstring.</remarks>
public MongoRepository()
: base() { }
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="url">Url to use for connecting to MongoDB.</param>
public MongoRepository(MongoUrl url)
: base(url) { }
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="url">Url to use for connecting to MongoDB.</param>
/// <param name="collectionName">The name of the collection to use.</param>
public MongoRepository(MongoUrl url, string collectionName)
: base(url, collectionName) { }
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
public MongoRepository(string connectionString)
: base(connectionString) { }
/// <summary>
/// Initializes a new instance of the MongoRepository class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
/// <param name="collectionName">The name of the collection to use.</param>
public MongoRepository(string connectionString, string collectionName)
: base(connectionString, collectionName) { }
}
}
namespace MongoRepository
{
using System.Collections.Generic;
using MongoDB.Driver;
using System;
/// <summary>
/// IRepositoryManager definition.
/// </summary>
/// <typeparam name="T">The type contained in the repository to manage.</typeparam>
/// <typeparam name="TKey">The type used for the entity's Id.</typeparam>
public interface IRepositoryManager<T, TKey>
where T : IEntity<TKey>
{
/// <summary>
/// Gets a value indicating whether the collection already exists.
/// </summary>
/// <value>Returns true when the collection already exists, false otherwise.</value>
bool Exists { get; }
/// <summary>
/// Gets the name of the collection as Mongo uses.
/// </summary>
/// <value>The name of the collection as Mongo uses.</value>
string Name { get; }
/// <summary>
/// Drops the repository.
/// </summary>
void Drop();
/// <summary>
/// Tests whether the repository is capped.
/// </summary>
/// <returns>Returns true when the repository is capped, false otherwise.</returns>
bool IsCapped();
/// <summary>
/// Drops specified index on the repository.
/// </summary>
/// <param name="keyname">The name of the indexed field.</param>
void DropIndex(string keyname);
/// <summary>
/// Drops specified indexes on the repository.
/// </summary>
/// <param name="keynames">The names of the indexed fields.</param>
void DropIndexes(IEnumerable<string> keynames);
/// <summary>
/// Drops all indexes on this repository.
/// </summary>
void DropAllIndexes();
/// <summary>
/// Ensures that the desired index exist and creates it if it doesn't exist.
/// </summary>
/// <param name="keyname">The indexed field.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// Index will be ascending order, non-unique, non-sparse.
/// </remarks>
void EnsureIndex(string keyname);
/// <summary>
/// Ensures that the desired index exist and creates it if it doesn't exist.
/// </summary>
/// <param name="keyname">The indexed field.</param>
/// <param name="descending">Set to true to make index descending, false for ascending.</param>
/// <param name="unique">Set to true to ensure index enforces unique values.</param>
/// <param name="sparse">Set to true to specify the index is sparse.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// </remarks>
void EnsureIndex(string keyname, bool descending, bool unique, bool sparse);
/// <summary>
/// Ensures that the desired indexes exist and creates them if they don't exist.
/// </summary>
/// <param name="keynames">The indexed fields.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// Index will be ascending order, non-unique, non-sparse.
/// </remarks>
void EnsureIndexes(IEnumerable<string> keynames);
/// <summary>
/// Ensures that the desired indexes exist and creates them if they don't exist.
/// </summary>
/// <param name="keynames">The indexed fields.</param>
/// <param name="descending">Set to true to make index descending, false for ascending.</param>
/// <param name="unique">Set to true to ensure index enforces unique values.</param>
/// <param name="sparse">Set to true to specify the index is sparse.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// </remarks>
void EnsureIndexes(IEnumerable<string> keynames, bool descending, bool unique, bool sparse);
/// <summary>
/// Ensures that the desired indexes exist and creates them if they don't exist.
/// </summary>
/// <param name="keys">The indexed fields.</param>
/// <param name="options">The index options.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// Index will be ascending order, non-unique, non-sparse.
/// </remarks>
void EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options);
/// <summary>
/// Tests whether indexes exist.
/// </summary>
/// <param name="keyname">The indexed fields.</param>
/// <returns>Returns true when the indexes exist, false otherwise.</returns>
bool IndexExists(string keyname);
/// <summary>
/// Tests whether indexes exist.
/// </summary>
/// <param name="keynames">The indexed fields.</param>
/// <returns>Returns true when the indexes exist, false otherwise.</returns>
bool IndexesExists(IEnumerable<string> keynames);
/// <summary>
/// Runs the ReIndex command on this repository.
/// </summary>
void ReIndex();
/// <summary>
/// Gets the total size for the repository (data + indexes).
/// </summary>
/// <returns>Returns total size for the repository (data + indexes).</returns>
long GetTotalDataSize();
/// <summary>
/// Gets the total storage size for the repository (data + indexes).
/// </summary>
/// <returns>Returns total storage size for the repository (data + indexes).</returns>
long GetTotalStorageSize();
/// <summary>
/// Validates the integrity of the repository.
/// </summary>
/// <returns>Returns a ValidateCollectionResult.</returns>
/// <remarks>You will need to reference MongoDb.Driver.</remarks>
ValidateCollectionResult Validate();
/// <summary>
/// Gets stats for this repository.
/// </summary>
/// <returns>Returns a CollectionStatsResult.</returns>
/// <remarks>You will need to reference MongoDb.Driver.</remarks>
CollectionStatsResult GetStats();
/// <summary>
/// Gets the indexes for this repository.
/// </summary>
/// <returns>Returns the indexes for this repository.</returns>
GetIndexesResult GetIndexes();
}
/// <summary>
/// IRepositoryManager definition.
/// </summary>
/// <typeparam name="T">The type contained in the repository to manage.</typeparam>
/// <remarks>Entities are assumed to use strings for Id's.</remarks>
public interface IRepositoryManager<T> : IRepositoryManager<T, string>
where T : IEntity<string> { }
}
namespace MongoRepository
{
using MongoDB.Driver;
using MongoDB.Driver.Builders;
using System;
using System.Collections.Generic;
using System.Linq;
// TODO: Code coverage here is near-zero. A new RepoManagerTests.cs class needs to be created and we need to
// test these methods. Ofcourse we also need to update codeplex documentation on this entirely new object.
// This is a work-in-progress.
// TODO: GetStats(), Validate(), GetIndexes and EnsureIndexes(IMongoIndexKeys, IMongoIndexOptions) "leak"
// MongoDb-specific details. These probably need to get wrapped in MongoRepository specific objects to hide
// MongoDb.
/// <summary>
/// Deals with the collections of entities in MongoDb. This class tries to hide as much MongoDb-specific details
/// as possible but it's not 100% \*yet\*. It is a very thin wrapper around most methods on MongoDb's MongoCollection
/// objects.
/// </summary>
/// <typeparam name="T">The type contained in the repository to manage.</typeparam>
/// <typeparam name="TKey">The type used for the entity's Id.</typeparam>
public class MongoRepositoryManager<T, TKey> : IRepositoryManager<T, TKey>
where T : IEntity<TKey>
{
/// <summary>
/// MongoCollection field.
/// </summary>
private MongoCollection<T> collection;
/// <summary>
/// Initializes a new instance of the MongoRepositoryManager class.
/// Uses the Default App/Web.Config connectionstrings to fetch the connectionString and Database name.
/// </summary>
/// <remarks>Default constructor defaults to "MongoServerSettings" key for connectionstring.</remarks>
public MongoRepositoryManager()
: this(Util<TKey>.GetDefaultConnectionString())
{
}
/// <summary>
/// Initializes a new instance of the MongoRepositoryManager class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
public MongoRepositoryManager(string connectionString)
{
this.collection = Util<TKey>.GetCollectionFromConnectionString<T>(connectionString);
}
/// <summary>
/// Initializes a new instance of the MongoRepositoryManager class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
/// <param name="collectionName">The name of the collection to use.</param>
public MongoRepositoryManager(string connectionString, string collectionName)
{
this.collection = Util<TKey>.GetCollectionFromConnectionString<T>(connectionString, collectionName);
}
/// <summary>
/// Gets a value indicating whether the collection already exists.
/// </summary>
/// <value>Returns true when the collection already exists, false otherwise.</value>
public virtual bool Exists
{
get { return this.collection.Exists(); }
}
/// <summary>
/// Gets the name of the collection as Mongo uses.
/// </summary>
/// <value>The name of the collection as Mongo uses.</value>
public virtual string Name
{
get { return this.collection.Name; }
}
/// <summary>
/// Drops the collection.
/// </summary>
public virtual void Drop()
{
this.collection.Drop();
}
/// <summary>
/// Tests whether the repository is capped.
/// </summary>
/// <returns>Returns true when the repository is capped, false otherwise.</returns>
public virtual bool IsCapped()
{
return this.collection.IsCapped();
}
/// <summary>
/// Drops specified index on the repository.
/// </summary>
/// <param name="keyname">The name of the indexed field.</param>
public virtual void DropIndex(string keyname)
{
this.DropIndexes(new string[] { keyname });
}
/// <summary>
/// Drops specified indexes on the repository.
/// </summary>
/// <param name="keynames">The names of the indexed fields.</param>
public virtual void DropIndexes(IEnumerable<string> keynames)
{
this.collection.DropIndex(keynames.ToArray());
}
/// <summary>
/// Drops all indexes on this repository.
/// </summary>
public virtual void DropAllIndexes()
{
this.collection.DropAllIndexes();
}
/// <summary>
/// Ensures that the desired index exist and creates it if it doesn't exist.
/// </summary>
/// <param name="keyname">The indexed field.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// Index will be ascending order, non-unique, non-sparse.
/// </remarks>
public virtual void EnsureIndex(string keyname)
{
this.EnsureIndexes(new string[] { keyname });
}
/// <summary>
/// Ensures that the desired index exist and creates it if it doesn't exist.
/// </summary>
/// <param name="keyname">The indexed field.</param>
/// <param name="descending">Set to true to make index descending, false for ascending.</param>
/// <param name="unique">Set to true to ensure index enforces unique values.</param>
/// <param name="sparse">Set to true to specify the index is sparse.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// </remarks>
public virtual void EnsureIndex(string keyname, bool descending, bool unique, bool sparse)
{
this.EnsureIndexes(new string[] { keyname }, descending, unique, sparse);
}
/// <summary>
/// Ensures that the desired indexes exist and creates them if they don't exist.
/// </summary>
/// <param name="keynames">The indexed fields.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// Index will be ascending order, non-unique, non-sparse.
/// </remarks>
public virtual void EnsureIndexes(IEnumerable<string> keynames)
{
this.EnsureIndexes(keynames, false, false, false);
}
/// <summary>
/// Ensures that the desired indexes exist and creates them if they don't exist.
/// </summary>
/// <param name="keynames">The indexed fields.</param>
/// <param name="descending">Set to true to make index descending, false for ascending.</param>
/// <param name="unique">Set to true to ensure index enforces unique values.</param>
/// <param name="sparse">Set to true to specify the index is sparse.</param>
/// <remarks>
/// This is a convenience method for EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options).
/// </remarks>
public virtual void EnsureIndexes(IEnumerable<string> keynames, bool descending, bool unique, bool sparse)
{
var ixk = new IndexKeysBuilder();
if (descending)
{
ixk.Descending(keynames.ToArray());
}
else
{
ixk.Ascending(keynames.ToArray());
}
this.EnsureIndexes(
ixk,
new IndexOptionsBuilder().SetUnique(unique).SetSparse(sparse));
}
/// <summary>
/// Ensures that the desired indexes exist and creates them if they don't exist.
/// </summary>
/// <param name="keys">The indexed fields.</param>
/// <param name="options">The index options.</param>
/// <remarks>
/// This method allows ultimate control but does "leak" some MongoDb specific implementation details.
/// </remarks>
public virtual void EnsureIndexes(IMongoIndexKeys keys, IMongoIndexOptions options)
{
this.collection.CreateIndex(keys, options);
}
/// <summary>
/// Tests whether indexes exist.
/// </summary>
/// <param name="keyname">The indexed fields.</param>
/// <returns>Returns true when the indexes exist, false otherwise.</returns>
public virtual bool IndexExists(string keyname)
{
return this.IndexesExists(new string[] { keyname });
}
/// <summary>
/// Tests whether indexes exist.
/// </summary>
/// <param name="keynames">The indexed fields.</param>
/// <returns>Returns true when the indexes exist, false otherwise.</returns>
public virtual bool IndexesExists(IEnumerable<string> keynames)
{
return this.collection.IndexExists(keynames.ToArray());
}
/// <summary>
/// Runs the ReIndex command on this repository.
/// </summary>
public virtual void ReIndex()
{
this.collection.ReIndex();
}
/// <summary>
/// Gets the total size for the repository (data + indexes).
/// </summary>
/// <returns>Returns total size for the repository (data + indexes).</returns>
[Obsolete("This method will be removed in the next version of the driver")]
public virtual long GetTotalDataSize()
{
return this.collection.GetStats().DataSize;
}
/// <summary>
/// Gets the total storage size for the repository (data + indexes).
/// </summary>
/// <returns>Returns total storage size for the repository (data + indexes).</returns>
[Obsolete("This method will be removed in the next version of the driver")]
public virtual long GetTotalStorageSize()
{
return this.collection.GetStats().StorageSize;
}
/// <summary>
/// Validates the integrity of the repository.
/// </summary>
/// <returns>Returns a ValidateCollectionResult.</returns>
/// <remarks>You will need to reference MongoDb.Driver.</remarks>
public virtual ValidateCollectionResult Validate()
{
return this.collection.Validate();
}
/// <summary>
/// Gets stats for this repository.
/// </summary>
/// <returns>Returns a CollectionStatsResult.</returns>
/// <remarks>You will need to reference MongoDb.Driver.</remarks>
public virtual CollectionStatsResult GetStats()
{
return this.collection.GetStats();
}
/// <summary>
/// Gets the indexes for this repository.
/// </summary>
/// <returns>Returns the indexes for this repository.</returns>
public virtual GetIndexesResult GetIndexes()
{
return this.collection.GetIndexes();
}
}
/// <summary>
/// Deals with the collections of entities in MongoDb. This class tries to hide as much MongoDb-specific details
/// as possible but it's not 100% \*yet\*. It is a very thin wrapper around most methods on MongoDb's MongoCollection
/// objects.
/// </summary>
/// <typeparam name="T">The type contained in the repository to manage.</typeparam>
/// <remarks>Entities are assumed to use strings for Id's.</remarks>
public class MongoRepositoryManager<T> : MongoRepositoryManager<T, string>, IRepositoryManager<T>
where T : IEntity<string>
{
/// <summary>
/// Initializes a new instance of the MongoRepositoryManager class.
/// Uses the Default App/Web.Config connectionstrings to fetch the connectionString and Database name.
/// </summary>
/// <remarks>Default constructor defaults to "MongoServerSettings" key for connectionstring.</remarks>
public MongoRepositoryManager()
: base() { }
/// <summary>
/// Initializes a new instance of the MongoRepositoryManager class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
public MongoRepositoryManager(string connectionString)
: base(connectionString) { }
/// <summary>
/// Initializes a new instance of the MongoRepositoryManager class.
/// </summary>
/// <param name="connectionString">Connectionstring to use for connecting to MongoDB.</param>
/// <param name="collectionName">The name of the collection to use.</param>
public MongoRepositoryManager(string connectionString, string collectionName)
: base(connectionString, collectionName) { }
}
}
using MongoRepository;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
namespace MongoRepositoryTests
{
/// <summary>
///This is a test class for MongoRepositoryManagerTest and is intended
///to contain all MongoRepositoryManagerTest Unit Tests
///</summary>
[TestClass()]
public class MongoRepositoryManagerTest
{
private TestContext testContextInstance;
/// <summary>
///Gets or sets the test context which provides
///information about and functionality for the current test run.
///</summary>
public TestContext TestContext
{
get
{
return testContextInstance;
}
set
{
testContextInstance = value;
}
}
#region Additional test attributes
//
//You can use the following additional attributes as you write your tests:
//
//Use ClassInitialize to run code before running the first test in the class
//[ClassInitialize()]
//public static void MyClassInitialize(TestContext testContext)
//{
//}
//
//Use ClassCleanup to run code after all tests in a class have run
//[ClassCleanup()]
//public static void MyClassCleanup()
//{
//}
//
//Use TestInitialize to run code before running each test
//[TestInitialize()]
//public void MyTestInitialize()
//{
//}
//
//Use TestCleanup to run code after each test has run
//[TestCleanup()]
//public void MyTestCleanup()
//{
//}
//
#endregion
}
}
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MongoDB.Driver;
using MongoRepository;
using MongoRepositoryTests.Entities;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
namespace MongoRepositoryTests
{
//TODO: We REALLY need some decent tests and cleanup this mess.
[TestClass]
public class RepoTests
{
[TestInitialize]
public void Setup()
{
this.DropDB();
}
[TestCleanup]
public void Cleanup()
{
this.DropDB();
}
private void DropDB()
{
var url = new MongoUrl(ConfigurationManager.ConnectionStrings["MongoServerSettings"].ConnectionString);
var client = new MongoClient(url);
client.GetServer().DropDatabase(url.DatabaseName);
}
[TestMethod]
public void AddAndUpdateTest()
{
IRepository<Customer> \_customerRepo = new MongoRepository<Customer>();
IRepositoryManager<Customer> \_customerMan = new MongoRepositoryManager<Customer>();
Assert.IsFalse(\_customerMan.Exists);
var customer = new Customer();
customer.FirstName = "Bob";
customer.LastName = "Dillon";
customer.Phone = "0900999899";
customer.Email = "Bob.dil@snailmail.com";
customer.HomeAddress = new Address
{
Address1 = "North kingdom 15 west",
Address2 = "1 north way",
PostCode = "40990",
City = "George Town",
Country = "Alaska"
};
\_customerRepo.Add(customer);
Assert.IsTrue(\_customerMan.Exists);
Assert.IsNotNull(customer.Id);
// fetch it back
var alreadyAddedCustomer = \_customerRepo.Where(c => c.FirstName == "Bob").Single();
Assert.IsNotNull(alreadyAddedCustomer);
Assert.AreEqual(customer.FirstName, alreadyAddedCustomer.FirstName);
Assert.AreEqual(customer.HomeAddress.Address1, alreadyAddedCustomer.HomeAddress.Address1);
alreadyAddedCustomer.Phone = "10110111";
alreadyAddedCustomer.Email = "dil.bob@fastmail.org";
\_customerRepo.Update(alreadyAddedCustomer);
// fetch by id now
var updatedCustomer = \_customerRepo.GetById(customer.Id);
Assert.IsNotNull(updatedCustomer);
Assert.AreEqual(alreadyAddedCustomer.Phone, updatedCustomer.Phone);
Assert.AreEqual(alreadyAddedCustomer.Email, updatedCustomer.Email);
Assert.IsTrue(\_customerRepo.Exists(c => c.HomeAddress.Country == "Alaska"));
}
[TestMethod]
public void ComplexEntityTest()
{
IRepository<Customer> \_customerRepo = new MongoRepository<Customer>();
IRepository<Product> \_productRepo = new MongoRepository<Product>();
var customer = new Customer();
customer.FirstName = "Erik";
customer.LastName = "Swaun";
customer.Phone = "123 99 8767";
customer.Email = "erick@mail.com";
customer.HomeAddress = new Address
{
Address1 = "Main bulevard",
Address2 = "1 west way",
PostCode = "89560",
City = "Tempare",
Country = "Arizona"
};
var order = new Order();
order.PurchaseDate = DateTime.Now.AddDays(-2);
var orderItems = new List<OrderItem>();
var shampoo = \_productRepo.Add(new Product() { Name = "Palmolive Shampoo", Price = 5 });
var paste = \_productRepo.Add(new Product() { Name = "Mcleans Paste", Price = 4 });
var item1 = new OrderItem { Product = shampoo, Quantity = 1 };
var item2 = new OrderItem { Product = paste, Quantity = 2 };
orderItems.Add(item1);
orderItems.Add(item2);
order.Items = orderItems;
customer.Orders = new List<Order>
{
order
};
\_customerRepo.Add(customer);
Assert.IsNotNull(customer.Id);
Assert.IsNotNull(customer.Orders[0].Items[0].Product.Id);
// get the orders
var theOrders = \_customerRepo.Where(c => c.Id == customer.Id).Select(c => c.Orders).ToList();
var theOrderItems = theOrders[0].Select(o => o.Items);
Assert.IsNotNull(theOrders);
Assert.IsNotNull(theOrderItems);
}
[TestMethod]
public void BatchTest()
{
IRepository<Customer> \_customerRepo = new MongoRepository<Customer>();
var custlist = new List<Customer>(new Customer[] {
new Customer() { FirstName = "Customer A" },
new Customer() { FirstName = "Client B" },
new Customer() { FirstName = "Customer C" },
new Customer() { FirstName = "Client D" },
new Customer() { FirstName = "Customer E" },
new Customer() { FirstName = "Client F" },
new Customer() { FirstName = "Customer G" },
});
//Insert batch
\_customerRepo.Add(custlist);
var count = \_customerRepo.Count();
Assert.AreEqual(7, count);
foreach (Customer c in custlist)
Assert.AreNotEqual(new string('0', 24), c.Id);
//Update batch
foreach (Customer c in custlist)
c.LastName = c.FirstName;
\_customerRepo.Update(custlist);
foreach (Customer c in \_customerRepo)
Assert.AreEqual(c.FirstName, c.LastName);
//Delete by criteria
\_customerRepo.Delete(f => f.FirstName.StartsWith("Client"));
count = \_customerRepo.Count();
Assert.AreEqual(4, count);
//Delete specific object
\_customerRepo.Delete(custlist[0]);
//Test AsQueryable
var selectedcustomers = from cust in \_customerRepo
where cust.LastName.EndsWith("C") || cust.LastName.EndsWith("G")
select cust;
Assert.AreEqual(2, selectedcustomers.ToList().Count);
count = \_customerRepo.Count();
Assert.AreEqual(3, count);
//Drop entire repo
new MongoRepositoryManager<Customer>().Drop();
count = \_customerRepo.Count();
Assert.AreEqual(0, count);
}
[TestMethod]
public void CollectionNamesTest()
{
var a = new MongoRepository<Animal>();
var am = new MongoRepositoryManager<Animal>();
var va = new Dog();
Assert.IsFalse(am.Exists);
a.Update(va);
Assert.IsTrue(am.Exists);
Assert.IsInstanceOfType(a.GetById(va.Id), typeof(Dog));
Assert.AreEqual(am.Name, "AnimalsTest");
Assert.AreEqual(a.CollectionName, "AnimalsTest");
var cl = new MongoRepository<CatLike>();
var clm = new MongoRepositoryManager<CatLike>();
var vcl = new Lion();
Assert.IsFalse(clm.Exists);
cl.Update(vcl);
Assert.IsTrue(clm.Exists);
Assert.IsInstanceOfType(cl.GetById(vcl.Id), typeof(Lion));
Assert.AreEqual(clm.Name, "Catlikes");
Assert.AreEqual(cl.CollectionName, "Catlikes");
var b = new MongoRepository<Bird>();
var bm = new MongoRepositoryManager<Bird>();
var vb = new Bird();
Assert.IsFalse(bm.Exists);
b.Update(vb);
Assert.IsTrue(bm.Exists);
Assert.IsInstanceOfType(b.GetById(vb.Id), typeof(Bird));
Assert.AreEqual(bm.Name, "Birds");
Assert.AreEqual(b.CollectionName, "Birds");
var l = new MongoRepository<Lion>();
var lm = new MongoRepositoryManager<Lion>();
var vl = new Lion();
//Assert.IsFalse(lm.Exists); //Should already exist (created by cl)
l.Update(vl);
Assert.IsTrue(lm.Exists);
Assert.IsInstanceOfType(l.GetById(vl.Id), typeof(Lion));
Assert.AreEqual(lm.Name, "Catlikes");
Assert.AreEqual(l.CollectionName, "Catlikes");
var d = new MongoRepository<Dog>();
var dm = new MongoRepositoryManager<Dog>();
var vd = new Dog();
//Assert.IsFalse(dm.Exists);
d.Update(vd);
Assert.IsTrue(dm.Exists);
Assert.IsInstanceOfType(d.GetById(vd.Id), typeof(Dog));
Assert.AreEqual(dm.Name, "AnimalsTest");
Assert.AreEqual(d.CollectionName, "AnimalsTest");
var m = new MongoRepository<Bird>();
var mm = new MongoRepositoryManager<Bird>();
var vm = new Macaw();
//Assert.IsFalse(mm.Exists);
m.Update(vm);
Assert.IsTrue(mm.Exists);
Assert.IsInstanceOfType(m.GetById(vm.Id), typeof(Macaw));
Assert.AreEqual(mm.Name, "Birds");
Assert.AreEqual(m.CollectionName, "Birds");
var w = new MongoRepository<Whale>();
var wm = new MongoRepositoryManager<Whale>();
var vw = new Whale();
Assert.IsFalse(wm.Exists);
w.Update(vw);
Assert.IsTrue(wm.Exists);
Assert.IsInstanceOfType(w.GetById(vw.Id), typeof(Whale));
Assert.AreEqual(wm.Name, "Whale");
Assert.AreEqual(w.CollectionName, "Whale");
}
[TestMethod]
public void CustomIDTest()
{
var x = new MongoRepository<CustomIDEntity>();
var xm = new MongoRepositoryManager<CustomIDEntity>();
x.Add(new CustomIDEntity() { Id = "aaa" });
Assert.IsTrue(xm.Exists);
Assert.IsInstanceOfType(x.GetById("aaa"), typeof(CustomIDEntity));
Assert.AreEqual("aaa", x.GetById("aaa").Id);
x.Delete("aaa");
Assert.AreEqual(0, x.Count());
var y = new MongoRepository<CustomIDEntityCustomCollection>();
var ym = new MongoRepositoryManager<CustomIDEntityCustomCollection>();
y.Add(new CustomIDEntityCustomCollection() { Id = "xyz" });
Assert.IsTrue(ym.Exists);
Assert.AreEqual(ym.Name, "MyTestCollection");
Assert.AreEqual(y.CollectionName, "MyTestCollection");
Assert.IsInstanceOfType(y.GetById("xyz"), typeof(CustomIDEntityCustomCollection));
y.Delete("xyz");
Assert.AreEqual(0, y.Count());
}
[TestMethod]
public void CustomIDTypeTest()
{
var xint = new MongoRepository<IntCustomer, int>();
xint.Add(new IntCustomer() { Id = 1, Name = "Test A" });
xint.Add(new IntCustomer() { Id = 2, Name = "Test B" });
var yint = xint.GetById(2);
Assert.AreEqual(yint.Name, "Test B");
xint.Delete(2);
Assert.AreEqual(1, xint.Count());
}
[TestMethod]
public void OverrideCollectionName()
{
IRepository<Customer> \_customerRepo = new MongoRepository<Customer>("mongodb://localhost/MongoRepositoryTests", "TestCustomers123");
\_customerRepo.Add(new Customer() { FirstName = "Test" });
Assert.IsTrue(\_customerRepo.Single().FirstName.Equals("Test"));
Assert.AreEqual("TestCustomers123", \_customerRepo.Collection.Name);
Assert.AreEqual("TestCustomers123", ((MongoRepository<Customer>)\_customerRepo).CollectionName);
IRepositoryManager<Customer> \_curstomerRepoManager = new MongoRepositoryManager<Customer>("mongodb://localhost/MongoRepositoryTests", "TestCustomers123");
Assert.AreEqual("TestCustomers123", \_curstomerRepoManager.Name);
}
#region Reproduce issue: https://mongorepository.codeplex.com/discussions/433878
public abstract class BaseItem : IEntity
{
public string Id { get; set; }
}
public abstract class BaseA : BaseItem
{ }
public class SpecialA : BaseA
{ }
[TestMethod]
public void Discussion433878()
{
var specialRepository = new MongoRepository<SpecialA>();
}
#endregion
#region Reproduce issue: https://mongorepository.codeplex.com/discussions/572382
public abstract class ClassA : Entity
{
public string Prop1 { get; set; }
}
public class ClassB : ClassA
{
public string Prop2 { get; set; }
}
public class ClassC : ClassA
{
public string Prop3 { get; set; }
}
[TestMethod]
public void Discussion572382()
{
var repo = new MongoRepository<ClassA>() {
new ClassB() { Prop1 = "A", Prop2 = "B" } ,
new ClassC() { Prop1 = "A", Prop3 = "C" }
};
Assert.AreEqual(2, repo.Count());
Assert.AreEqual(2, repo.OfType<ClassA>().Count());
Assert.AreEqual(1, repo.OfType<ClassB>().Count());
Assert.AreEqual(1, repo.OfType<ClassC>().Count());
}
#endregion
}
}
using MongoRepository;
using MongoDB.Bson.Serialization.Attributes;
namespace MongoRepositoryTests.Entities
{
[CollectionName("AnimalsTest")]
[BsonKnownTypes(typeof(Bird), typeof(Dog))]
public abstract class Animal : Entity { }
[CollectionName("Catlikes")]
[BsonKnownTypes(typeof(Lion), typeof(Cat))]
public abstract class CatLike : Animal { }
[CollectionName("Birds")]
[BsonKnownTypes(typeof(Parrot))]
public class Bird : Animal { }
public class Lion : CatLike { }
public class Cat : CatLike { }
public class Dog : Animal { }
public class Parrot : Bird { }
public class Macaw : Parrot { }
public class Whale : Entity { }
}
using System;
using System.Collections.Generic;
using MongoRepository;
using MongoDB.Bson.Serialization.Attributes;
namespace MongoRepositoryTests.Entities
{
/// <summary>
/// Business Entity for Customer
/// </summary>
[CollectionName("CustomersTest")]
public class Customer : Entity
{
public Customer()
{
}
[BsonElement("fname")]
public string FirstName { get; set; }
[BsonElement("lname")]
public string LastName { get; set; }
public string Email { get; set; }
public string Phone { get; set; }
public Address HomeAddress { get; set; }
public IList<Order> Orders { get; set; }
}
public class Order
{
public DateTime PurchaseDate { get; set; }
public IList<OrderItem> Items;
}
public class OrderItem
{
public Product Product
{
get;
set;
}
public int Quantity { get; set; }
}
public class Address
{
public string Address1 { get; set; }
public string Address2 { get; set; }
public string City { get; set; }
public string PostCode { get; set; }
[BsonIgnoreIfNull]
public string Country { get; set; }
}
public class IntCustomer : IEntity<int>
{
public int Id { get; set; }
public string Name { get; set; }
}
}
using MongoRepository;
using System;
using MongoDB.Bson.Serialization.Attributes;
namespace MongoRepositoryTests.Entities
{
public class CustomIDEntity : IEntity
{
private string \_id;
public string Id
{
get { return \_id; }
set { \_id = value; }
}
}
[CollectionName("MyTestCollection")]
public class CustomIDEntityCustomCollection : CustomIDEntity { }
}
using System;
using MongoRepository;
namespace MongoRepositoryTests.Entities
{
/// <summary>
/// Business Entity for Product
/// </summary>
public class Product : Entity
{
public Product()
{
}
public string Name { get; set; }
public string Description { get; set; }
public decimal Price { get; set; }
}
}
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("MongoRepositoryTests")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("MongoRepositoryTests")]
[assembly: AssemblyCopyright("Copyright © 2011 - 2013")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]
// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("6354bd62-0005-46c7-8412-213aaffbde48")]
// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '\*' as shown below:
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
using System;
using System.Web;
using NPOI.SS.UserModel;
using System.IO;
using Zhidian.Web;
using Microsoft.AspNetCore.Mvc;
using System.Net;
namespace Wholesale.API
{
[TypeFilter(typeof(LoginAttribute))]
public abstract class BaseModule : BaseController
{
public IActionResult Excel(IWorkbook workBook, string name)
{
string filename = HttpUtility.UrlEncode($"{name}{ DateTime.Now:yyyyMMdd}");
//string disposition = "attachment; filename=" + filename + ".xls;filename\*=UTF-8''" + filename + ".xls";
//response.Headers.Add("Content-disposition", disposition);
byte[] buffer = null;
using (MemoryStream ms = new MemoryStream())
{
workBook.Write(ms);
buffer = ms.GetBuffer();
ms.Close();
}
return File(buffer, "application/vnd.ms-excel", filename);
}
public void Redirect303(string url)
{
Response.Redirect(url);
}
public void Redirect500()
{
Response.StatusCode = (int)HttpStatusCode.InternalServerError;
}
}
}
using System;
using System.Linq;
using System.Net;
using System.Reflection;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;
using Zhidian.Common.Configurations.Configs;
using Zhidian.Common.Utility.Results;
using Zhidian.ConfigAgent;
using Zhidian.Identity.Services;
namespace Wholesale.API
{
public class LoginAttribute : ActionFilterAttribute
{
public readonly IIdentityService \_dentity;
public LoginAttribute(IIdentityService dentity)
{
this.\_dentity = dentity;
}
public override void OnActionExecuting(ActionExecutingContext filterContext)
{
var enabledAuth = Configuration<InventoryConfig>.Instance.EnabledAuth;//是否开启授权
if (enabledAuth)
{
var attributes = ((Microsoft.AspNetCore.Mvc.Controllers.ControllerActionDescriptor)filterContext.ActionDescriptor)
.MethodInfo.CustomAttributes.ToList();
//如果是不需要授权的，直接跳过
if (attributes.Count(it => it.AttributeType.Name.Contains("UnAuthAttribute")) > 0)
return;
//如果包含授权特性则进行，授权验证
try
{
//登录验证
\_dentity.GetCurrentUserOrThrow();
if (attributes.Count(it => it.AttributeType.Name.Contains("AuthAttribute")) > 0)
{
var pUser = \_dentity.GetCurrentUserOrThrow();
CustomAttributeData attribute = attributes.Where(it => it.AttributeType.Name.Contains("AuthAttribute")).FirstOrDefault() as CustomAttributeData;
if (attribute != null)
{
var result = \_dentity.ExistAuthority(Convert.ToString(attribute.ConstructorArguments[0].Value));
if (!result.IsSuccessed)
{
filterContext.Result = new JsonResult(result);
return;
}
}
}
}
catch (Exception)
{
filterContext.Result = new JsonResult(Result.LoginError);
return;
}
if (filterContext.HttpContext.Response != null &&
filterContext.HttpContext.Response.StatusCode == (int)HttpStatusCode.Unauthorized)
{
filterContext.Result = new JsonResult(Result.LoginError);
}
}
}
}
}
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
namespace Wholesale.API
{
public class Program
{
public static void Main(string[] args)
{
CreateWebHostBuilder(args).Build().Run();
}
public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
WebHost.CreateDefaultBuilder(args)
.UseStartup<Startup>();
}
}
using System;
using System.Linq;
using System.IO;
using System.Runtime.Loader;
using System.Reflection;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Configuration;
using Zhidian.Cache.IocExtensions;
using Zhidian.Common;
using Zhidian.Common.Configurations;
using Zhidian.Common.Configurations.Configs;
using Zhidian.Common.Extensions;
using Zhidian.Common.Infrastructure;
using Zhidian.Common.Infrastructure.Startups;
using Zhidian.ConfigAgent;
using Zhidian.Log.Extensions;
using Zhidian.MQ.Configs;
using Zhidian.MQ.Extensions;
using Zhidian.Web.Extensions;
using Wholesale.Common.Configurations;
using Autofac;
using Autofac.Extensions.DependencyInjection;
using Wholesale.Common;
using Wholesale.DataAccess.Redis;
using Wholesale.Services;
using Zhidian.Common.Utility.Extensions;
using Zhidian.MQ;
using Zhidian.MQ.ActiveMQ;
using Zhidian.MQ.RabbitMQ;
using Zhidian.UserCenter;
using Zhidian.UserCenter.Configuration;
namespace Wholesale.API
{
public class Startup
{
public Startup(IConfiguration configuration)
{
Configuration = configuration;
ConfigurationManager.Configuration = configuration;
}
public IContainer ApplicationContainer { get; private set; }
public IConfiguration Configuration { get; }
// This method gets called by the runtime. Use this method to add services to the container.
public IServiceProvider ConfigureServices(IServiceCollection services)
{
StaticHttpContextExtend.AddHttpContextAccessor(services);
services.AddMvc().AddMvcOptions(it =>
{
it.InputFormatters.Clear();
it.InputFormatters.Add(new JsonInputFormater(pageIndexBeginZero: true));
it.OutputFormatters.Clear();
it.OutputFormatters.Add(new JsonOutputFormater());
});
var identityConfig = Configuration<UserCenterConfig>.Instance;
#if DEBUG
//identityConfig.IdentityHost = "http://localhost:5000";
#endif
services.AddUserCenter(options =>
{
options.ClientId = identityConfig.ClientId;// "dht.saas";
options.ClientSecret = identityConfig.ClientSecret; //"secret";
options.ApiName = identityConfig.ApiName;// "DHT.SAAS.Api";
options.ApiSecret = identityConfig.ApiSecret;
options.Scopes = identityConfig.Scopes;// new[] { "dht.saas.webapi" };
options.IdentityHost = identityConfig.IdentityHost;
});
var builder = new ContainerBuilder();
#region IOC全量注册
string[] assemblyNameName = { "Wholesale" };
IocManager.Initialize(builder, assemblyNameName);
#endregion
var assembiles = AppDomain.CurrentDomain.GetAssemblies();
var types = assembiles.SelectMany(s => s.GetTypes())
.Where(w => typeof(IExportToExcelProvider).IsAssignableFrom(w) && !w.IsInterface && !w.IsAbstract);
foreach (var type in types)
{
var attrs = type.GetAttributes<ProviderNameAttribute>();
foreach (var attr in attrs)
{
builder.RegisterType(type).Named<IExportToExcelProvider>(attr.Name);
}
}
builder.Populate(services);
var container = builder.Build();
this.ApplicationContainer = container;
#region 调用组件的Use扩展方法将对象注册到Autofac IOC容器
#region 启用Zhidian.Common
var messageConfig = Configuration<MessageConfig>.Instance;
var inventoryConfig = Configuration<InventoryConfig>.Instance;
container.UseCommon(messageConfig, inventoryConfig);
#endregion
#region Redis注册
var config = Configuration<RedisConfig>.Instance;
container.UseRedisCache(config.Erp);
container.BindReids<PriceRedis>(config.PriceSetting);
#endregion
#region Log MQ配置注入
var logConfig = Configuration<ActiveMQConfig>.Instance.Log;
var operationConfig = Configuration<ActiveMQConfig>.Instance.OperationLog;
container.UseMQLog(logConfig, operationConfig);
#endregion
#region webpush注入方法
var WebPushConfig = Configuration<WebPushConfig>.Instance;
container.UseWebMQ(WebPushConfig);
#endregion
var activeMQConfig = Configuration<ActiveMQConfig>.Instance;
container.Update(ctx =>
{
ctx.Register<IMQContext>(c =>
{
var content = new ActiveMQContext(activeMQConfig.WholesaleStockNotice);
return content;
}).SingleInstance();
ctx.Register(c =>
{
var content = new RabbitMQContext(activeMQConfig.CanalOpenApiNotice);
return content;
}).Named<IMQContext>("rabbitmq\_orderweal").SingleInstance();
});
#region mongodb日志仓储对象注入
var mongoConfig = Configuration<MongoConfig>.Instance;
container.UseMongoLog(mongoConfig.Log, mongoConfig.OperationLog);
#endregion
#endregion
#region 反射加载程序集dll到运行时，.net core 2.0不支持引用第三方DLL直接加载 TODO 升级2.0.3 或者Nuget加载
Assembly entry = Assembly.GetEntryAssembly();
string dir = Path.GetDirectoryName(entry.Location);
var dllPath = dir + "\\MongoRepository.Net45.dll";
AssemblyLoadContext.Default.LoadFromAssemblyPath(dllPath);
#endregion
StartupExecute.Excute(container);
return new AutofacServiceProvider(this.ApplicationContainer);
}
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
app.UseStaticHttpContext();
// 开发模式显示异常页
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}
app.UseAuthentication();
// 跨域策略
app.UseCors(builder => builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader().AllowCredentials());
app.UseMiddleware<RequestBodyBufferMiddleware>();
app.UseMvc();
}
}
}
using System;
using System.Collections.Generic;
using System.DrawingCore.Imaging;
using System.IO;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DataAccess.Database;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Wholesale.Services.Acount;
using Wholesale.Services.Storages;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.Services;
using Zhidian.Java.DTO;
using Zhidian.Java.Model;
using Zhidian.Java.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class AccountController : BaseController
{
private readonly IWholesaleAccountService \_wholesaleAccountService;
private readonly IAccountWithdrawalService \_with;
private readonly IAccountDetailService \_iaccount;
private readonly IPointDetailService \_ipoint;
private readonly IStorageService \_istorage;
private readonly IBankCardService \_service;
private readonly IIdentityService \_dentity;
private readonly IUserStorageService \_userStorage;
private readonly ISMSCode \_smsCode;
public AccountController(IIdentityService identity,
IBankCardService service,
IAccountDetailService iaccount,
IPointDetailService ipoint,
IStorageService istorage,
IUserStorageService userStorage,
IPayPasswordService payservice,
ISMSCode smsCode,
WholesaleDatabase db,
IAccountWithdrawalService with,
IWholesaleAccountService wholesaleAccountService)
{
\_dentity = identity;
\_service = service;
\_iaccount = iaccount;
\_ipoint = ipoint;
\_istorage = istorage;
\_smsCode = smsCode;
\_with = with;
\_userStorage = userStorage;
\_wholesaleAccountService = wholesaleAccountService;
}
/// <summary>
/// 撤销申请号为空的提现申请
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[UnAuth]
[HttpPost]
[Route("/account/withdrawdeposit/cancel")]
public dynamic CancelAccountWithdrawDeposit([FromBody] CancelAccountWithdrawDepositInput cancelAccountWithdrawDeposit)
{
return \_with.CancelAccountWithdrawDeposit(cancelAccountWithdrawDeposit);
}
[UnAuth]
[HttpGet]
[Route("/account/info")]
public dynamic GetAccountInfo(string phone)
{
return \_wholesaleAccountService.GetFullAccountInfo(phone);
}
/// <summary>
/// 分页获取提现记录
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/withdrawdeposit/list")]
public dynamic GetWithdrawDeposits([FromBody] PagedCondition pager, [FromBody] GetWithdrawDepositsInput getWithdrawDeposits)
{
var us = \_dentity.GetCurrentUserOrThrow();
getWithdrawDeposits.MallUserId = us.MallUserId;
var result = \_with.GetWithdrawDeposits(pager, getWithdrawDeposits);
return result.Map<WithdrawDepositDTO, WithdrawDepositOutput>();
}
/// <summary>
/// 获取账户流水通过Java接口获取
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/list")]
public dynamic GetJavaList([FromBody] PagedCondition pager, [FromBody]AccountSearch accountSearch)
{
var result = \_iaccount.GetJavaList(pager, accountSearch);
return result;
}
/// <summary>
/// 获取积分记录
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/pointlist")]
public dynamic GetPointDetailList([FromBody] PagedCondition pager, [FromBody]AccountSearch accountSearch)
{
var result = \_ipoint.GetPointDetailList(pager, accountSearch);
return result.Map<PointDetail, PointDetailOutput>();
}
/// <summary>
/// 结算历史满足预期积分的订单进行预期积分计算
/// </summary>
/// <returns></returns>
[HttpPost]
[Route("/account/settleexpectpoint")]
public dynamic SettleExpectPoint()
{
var result = \_ipoint.SettleExpectPoint();
return result;
}
/// <summary>
/// 获取用户汇总积分信息
/// </summary>
/// <returns></returns>
[HttpPost]
[Route("/account/userpoint")]
public dynamic GetUserPoint()
{
var result = \_ipoint.GetUserPoint();
return result;
}
/// <summary>
/// 通过接口获取用户可以积分列表
/// </summary>
/// <param name="page"></param>
/// <param name="search"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/cashpointlist")]
public dynamic GetCashPointList([FromBody] PagedCondition paged, [FromBody] AccountSearch search)
{
var result = \_ipoint.GetCashPointList(paged, search);
return result;
}
/// <summary>
/// 查询销售预期积分
/// </summary>
/// <returns></returns>
[HttpPost]
[Route("/account/expectpointlist")]
public dynamic GetSaleExpectPointList([FromBody] PagedCondition pager, [FromBody] AccountSearch search)
{
var result = \_ipoint.GetSaleExpectPointList(pager, search);
return result;
}
/// <summary>
/// 获取用户钱包
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("/account/purse")]
public dynamic UserPurse()
{
Result<PurseDTO> result = \_iaccount.UserPurse();
return result;
}
/// <summary>
/// 获用户取银行卡列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/get")]
public dynamic GetUserBankCard()
{
var result = new Result();
try
{
string userId = \_dentity.GetCurrentUserOrThrow().MallUserId;
var data = \_service.GetUserBankCardMobile(userId, "007");
result.Successed(data, "获用户取银行卡列表成功");
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 解绑银行卡 000 ：成功
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/remove")]
public dynamic RemoveUserBankCard([FromBody]CardIdInput cardId)
{
var result = new Result();
try
{
string userId = \_dentity.GetCurrentUserOrThrow().MallUserId;
var data = \_service.RemoveUserBankCardMobile(userId, cardId.cardId, "007");
result.Successed(data, "解绑银行卡成功");
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 查询可提金额信息
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/querymoney")]
public dynamic QueryMoneyAndBankCard()
{
var result = new Result();
try
{
result = \_with.QueryMoneyAndBankCard();
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 提现
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/withdraw")]
public dynamic WithdrawDeposit([FromBody] applyAmountInput applyAmount)
{
var result = new Result();
try
{
var user = \_dentity.GetCurrentUserOrThrow();
var data = \_with.WithdrawDepositMobile(user.MallUserId, user.UserName, user.StroageId, applyAmount.cardId, applyAmount.applyAmount, "007");
result.Successed(data, "提现成功");
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 添加银行卡
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/add")]
public dynamic AddBankCard([FromBody]AddBankCardInput addBankCard)
{
var result = new Result();
try
{
string userId = \_dentity.GetCurrentUserOrThrow().MallUserId;
var data = \_service.AddBankCardMobile(addBankCard, userId, "007");
result.Successed(data, "添加银行卡成功");
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 查询卡号对应银行卡信息
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/check")]
public dynamic CheckBankCard([FromBody] cardNoInput cardNo)
{
var result = new Result();
try
{
string userId = \_dentity.GetCurrentUserOrThrow().MallUserId;
var data = \_service.CheckBankCardMobile(userId, cardNo.cardNo, "007");
result.Successed(data, "查询卡号对应银行卡信息成功");
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 获取银行卡列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet]
[Route("/account/bank/getlist")]
public dynamic GetListBankCard()
{
var result = new Result();
try
{
string userId = \_dentity.GetCurrentUserOrThrow().MallUserId;
var data = \_service.GetListBankCardMobile(userId, "null", "007");
List<BankListData> b = new List<BankListData>();
foreach (var item in data)
{
b.AddRange(item.Value);
}
result.Successed(b, "获取银行卡列表成功");
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 提现记录
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/record")]
public dynamic WithdrawalRecordBankCard([FromBody] WithdrawalRecordWebInput withdrawalRecordWeb)
{
var result = new Result<PagedListAndSum<WithdrawalRecordList>>();
try
{
result = \_with.WithdrawalRecordBankCard(withdrawalRecordWeb);
}
catch (Exception ex)
{
result.Errored(ex.Message);
}
return result;
}
/// <summary>
/// 验证支付密码 web端
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/paypassword")]
public dynamic VerifyPayPassword([FromBody] payPasswordInput payPassword)
{
var result = new Result();
try
{
string userId = \_dentity.GetCurrentUserOrThrow().MallUserId;
var data = \_service.VerifyPayPassword(payPassword.password, userId);
result.Successed(data, "验证支付密码成功");
}
catch (Exception ex)
{
result.Errored("验证支付密码失败" + ex.Message);
}
return result;
}
/// <summary>
/// 提现加验证支付密码
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/withdrawpassword")]
public dynamic WithdrawPassword([FromBody] payPasswordInput payPassword, [FromBody]applyAmountInput applyAmount)
{
var result = new Result();
try
{
var user = \_dentity.GetCurrentUserOrThrow();
var data = \_service.VerifyPayPassword(payPassword.password, user.MallUserId);
var wata = \_with.WithdrawDepositMobile(user.MallUserId, user.UserName, user.StroageId, applyAmount.cardId, applyAmount.applyAmount, "007");
result.Successed(wata, "提现成功");
}
catch (Exception ex)
{
result.Errored("提现失败" + ex.Message);
}
return result;
}
//v3
/// <summary>
/// 获取图片验证码
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[UnAuth]
[HttpGet]
[Route("/account/bank/GetImageCode")]
public ActionResult GetImageCode(GetImageCode getImageCode)
{
var bmp = \_smsCode.GetImageCode(getImageCode);
MemoryStream ms = new MemoryStream();
bmp.Save(ms, ImageFormat.Jpeg);
return File(ms.ToArray(), "image/jpeg");
}
/// <summary>
/// 获取短信验证码
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/GetSMSCode")]
public dynamic GetSMSCode([FromBody] GetSMSInput input)
{
var result = new Result();
try
{
result = \_with.GetSMSCode(input);
}
catch (Exception ex)
{
result.Errored("获取短信验证码" + ex.Message);
}
return result;
}
/// <summary>
/// 提现加验证手机短信加移除短信缓存
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/WithdrawPhone")]
public dynamic WithdrawPhone([FromBody] CheckSMSInput cs, [FromBody]applyAmountInput ap)
{
var result = new Result();
try
{
result = \_with.WithdrawPhone(cs, ap);
}
catch (Exception ex)
{
result.Errored("提现失败" + ex.Message);
}
return result;
}
/// <summary>
/// 验证短信验证码
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/bank/CheckSMSCode")]
public dynamic CheckSMSCode([FromBody] CheckSMSInput cs)
{
var result = new Result();
try
{
result = \_with.CheckSMSCode(cs);
}
catch (Exception ex)
{
result.Errored("验证短信验证码失败" + ex.Message);
}
return result;
}
/// <summary>
/// 获取提现记录
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost]
[Route("/account/withdrawdeposit/record")]
public dynamic GetWithdrawDepositById([FromBody] IdInput input)
{
\_with.FirstCheckFailedWithdrawDeposit(input.Id); //执行一次首次核对
var result = \_with.GetWithdrawDepositById(input);
return result.Map<WithdrawDepositDTO, WithdrawDepositOutput>();
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO.Activities;
using Wholesale.Services.Activitys;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.Services;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class ActivitieController : BaseController
{
private readonly IActivityService \_activityService;
private readonly IIdentityService \_dentity;
private readonly IBookingActivityService \_bookingActivityService;
public ActivitieController(IActivityService activityService,
IIdentityService identity,
IBookingActivityService bookingActivityService)
{
\_activityService = activityService;
\_dentity = identity;
\_bookingActivityService = bookingActivityService;
}
#region 调价活动
/// <summary>
/// 获取调价活动列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看调价活动")]
[HttpGet]
[Route("/activitie/pricing/list")]
public dynamic GetActivityList(PagedCondition page, ActivitySearchInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.GetActivityList(page, condition, user);
return result;
}
/// <summary>
/// 调价活动新增
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("新增调价")]
[HttpPost]
[Route("/activitie/pricing")]
public dynamic InsertActivitySku([FromBody]ActivitySkuInsertInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.InsertActivitySku(condition, user);
return result;
}
/// <summary>
/// 编辑调价活动
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑调价")]
[HttpPut]
[Route("/activitie/pricing")]
public dynamic EditActivity([FromBody]ActivitySkuInsertInput condition, int id)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.EditActivity(condition, user);
return result;
}
/// <summary>
/// 调价获取商品
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看调价活动")]
[HttpGet]
[Route("/activitie/pricing/goods")]
public dynamic AccessGoods(PagedCondition page, ActivityGoodsInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.AccessGoods(page, condition, user);
return result;
}
/// <summary>
/// 查看调价活动
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看调价活动")]
[HttpGet]
[Route("/activitie/pricing")]
public dynamic ViewActivity(int id)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.ViewActivity(id, user);
return result;
}
/// <summary>
/// 调价活动 0启用 1暂停 2删除
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑调价")]
[HttpPut]
[Route("/activitie/pricingforbade")]
public dynamic ActivityChangeStates([FromBody] ActivitieForbadeInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.ActivityChangeStates(condition, user);
return result;
}
/// <summary>
/// 批量刷缓存
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看调价活动")]
[HttpGet]
[Route("/activitie/BatchSetCache")]
public dynamic BatchSetCache(string type)
{
var result = \_activityService.BatchSetCache(type);
return result;
}
#endregion
#region 满减活动
/// <summary>
/// 新增满减活动
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("新增满减")]
[HttpPost]
[Route("/activitie/subtraction")]
public dynamic InsertActivitySubtraction([FromBody]ActivitySkuInsertInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.InsertActivitySubtraction(condition, user);
return result;
}
/// <summary>
/// 满减活动 0启用 1暂停 2删除
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑满减")]
[HttpPut]
[Route("/activitie/subtractionforbade")]
public dynamic ActivityChangeStatesSubtraction([FromBody]ActivitieForbadeInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.ActivityChangeStates(condition, user);
return result;
}
/// <summary>
/// 查看满减活动
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看满减活动")]
[HttpGet]
[Route("/activitie/subtraction")]
public dynamic ViewActivitySubtraction(int id)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.ViewActivitySubtraction(id, user);
return result;
}
/// <summary>
/// 获取满减活动列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看满减活动")]
[HttpGet]
[Route("/activitie/subtraction/list")]
public dynamic GetActivitySubtractionList(PagedCondition page, ActivitySearchInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.GetActivitySubtractionList(page, condition, user);
return result;
}
/// <summary>
/// 编辑满减活动
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑满减")]
[HttpPut]
[Route("/activitie/subtraction")]
public dynamic EditActivitySubtraction([FromBody] ActivitySkuInsertInput condition)
{
var user = \_dentity.GetCurrentUserOrThrow();
var result = \_activityService.EditActivitySubtraction(condition, user);
return result;
}
#endregion
#region 预定活动
/// <summary>
/// 预订活动列表
/// </summary>
[Auth("查看预订活动列表")]
[HttpGet]
[Route("/activitie/booking/list")]
public dynamic BookingList(PagedCondition pager, BookingActivitySearch input)
{
return \_bookingActivityService.BookingList(input, pager);
}
/// <summary>
/// 选择活动商品
/// </summary>
[HttpGet]
[Route("/activitie/booking/SkuList")]
public dynamic BookingSkuList(PagedCondition pager, BookingSkuSearch input)
{
return \_bookingActivityService.BookingSkuList(input, pager);
}
/// <summary>
/// 获取预订活动信息
/// </summary>
[HttpPost]
[Route("/activitie/booking/info")]
public dynamic BookingInfo([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingInfo(input.ActivityId);
}
/// <summary>
/// 启用预订活动
/// </summary>
[HttpPost]
[Route("/activitie/booking/enabled")]
public dynamic BookingEnabled([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingEnabled(input.ActivityId);
}
/// <summary>
/// 暂停预订活动
/// </summary>
[HttpPost]
[Route("/activitie/booking/pause")]
public dynamic BookingPause([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingPause(input.ActivityId);
}
/// <summary>
/// 终止预订活动
/// </summary>
[HttpPost]
[Route("/activitie/booking/abort")]
public dynamic BookingAbort([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingAbort(input.ActivityId);
}
/// <summary>
/// 删除预订活动
/// </summary>
[HttpPost]
[Route("/activitie/booking/delete")]
public dynamic BookingDelete([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingDelete(input.ActivityId);
}
/// <summary>
/// 新增预订活动
/// </summary>
[HttpPost]
[Route("/activitie/booking/add")]
public dynamic BookingAdd([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingAdd(input);
}
/// <summary>
/// 编辑预订活动
/// </summary>
[HttpPost]
[Route("/activitie/booking/edit")]
public dynamic BookingEdit([FromBody]BookingActivityInput input)
{
return \_bookingActivityService.BookingEdit(input);
}
/// <summary>
/// 销售统计（商品）
/// </summary>
[HttpGet]
[Route("/activitie/booking/producttotal")]
public dynamic BookingProductTotal(PagedCondition pager, BookingProductTotalSearch input)
{
return \_bookingActivityService.BookingProductTotal(input, pager);
}
/// <summary>
/// 销售统计（订单）
/// </summary>
[HttpGet]
[Route("/activitie/booking/ordertotal")]
public dynamic BookingOrderTotal(PagedCondition pager, BookingOrderTotalSearch input)
{
return \_bookingActivityService.BookingOrderTotal(input, pager);
}
/// <summary>
/// 预订活动列表导出任务
/// </summary>
[HttpPost]
[Route("/activitie/booking/ListExportTask")]
public dynamic BookingListExportTask([FromBody]HeaderItemInput header, [FromBody]BookingActivitySearch input)
{
return \_bookingActivityService.BookingListExportTask(input, header);
}
/// <summary>
/// 销售统计（商品）导出任务
/// </summary>
[HttpPost]
[Route("/activitie/booking/ProductTotalExportTask")]
public dynamic BookingProductTotalExportTask([FromBody]HeaderItemInput header, [FromBody]BookingProductTotalSearch input)
{
return \_bookingActivityService.BookingProductTotalExportTask(input, header);
}
/// <summary>
/// 销售统计（订单）导出任务
/// </summary>
[HttpPost]
[Route("/activitie/booking/OrderTotalExportTask")]
public dynamic BookingOrderTotalExportTask([FromBody]HeaderItemInput header, [FromBody]BookingOrderTotalSearch input)
{
return \_bookingActivityService.BookingOrderTotalExportTask(input, header);
}
#endregion
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services.Ad;
using Wholesale.Services.Storages;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
namespace Wholesale.API.Controllers
{
public class AdController : BaseModule
{
private readonly IAdService \_service;
private readonly IStorageService \_storageService;
public AdController(IAdService service, IStorageService storageService)
{
\_service = service;
\_storageService = storageService;
}
/// <summary>
/// 获取企业介绍
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
[HttpGet]
[Route("/ad/storage/myselfStorage")]
public dynamic GetMySelfStorage(string id)
{
return \_storageService.GetMySelfStorage(id);
}
/// <summary>
/// 获取企业资质
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
[HttpGet]
[Route("/ad/storage/shopCompanyProfile")]
public dynamic ShopCompanyProfile(string id)
{
return \_storageService.ShopCompanyProfile(id); ;
}
/// <summary>
/// 更新企业资质
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost]
[Route("/ad/storage/updateShopCompanyProfile")]
public dynamic UpdateShopCompanyProfile([FromBody] ShopCompanyProfileOutputDto input)
{
return \_storageService.UpdateShopCompanyProfile(input); ;
}
/// <summary>
/// v2.1-选择店铺数据接口
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost]
[Route("/ad/storage/select")]
public dynamic AdStorageSelect([FromBody]PagedCondition page, [FromBody]AdStorageSelectInput input)
{
return \_service.AdStorageSelect(page, input);
}
/// <summary>
/// v2.1-选择商品接口
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost]
[Route("/ad/product/select")]
public dynamic AdProductSelect([FromBody]PagedCondition page, [FromBody]AdProductSelectInput input)
{
return \_service.AdProductSelect(page, input);
}
/// <summary>
/// v2.1-banner列表
/// </summary>
/// <param name="input"></param>
/// <param name="page"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/banner/list")]
public dynamic AdBannerList([FromBody]PagedCondition page, [FromBody]AdBannerSearchInput input)
{
return \_service.GetPageList(page, input);
}
/// <summary>
/// v2.1-banner新增
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/banner/add")]
public dynamic AdBannerAdd([FromBody]AdBannerInput input)
{
return \_service.AdDetailAdd(input);
}
/// <summary>
/// v2.1-banner更新
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/banner/update")]
public dynamic AdBannerUpdate([FromBody]AdBannerInput input)
{
return \_service.AdDetailUpdate(input);
}
/// <summary>
/// v2.1-banner详情
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/banner/detail")]
public dynamic AdBannerDetail([FromBody]IdInput input)
{
return \_service.AdDetailBannerDetail(input);
}
/// <summary>
/// v2.1-优质商家列表
/// </summary>
/// <param name="input"></param>
/// <param name="page"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/storage/list")]
public dynamic AdStorageList([FromBody]PagedCondition page, [FromBody]AdStorageSearchInput input)
{
return \_service.GetPageList(page, input);
}
/// <summary>
/// 2.1-优质商家新增
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/storage/add")]
public dynamic AdStorageAdd([FromBody]AdStorageInput input)
{
return \_service.AdDetailAdd(input);
}
/// <summary>
/// 2.1-优质商家更新
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/storage/update")]
public dynamic AdStorageUpdate([FromBody]AdStorageInput input)
{
return \_service.AdDetailUpdate(input);
}
/// <summary>
/// 2.1-删除
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/banner/delete")]
[HttpPost("/ad/storage/delete")]
[HttpPost("/ad/category/delete")]
public dynamic AdStorageDetele([FromBody]IdInput input)
{
return \_service.AdDetialDelete(input);
}
/// <summary>
/// 2.1-排序
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/banner/sort")]
[HttpPost("/ad/storage/sort")]
[HttpPost("/ad/category/sort")]
public dynamic AdDetailUpdateSort([FromBody]AdSortInput input)
{
return \_service.AdDetailUpdateSort(input);
}
/// <summary>
/// 2.1-优质商家详情
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/storage/detail")]
public dynamic AdStorageDetail([FromBody]IdInput input)
{
return \_service.AdDetailStorageDetail(input);
}
/// <summary>
/// v2.1-热门分类列表
/// </summary>
/// <param name="input"></param>
/// <param name="page"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/list")]
public dynamic AdCategoryList([FromBody]PagedCondition page, [FromBody]AdCategorySearchInput input)
{
return \_service.GetPageList(page, input);
}
/// <summary>
/// 2.1-热门分类新增
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/add")]
public dynamic AdCategoryAdd([FromBody]AdCategoryInput input)
{
return \_service.AdDetailAdd(input);
}
/// <summary>
/// 2.1-热门分类更新
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/update")]
public dynamic AdCategoryUpdate([FromBody]AdCategoryInput input)
{
return \_service.AdDetailUpdate(input);
}
/// <summary>
/// 2.1-热门分类详情
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/detail")]
public dynamic AdCategoryDetail([FromBody]IdInput input)
{
return \_service.AdDetailCategoryDetail(input);
}
/// <summary>
/// v2.1-分类banner列表
/// </summary>
/// <param name="page"></param>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/banner/list")]
public dynamic AdCategoryBannerList([FromBody]PagedCondition page, [FromBody]AdCategoryBannerSearchInput input)
{
return \_service.GetPageList(page, input);
}
/// <summary>
/// v2.1-分类banner新增
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/banner/add")]
public dynamic AdCategoryBannerAdd([FromBody]AdCategoryBannerInput input)
{
return \_service.AdDetailListAdd(input);
}
/// <summary>
/// v2.1-分类banner更新
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/banner/update")]
public dynamic AdCategoryBannerUpdate([FromBody]AdCategoryBannerInput input)
{
return \_service.AdDetailListUpdate(input);
}
/// <summary>
/// v2.1-分类banner删除
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/banner/delete")]
public dynamic AdCategoryBannerDelete([FromBody]IdInput input)
{
return \_service.AdDetailListDelete(input);
}
/// <summary>
/// v2.1-分类banner排序
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/banner/sort")]
public dynamic AdCategoryBannerSort([FromBody]AdSortInput input)
{
return \_service.AdDetailListSort(input);
}
/// <summary>
/// 2.1-分类banner详情
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/banner/detail")]
public dynamic AdCategoryBannerDetail([FromBody]IdInput input)
{
return \_service.AdDetailListBannerDetail(input);
}
/// <summary>
/// v2.1-分类商家列表
/// </summary>
/// <param name="page"></param>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/storage/list")]
public dynamic AdCategoryStorageList([FromBody]PagedCondition page, [FromBody]AdCategoryStorageSearchInput input)
{
return \_service.GetPageList(page, input);
}
/// <summary>
/// v2.1-分类商家新增
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/storage/add")]
public dynamic AdCategoryStorageAdd([FromBody]AdCategoryStorageInput input)
{
return \_service.AdDetailListAdd(input);
}
/// <summary>
/// v2.1-分类商家更新
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/storage/update")]
public dynamic AdCategoryStorageUpdate([FromBody]AdCategoryStorageInput input)
{
return \_service.AdDetailListUpdate(input);
}
/// <summary>
/// v2.1-分类商家删除
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/storage/delete")]
public dynamic AdCategoryStorageDelete([FromBody]IdInput input)
{
return \_service.AdDetailListDelete(input);
}
/// <summary>
/// v2.1-分类商家排序
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/storage/sort")]
public dynamic AdCategoryStorageSort([FromBody]AdSortInput input)
{
return \_service.AdDetailListSort(input);
}
/// <summary>
/// 2.1-分类storage详情
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("广告页")]
[HttpPost("/ad/category/storage/detail")]
public dynamic AdCategoryStorageDetail([FromBody]IdInput input)
{
return \_service.AdDetailListStorageDetail(input);
}
}
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Wholesale.App.Model.DTO;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
namespace Wholesale.API.Controllers
{
public class AddressController : BaseModule
{
public IAddressService \_iIAddressService;
public AddressController(IAddressService iIAddressService)
{
this.\_iIAddressService = iIAddressService;
}
/// <summary>
/// 收货列表
/// </summary>
/// <returns></returns>
[HttpPost("address/receviceAddrssList")]
public dynamic ReceiveAddressList([FromBody]PagedCondition pager, [FromBody]ReceiveAddressSearceInput searceInput)
{
return \_iIAddressService.GetReceiveAddressPage(pager, searceInput);
}
/// <summary>
/// 根据地址id查询收货地址
/// </summary>
/// <returns></returns>
[HttpPost("address/id")]
public dynamic GetReceiveAddressForId([FromBody]IdInput input)
{
return \_iIAddressService.GetReceiveAddressFor(input.Id);
}
/// <summary>
/// 收货列表
/// </summary>
/// <returns></returns>
[HttpPost("address/default")]
public dynamic GetDefaultReceiveAddress()
{
return \_iIAddressService.GetDefaultAddress();
}
/// <summary>
/// 新增收货列表
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost("address/addOrUpdateAddress")]
public dynamic AddOrUpdateAddress([FromBody]ReceiveAddressDto dto)
{
return \_iIAddressService.AddOrUpdateReceiveAddress(dto);
}
/// <summary>
/// 删除收货列表
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost("address/deleteAddress")]
public dynamic DeleteAddress([FromBody]IdInput dto)
{
return \_iIAddressService.DeleteReceiveAddress(dto);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class AdminHomeController : BaseController
{
IAdminHomeService \_service;
public AdminHomeController(IAdminHomeService service)
{
\_service = service;
}
[HttpPost]
[Route("/admin/home/init")]
public dynamic Init([FromBody] DTO.AdminOrderStatisticsDayViewInput input)
{
return \_service.StatisticsByDateRange(input.BeginDate.Value, input.EndDate.Value);
}
/// <summary>
/// 管理员首页统计
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost]
[Route("/admin/home/total")]
public dynamic GetOrderStatistics()
{
return \_service.GetOrderStatisticsFromCache();
}
/// <summary>
/// 管理首页统计（按日统计，线图）
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost]
[Route("/admin/home/totaldayview")]
public dynamic GetOrderStatisticsDayView([FromBody]DTO.AdminOrderStatisticsDayViewInput input)
{
return \_service.GetOrderStatisticsDayView(input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class AppVersionController : BaseController
{
private readonly IAppVersionService \_appversion;
public AppVersionController(IAppVersionService appversion)
{
this.\_appversion = appversion;
}
/// <summary>
/// 获取app版本管理列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看APP版本")]
[HttpGet]
[Route("/version/list")]
public dynamic GetAppVersionList(PagedCondition pager, AppVersionInput input)
{
Result<PagedList<AppVersion>> result = \_appversion.GetAppVersionList(pager, input);
return result;
}
/// <summary>
/// 获取app版本管理
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("查看APP版本")]
[HttpGet]
[Route("/version")]
public dynamic GetAppVersion(int id)
{
var result = \_appversion.GetAppVersion(id);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
/// <summary>
/// 新增app版本管理
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("新增APP版本")]
[HttpPost]
[Route("/version")]
public dynamic InsertAppVersion([FromBody]AppVersionInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_appversion.InsertAppVersion(input);
return result;
}
/// <summary>
/// 编辑app版本管理
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("编辑APP版本")]
[HttpPut]
[Route("/version")]
public dynamic UpdateAppVersion([FromBody] AppVersionInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_appversion.UpdateAppVersion(input);
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class AssistBuyController : BaseController
{
IAssistBuyService \_service;
public AssistBuyController(IAssistBuyService service)
{
\_service = service;
}
/// <summary>
/// 获取商品列表
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost("/assistbuy/search")]
public dynamic AssistBuySearch([FromBody]AssistBuySearchInput input, [FromBody] PagedCondition page)
{
return \_service.SearchSku(page, input);
}
/// <summary>
/// 获取商品列表
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost("/assistbuy/categories")]
public dynamic AssistBuyCategories([FromBody]StorageIdInput input)
{
return \_service.GetCategoryList(input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Identity.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class AuditController : BaseController
{
private readonly IAuditService \_audit;
private readonly IIdentityService \_dentity;
public AuditController(IAuditService audit, IIdentityService dentity)
{
this.\_audit = audit;
this.\_dentity = dentity;
}
/// <summary>
/// 获取用户待办/已办消息
/// </summary>
/// <param name="pager"></param>
/// <param name="msgType"></param>
/// <param name="msgStatus"></param>
/// <returns></returns>
[HttpGet]
[Route("/audit/queryNoProcessMsg")]
public dynamic QueryNoProcessMsg(PagerModel pager, int msgType, int msgStatus)
{
var loginUser = \_dentity.GetCurrentUser();
var result = \_audit.queryNoProcessMsg(loginUser.UserName, loginUser.RealName, pager, msgType, msgStatus);
return result;
}
/// <summary>
/// 审核
/// </summary>
/// <param name="batchFlow"></param>
/// <param name="loginUser"></param>
/// <returns></returns>
[HttpPost]
[Route("/audit/audit")]
public dynamic Audit([FromBody] BatchFlowModel batchFlow)
{
var loginUser = \_dentity.GetCurrentUser();
var result = \_audit.audit(batchFlow, loginUser);
return result;
}
/// <summary>
/// 根据流程记录与审核操作人审核流程
/// </summary>
/// <param name="recordModel"></param>
/// <param name="loginUser"></param>
/// <returns></returns>
[HttpPost]
[Route("/audit/auditRecord")]
public dynamic AuditRecord([FromBody]AuditRecordModel recordModel)
{
var loginUser = \_dentity.GetCurrentUser();
var result = \_audit.auditRecord(recordModel, loginUser);
return result;
}
/// <summary>
/// 重启审批流程
/// </summary>
/// <param name="recordId">流程记录标识</param>
/// <param name="limitType">重启流程操作用户限制类型：0不限制，1必须创建人</param>
/// <param name="loginUser"></param>
/// <returns></returns>
[HttpPost]
[Route("/audit/resetRecord")]
public dynamic ResetRecord(string recordId, string limitType)
{
var loginUser = \_dentity.GetCurrentUser();
var result = \_audit.resetRecord(new ResetFlowModel() { recordId = recordId, limitType = limitType }, loginUser);
return result;
}
/// <summary>
/// 获取流程消息记录
/// </summary>
/// <param name="recordId">流程记录标识</param>
/// <returns></returns>
[HttpGet]
[Route("/audit/auditlist")]
public dynamic Auditlist(string recordId)
{
var result = \_audit.auditlist(recordId);
return result;
}
/// <summary>
/// 获取消息详情列表
/// </summary>
/// <param name="todoId"></param>
/// <returns></returns>
[HttpGet]
[Route("/audit/flownode")]
public dynamic Flownode(string todoId)
{
var result = \_audit.flownode(todoId);
return result;
}
/// <summary>
/// 获取消息数量
/// </summary>
/// <param name="flowCode">流程编码（可传，可不传）</param>
/// <returns></returns>
[HttpGet]
[Route("/audit/getmsgcount")]
public dynamic Getmsgcount(string flowCode)
{
var loginUser = \_dentity.GetCurrentUser();
var result = \_audit.getmsgcount(flowCode, loginUser);
return result;
}
/// <summary>
/// 获取下一条待办消息
/// </summary>
/// <param name="orderbyType">根据时间排序：0倒序，1正序</param>
/// <param name="todoId">当前处理的待办标识Id</param>
/// <returns></returns>
[HttpGet]
[Route("/audit/GetNextTodo")]
public dynamic GetNextTodo(string orderbyType,string todoId)
{
var loginUser = \_dentity.GetCurrentUser();
var result = \_audit.GetNextTodo(orderbyType, todoId, loginUser);
return result;
}
}
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Wholesale.Model.DTO;
using Wholesale.Model.DTO.Basket;
using Wholesale.Services;
using Wholesale.Services.Storages;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class BasketController : BaseModule
{
private readonly IBasketService \_service;
private readonly IStorageService \_storageService;
public BasketController(IBasketService service, IStorageService storageService)
{
\_service = service;
\_storageService = storageService;
}
[Auth("入驻店铺销售统计")]
[Route("/basket/storage/sales")]
public dynamic GetBasketStorageSales([FromBody]PagedCondition page, [FromBody]StorageSalesSearchInput input)
{
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetBasketStorageSales(page, input);
}
[Auth("入驻店铺销售统计")]
[Route("/basket/storage/sales/detail")]
public dynamic GetBasketStorageSalesDetail([FromBody]PagedCondition page, [FromBody]StorageSalesDetailSearchInput input)
{
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetBasketStorageSalesDetail(page, input);
}
/// <summary>
/// 入驻店铺管理列表查询
/// </summary>
/// <param name="pager"></param>
/// <param name="eInput"></param>
/// <returns></returns>
[HttpPost("/basket/store/list")]
public dynamic GetList([FromBody] PagedCondition pager, [FromBody] EnterStoreInput input)
{
return \_storageService.GetStorageList(pager, input);
}
/// <summary>
/// 设置下级店铺展示
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/basket/store/setshow")]
public dynamic SetShow([FromBody] SetShowInput input)
{
return \_storageService.SetShow(input);
}
/// <summary>
/// 设置下级提成比例
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/basket/store/setratio")]
public dynamic SetRatio([FromBody] SetRatioInput input)
{
return \_storageService.SetRatio(input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 批量任务查询
/// </summary>
///
[Route("/batchTask")]
public class BatchTaskController : BaseController
{
private readonly IBatchTaskCommonService \_batchTaskCommonService;
public BatchTaskController(IBatchTaskCommonService batchTaskCommonService)
{
\_batchTaskCommonService = batchTaskCommonService;
}
[HttpPost("getBatchTaskList")]
public dynamic GeBatchTaskList([FromBody]BatckTaskInputDto input, [FromBody] PagedCondition pagedCondition)
{
return \_batchTaskCommonService.GeBatchTaskList(input, pagedCondition);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class BillStatisticsController : BaseController
{
private readonly IBillStatisticsService \_statisticsService;
public BillStatisticsController(IBillStatisticsService statisticsService)
{
this.\_statisticsService = statisticsService;
}
/// <summary>
/// 获取订单日账单统计数据
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/statistics/list")]
public dynamic GetBillStatisticsList([FromBody] PagedCondition page, [FromBody] BillStatisticsSearchInput search)
{
var result = \_statisticsService.GetBillStatisticsList(page, search);
return result;
}
/// <summary>
/// 统计一个时间区间内的订单数据
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/statistics/order")]
public dynamic StatisticsOrderInterval([FromBody]BillStatisticsSearchInput input)
{
var result = \_statisticsService.StatisticsOrderInterval(input);
return result;
}
/// <summary>
/// 统计一个时间区间内的第三方支付异常数据
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost]
[Route("/statistics/thirdpay")]
public dynamic StatisticsThirdPayInterval([FromBody]BillStatisticsSearchInput input)
{
var result = \_statisticsService.StatisticsThirdPayInterval(input);
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
namespace Wholesale.API.Controllers
{
public class BlackListController : BaseModule
{
CommissionBlackListService \_blackList;
public BlackListController(CommissionBlackListService blackList)
{
\_blackList = blackList;
}
/// <summary>
/// 添加黑名单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("添加提成黑名单")]
[HttpPost("/commission/blacklist/add")]
public Result Add([FromBody]CBlackListInput input)
{
return \_blackList.Add(input);
}
/// <summary>
/// 删除黑名单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("移除提成黑名单")]
[HttpPost("/commission/blacklist/del")]
public Result Del([FromBody]IdArrInput input)
{
return \_blackList.Del(input);
}
/// <summary>
/// 修改备注
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("添加提成黑名单")]
[HttpPost("/commission/blacklist/updateremark")]
public Result UpdateRemark([FromBody]CBlackListRemarkInput input)
{
return \_blackList.UpdateRemark(input);
}
/// <summary>
/// 黑名单列表
/// </summary>
/// <param name="page"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看提成黑名单")]
[HttpPost("/commission/blacklist/list")]
public Result<PagedList<CBlackListItemOutput>> GetList([FromBody]PagedCondition page, [FromBody]CBlackListSearchInput input)
{
return \_blackList.GetList(page, input);
}
/// <summary>
/// 导出
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("导出提成黑名单")]
[HttpPost("/commission/blacklist/export")]
public Result Export([FromBody]CBlackListSearchInput input)
{
return \_blackList.Export(input);
}
/// <summary>
/// 导入
/// </summary>
/// <param name="stream"></param>
/// <param name="fileName"></param>
/// <returns></returns>
[Auth("导入提成黑名单")]
[HttpPost("/commission/blacklist/import")]
public Result Import()
{
var httpFile = HttpContext.Request.Form.Files.FirstOrDefault();
return \_blackList.Import(httpFile.OpenReadStream(), httpFile.FileName);
}
/// <summary>
/// 操作记录
/// </summary>
/// <param name="page"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看提成黑名单历史记录")]
[HttpPost("/commission/blacklist/logs")]
public Result<PagedList<CBlackListLogOutput>> GetLogs([FromBody]PagedCondition page, [FromBody]CBlackListLogSearchInput input)
{
return \_blackList.GetLogs(page, input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO.Categories;
using Wholesale.Services.Products;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class CategoriesOffsetController : BaseController
{
private readonly ICategoriesOffsetService \_categoryService;
public CategoriesOffsetController(ICategoriesOffsetService categoryService)
{
\_categoryService = categoryService;
}
/// <summary>
/// 列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看分类扣点")]
[HttpGet]
[Route("/categoriesOffset/list")]
public dynamic GetCategoriesOffsetList(CategoriesOffsetListInput condition)
{
var result = \_categoryService.GetCategoriesOffsetList(condition);
return result;
}
/// <summary>
/// 设置
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("设置分类扣点")]
[HttpPost]
[Route("/categoriesOffset/add")]
public dynamic AddCategoriesOffset([FromBody]CategoriesOffsetAddInput condition)
{
var result = \_categoryService.AddCategoriesOffset(condition);
return result;
}
/// <summary>
/// 批量设置
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("设置分类扣点")]
[HttpPost]
[Route("/categoriesOffset/batchAdd")]
public dynamic BatchAddCategoriesOffset([FromBody] CategoriesOffsetAddInput condition)
{
var result = \_categoryService.BatchAddCategoriesOffset(condition);
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Wholesale.Services.Products;
using Zhidian.Common.AtuoMapper;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class CategoryController : BaseController
{
private readonly ICategoryService \_categoryService;
private readonly IUnitService \_unitService;
public CategoryController(ICategoryService categoryService, IUnitService unitService)
{
\_categoryService = categoryService;
\_unitService = unitService;
}
/// <summary>
/// 栏目列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet]
[Route("/category/list")]
public dynamic List(string storageId)
{
var result = \_categoryService.GetCategories(storageId);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<Category, CateogryListDto>();
}
}
}
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.DTO;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 城市服务费
/// </summary>
public class CityJoinFeeController : BaseController
{
private readonly ICityJoinFeeService \_custser;
public CityJoinFeeController(ICityJoinFeeService \_custser)
{
this.\_custser = \_custser;
}
/// <summary>
/// 获取服务费列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看服务费")]
[HttpGet]
[Route("/cityjoinfee/list")]
public dynamic GetCityJoinFeeList(PagedCondition pager, CityJoinFeeDto input)
{
var result = \_custser.GetCityJoinFeeList(pager, input);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
/// <summary>
/// 获取服务费
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("查看服务费")]
[HttpGet]
[Route("/cityjoinfee")]
public dynamic GetCityJoinFee(string id)
{
var result = \_custser.GetCityJoinFee(id);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
/// <summary>
/// 新增服务费
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("新增服务费")]
[HttpPost]
[Route("/cityjoinfee")]
public dynamic InsertCityJoinFee([FromBody]CityJoinFeeDto input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_custser.InsertCityJoinFee(input);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<CityJoinFee, CityJoinFeeDto>();
}
/// <summary>
/// 编辑服务费
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("编辑服务费")]
[HttpPut]
[Route("/cityjoinfee")]
public dynamic UpdateCityJoinFee([FromBody]CityJoinFeeDto input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_custser.UpdateCityJoinFee(input);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<CityJoinFee, CityJoinFeeDto>();
}
/// <summary>
/// 删除服务费
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("删除服务费")]
[HttpDelete]
[Route("/cityjoinfee/{id}")]
public dynamic DeleteCityJoinFee([FromBody]IdsInput module, string id)
{
if (id.IsNullOrEmpty())
{
id = module.ids;
}
var ids = id.Split(',').Select(s => s.ToStr()).ToList();
var result = \_custser.DeleteCityJoinFee(ids);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO.Copartners;
using Wholesale.Model.DTO;
using Wholesale.Model.DTO.Copartners;
using Wholesale.Services;
using Wholesale.Services.Copartners;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 合伙人令牌
/// </summary>
public class CopartnerTokenController : BaseModule
{
private readonly ICopartnerTokenService \_copartnerTokenService;
private readonly ICopartnerService \_copartnerService;
public CopartnerTokenController(
ICopartnerTokenService copartnerTokenService,
ICopartnerService copartnerService)
{
\_copartnerService = copartnerService;
\_copartnerTokenService = copartnerTokenService;
}
//[Auth("")]
[HttpPost]
[Route("/copartner/tokens")]
public dynamic List([FromBody]PagedCondition pager, [FromBody] CopartnerTokenSearchInput input)
{
return \_copartnerTokenService.List(input, pager);
}
//[Auth("")]
[HttpPost]
[Route("/copartner/tokens/export")]
public dynamic ListExportTask([FromBody]HeaderItemInput headers, [FromBody]CopartnerTokenSearchInput input)
{
return \_copartnerTokenService.ListExportTask(input, headers);
}
/// <summary>
/// 当前登录用户是不是合伙人
/// </summary>
/// <returns></returns>
[HttpPost("/copartner/iscopartner")]
public dynamic IsCopartner()
{
return \_copartnerTokenService.IsCopartner();
}
/// <summary>
/// 合伙人收入--供应商下拉框
/// </summary>
/// <returns></returns>
[HttpPost("/copartner/storagelist")]
public dynamic StorageList()
{
return \_copartnerTokenService.StorageList();
}
/// <summary>
/// 合伙人收入列表
/// </summary>
/// <param name="copartnerInComeSearchInput"></param>
/// <param name="pagedCondition"></param>
/// <returns></returns>
[HttpPost("/copartner/coparnterincomelist")]
public dynamic CoparnterInComeList([FromBody] CopartnerInComeSearchInput copartnerInComeSearchInput, [FromBody] PagedCondition pagedCondition)
{
var result = \_copartnerTokenService.CoparnterInComeList(copartnerInComeSearchInput, pagedCondition);
return result.Map<CopartnerInComeDto, CopartnerInComeOutput>();
}
/// <summary>
/// 合伙人收入列表 >（合伙人发展合伙人）
/// </summary>
/// <param name="input"></param>
/// <param name="pagedCondition"></param>
/// <returns></returns>
[HttpPost("/copartner/developCoparnterIncomeList")]
public dynamic CopartnerDevelopCopartnerIncomeList([FromBody]CopartnerDevelopCopartnerIncomeSearchInput input, [FromBody]PagedCondition pagedCondition)
{
var result = \_copartnerTokenService.CorparnterDevelopCorparnterInComeList(input, pagedCondition);
return result.Map<CopartnerGroupWealOutputDto, CopartnerGroupWealOutputOutput>();
}
/// <summary>
/// 合伙人入驻统计
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("合伙人入驻统计")]
[HttpPost("/copartner/statisticsEnteredCorpater")]
public dynamic StatisticsEnterdCopartner([FromBody]CopartnerEnteredStatisticsInput input, [FromBody] PagedCondition pagedCondition)
{
return \_copartnerTokenService.StatisticsEnterdCopartner(input, pagedCondition);
}
/// <summary>
/// 合伙人入驻统计导出
/// </summary>
/// <param name="headers"></param>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost]
[Route("/copartner/CopartnerEntered/export")]
public dynamic CopartnerEnteredExportTask([FromBody]HeaderItemInput headers, [FromBody]CopartnerEnteredStatisticsInput input)
{
return \_copartnerTokenService.CopartnerEnteredExportTask(input, headers);
}
}
}
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.DTO;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class CustomerLevelController : BaseController
{
private readonly ICustomerLevelService \_custser;
public CustomerLevelController(ICustomerLevelService \_custser)
{
this.\_custser = \_custser;
}
/// <summary>
/// 获取客户级别列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看客户级别")]
[HttpGet]
[Route("/customerlevel/list")]
public dynamic GetCustomerLevelList(PagedCondition pager, CustomerLevelDto input)
{
var result = \_custser.GetCustomerLevelList(pager, input);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<CustomerLevel, CustomerLevelDto>();
}
/// <summary>
/// 获取客户级别
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("查看客户级别")]
[HttpGet]
[Route("/customerlevel")]
public dynamic GetCustomerLevel(string id)
{
var result = \_custser.GetCustomerLevel(id);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
/// <summary>
/// 新增客户级别
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("新增客户级别")]
[HttpPost]
[Route("/customerlevel")]
public dynamic InsertCustomerLevel([FromBody]CustomerLevelDto input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_custser.InsertCustomerLevel(input);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<CustomerLevel, CustomerLevelDto>();
}
/// <summary>
/// 编辑客户级别
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("编辑客户级别")]
[HttpPut]
[Route("/customerlevel")]
public dynamic UpdateCustomerLevel([FromBody] CustomerLevelDto input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_custser.UpdateCustomerLevel(input);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<CustomerLevel, CustomerLevelDto>();
}
/// <summary>
/// 删除客户级别
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("删除客户级别")]
[HttpDelete]
[Route("/customerlevel/{id}")]
public dynamic DeleteCustomerLevel([FromBody]IdsInput module, string id)
{
if (id.IsNullOrEmpty())
{
id = module.ids;
}
var ids = id.Split(',').Select(s => s.ToStr()).ToList();
var result = \_custser.DeleteCustomerLevel(ids);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
/// <summary>
/// 获取客户级别列表下拉
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet]
[Route("/customerlevel/listdrop")]
public dynamic GetCustomerLevelListdrop(CustomerLevelDto input)
{
var result = \_custser.GetCustomerLevelListdrop(input);
return result;
}
/// <summary>
/// 获取客户级别-商品发布
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet]
[Route("/customerlevel/listpro")]
public dynamic GetCustomerLevelListPro(CustomerLevelDto input)
{
var result = \_custser.GetCustomerLevelListPro(input);
return result;
}
/// <summary>
/// 获取客户下拉-商品发布
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet]
[Route("/customerlevel/listprodrop")]
public dynamic GetCustomerPro()
{
var result = \_custser.GetCustomerPro();
return result;
}
}
}
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class CustomerSysController : BaseController
{
private readonly IUnionOrderService \_unionOrderService;
private readonly IDeliveryService \_deliveryService;
#region static ctor 枚举缓存
static Dictionary<string, Type> typeCache = new Dictionary<string, Type>(new IgnoreCaseComparer());
static Dictionary<string, List<EnumInfoItem>> enumInfoCache = new Dictionary<string, List<EnumInfoItem>>(new IgnoreCaseComparer());
static CustomerSysController()
{
//获取枚举类型
var types = AppDomain.CurrentDomain
.GetAssemblies()
.Where(a => a.ManifestModule.Name == "Inventory.Model.dll" || a.ManifestModule.Name == "Zhidian.Identity.dll")
.SelectMany(a => a.GetTypes())
.Where(t => t.IsEnum);
foreach (Type type in types)
{
if (!typeCache.Any(c => c.Key == type.Name))
{
typeCache.Add(type.Name, type);
}
}
}
#endregion
public CustomerSysController(IUnionOrderService unionOrderService, IDeliveryService deliveryService)
{
\_unionOrderService = unionOrderService;
\_deliveryService = deliveryService;
}
/// <summary>
/// 驳回差异申请
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/reject")]
public dynamic RejectDiff([FromBody] RejectDiffInput input)
{
var result = \_unionOrderService.RejectDiff(input);
return result;
}
/// <summary>
/// 获取枚举信息(用于填充下拉项)
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/enum/{name}")]
public dynamic GetEnumInfo(string name)
{
var result = new Result<List<EnumInfoItem>>();
var infos = GetEnumInfoForCache(name);
if (infos != null)
{
result.Successed(infos, "获取枚举信息成功");
return result;
}
Type type;
if (typeCache.TryGetValue(name, out type))
{
infos = type.GetEnumInfo();
enumInfoCache.Add(name, infos);
result.Successed(infos, "获取枚举信息成功");
return result;
}
return new Result()
{
Message = "不存在的枚举类型",
Status = ResultStatus.Error
};
}
/// <summary>
/// 从枚举信息缓存集合中获取枚举信息
/// </summary>
/// <param name="enumName"></param>
/// <returns></returns>
private List<EnumInfoItem> GetEnumInfoForCache(string enumName)
{
if (enumInfoCache.ContainsKey(enumName))
{
return enumInfoCache[enumName];
}
return null;
}
/// <summary>
/// 联营订单-查看差异详情
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/searchdiff")]
public dynamic SearchDiff([FromBody]IdInput input)
{
var result = \_unionOrderService.SearchDiff(input.Id);
return result;
}
/// <summary>
/// 录入差异
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/adddiff")]
public dynamic AddDiff([FromBody] AddDiffInput input)
{
var result = \_unionOrderService.AddDiff(input);
return result;
}
/// <summary>
/// 查看商家提交的差异
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/sellerdif")]
public dynamic SearchSellerDiffHandle([FromBody] IdInput input)
{
var result = \_unionOrderService.SearchSellerDiffHandle(input.Id);
return result;
}
/// <summary>
/// 联营订单-通知可发货
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/candeliver")]
public dynamic CanDeliver([FromBody] CanDeliverInput input)
{
var result = \_unionOrderService.CanDeliver(input);
return result;
}
/// <summary>
/// 客服系统-获取联营订单列表
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/list")]
public dynamic GetUnionOrders([FromBody]PagedCondition pager, [FromBody]GetUnionOrderListInput input)
{
var result = \_unionOrderService.GetUnionOrderList(pager, input);
return result.Map<UnionOrder, UnionOrderOutput>();
}
/// <summary>
/// 客服系统客服关闭订单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/close")]
public dynamic CustomerClose([FromBody]CloseOrderInput input)
{
var result = \_unionOrderService.CustomerClose(input);
return result;
}
/// <summary>
/// 修改订单送货单收货人的经纬度，并重推物流接口
/// </summary>
/// <param name="idInput"></param>
/// <returns></returns>
[HttpPost]
[Route("/unionorder/updatelnglat")]
public dynamic UpdateDeliveryLngLat([FromBody]IdInput idInput)
{
return \_deliveryService.UpdateDeliveryLngLat(idInput);
}
}
}
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Enums;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class DeliveryController : BaseController
{
private readonly IDeliveryService idelivery;
private readonly ILogisticsService ilogistics;
public DeliveryController(IDeliveryService \_idelivery, ILogisticsService \_ilogistics)
{
this.idelivery = \_idelivery;
this.ilogistics = \_ilogistics;
}
/// <summary>
/// 批发订单送货(点击送货直接标识订单为送货完成)
/// </summary>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost]
[Route("/delivery")]
public dynamic Delivery([FromBody]DeliveryInput input)
{
var result = idelivery.InsertDelivery(input);
return result;
}
/// <summary>
/// 批发订单送货(点击送货直接标识订单为送货完成)
/// </summary>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost]
[Route("/delivery/diffdelivery")]
public dynamic DiffDelivery([FromBody]DeliveryInput input)
{
var result = idelivery.InsertDelivery(input);
return result;
}
/// <summary>
/// 根据仓库编码查名下车辆车牌号和车型
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
[HttpGet]
[Route("/delivery/vehicle")]
public dynamic GetVehicleList(StorageNoInput storageNo)
{
var result = ilogistics.GetVehicleList(storageNo);
return result;
}
/// <summary>
/// 2.4.2. 查询订单物流轨迹
/// </summary>
/// <param name="search"></param>
/// <returns></returns>
[HttpGet]
[Route("/delivery/trace")]
public dynamic QueryTrace(DriverSearchInput search)
{
var result = ilogistics.QueryTrace(search);
return result;
}
/// <summary>
/// 获取物流公司信息
/// </summary>
/// <returns></returns>
[HttpGet]
[Route("/delivery/logisticscompany")]
public dynamic GetLogisticsCompanyList(int type)
{
Result<List<LogisticsCompany>> result = ilogistics.GetLogisticsCompanyList((ELogisticsCompanyType)type);
if (result.IsSuccessed)
{
return result.Map<LogisticsCompany, LogisticsCompanyOutput>();
}
return result;
}
/// <summary>
/// 发件方更新订单状态-备货完成
/// </summary>
/// <returns></returns>
[Auth("发货备货")]
[HttpPost]
[Route("/delivery/readied")]
public dynamic GoodsReadied([FromBody]EntityOrderNum orderNum)
{
Result result = idelivery.PushGoodsReadied(orderNum);// ilogistics.GoodsReadied(orderNum);
return result;
}
/// <summary>
/// 发货方式
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
[HttpGet]
[Route("/delivery/types")]
public dynamic DeliveryTypes()
{
return idelivery.DeliveryTypes();
}
/// <summary>
/// 联营商家代扣运费计算
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
[HttpPost]
[Route("/delivery/freightWithhold")]
public dynamic CalculateFreightWithhold([FromBody]FreightWithholdInput input)
{
return idelivery.CalculateFreightWithhold(input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model.DTO;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class DemandController : BaseModule
{
/// <summary>
/// 挂单服务
/// </summary>
private readonly IDemandPurchaseService \_serice;
//采购单收益报表
private readonly IDemandPurchaseOrderIncomeService \_incomeService;
private readonly IDemandOrderService \_demandOrderService;
public DemandController(IDemandPurchaseService service, IDemandPurchaseOrderIncomeService incomeService, IDemandOrderService demandOrderService)
{
\_serice = service;
\_incomeService = incomeService;
\_demandOrderService = demandOrderService;
}
/// <summary>
/// (挂单) 商家-放弃采购
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("商家放弃采购")]
[HttpPost]
[Route("/demand/wholesaler/purchase/cancel")]
public dynamic WholesalerPurchaseCancel([FromBody]IdInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_serice.WholesalerPurchaseCancel(input.Id);
}
/// <summary>
/// (挂单) 采购单-线下采购
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("线下采购")]
[HttpPost]
[Route("/demand/purchase/offline")]
public dynamic PurchaseOffline([FromBody]IdInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_serice.PurchaseOffline(input.Id);
}
/// <summary>
/// (挂单) 采购单-线下采购-推送价格
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("线下采购")]
[HttpPost]
[Route("/demand/purchase/offline/push/price")]
public dynamic PurchasePushPrice([FromBody]OfflinePushPriceInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_serice.PurchasePushPrice(input);
}
/// <summary>
/// (挂单) 采购单-线下采购-确认采购
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("线下采购")]
[HttpPost]
[Route("/demand/purchase/offline/push/confirm")]
public dynamic PurchasePushConfirm([FromBody]OfflinePushConfirmInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_serice.PurchasePushConfirm(input);
}
/// <summary>
/// 挂单收益报表
/// </summary>
/// <returns></returns>
[HttpPost("/demand/purchaseorder/Income")]
public dynamic DemandOrderIncome([FromBody]DemandPurchaseOrderIncomeDto input)
{
return \_incomeService.GetPurchaseOrderIncome(input);
}
/// <summary>
/// 获取需求单列表
/// </summary>
/// <param name="input"></param>
/// <param name="page"></param>
/// <returns></returns>
[Auth("需求列表")]
[HttpPost("/demand/order/list")]
public dynamic GetDemandOrderList([FromBody]DemandOrderListInput input, [FromBody]PagedCondition page)
{
return \_demandOrderService.GetDemandOrderList(input, page);
}
/// <summary>
/// 首页挂单待处理订单统计
/// </summary>
/// <returns></returns>
[HttpGet("/demand/order/statistics")]
public dynamic GetDemandStatistics()
{
return \_demandOrderService.GetDemandOrderStatistics();
}
/// <summary>
/// 首页根据国标码 取出购买的数量Top5
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/demand/gbcode/statistics")]
public dynamic GetDemandOrderGbCodeStatistics([FromBody] DemandOrderGbCodeStatisticsInputDto input)
{
return \_demandOrderService.GetDemandOrderGbCodeStatistics(input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model.DTO;
using Wholesale.Services;
using Wholesale.Services.Demand;
using Wholesale.Services.Order;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 挂单-采购单
/// </summary>
public class DemandPurchaseOrderController : BaseModule
{
private readonly IDemandPurchaseOrderService \_demandPurchaseOrderService;
private readonly IDemandWholesaleOrderService \_demandWholesaleOrderService;
private readonly IDemandPurchaseOrderConfigService \_demandPurchaseOrderConfigService;
private readonly IDemandPurchaseService \_demandPurchaseService;
public DemandPurchaseOrderController(
IDemandPurchaseOrderService demandPurchaseOrderService,
IDemandWholesaleOrderService demandWholesaleOrderService,
IDemandPurchaseOrderConfigService demandPurchaseOrderConfigService,
IDemandPurchaseService demandPurchaseService)
{
\_demandPurchaseOrderService = demandPurchaseOrderService;
\_demandWholesaleOrderService = demandWholesaleOrderService;
\_demandPurchaseOrderConfigService = demandPurchaseOrderConfigService;
\_demandPurchaseService = demandPurchaseService;
}
#region 测试使用，调用服务处理接口，上线前移除；暂不移除，暂定通过【查看采购单】权限控制
/// <summary>
/// 集单作业
/// </summary>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/test/QuartzPurchaseOrder")]
public dynamic QuartzPurchaseOrder()
{
return \_demandPurchaseOrderService.QuartzPurchaseOrder();
}
/// <summary>
/// 自动生成商家采购单 次日09:00时
/// </summary>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/test/AutoPushSellerPurchaseOrder")]
public dynamic AutoPushSellerPurchaseOrder()
{
\_demandPurchaseService.AutoPushSellerPurchaseOrder();
return Ok();
}
/// <summary>
/// 报价限时结束推送买家付款
/// </summary>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/test/AutoQuoteEndToBuyer")]
public dynamic AutoQuoteEndToBuyer()
{
\_demandPurchaseService.AutoQuoteEndToBuyer();
return Ok();
}
/// <summary>
/// 支付限时结束
/// </summary>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/test/AutoPayEndToConfirm")]
public dynamic AutoPayEndToConfirm()
{
\_demandPurchaseService.AutoPayEndToConfirm();
return Ok();
}
/// <summary>
/// 商家确认限时结束
/// 没确认的商家采购单才需要调用此接口
/// </summary>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/test/AutoSellerConfirmEndTime")]
public dynamic AutoSellerConfirmEndTime()
{
\_demandPurchaseService.AutoSellerConfirmEndTime();
return Ok();
}
#endregion
/// <summary>
/// 创建采购单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/demand/purchaseOrder/add")]
[Auth("创建采购单")]
public dynamic AddOrder([FromBody]DemandPurchaseOrderAddInput input)
{
return \_demandPurchaseOrderService.AddOrder(input);
}
/// <summary>
/// 取消采购单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/demand/purchaseOrder/cancel")]
[Auth("取消采购单")]
public dynamic CancelOrder([FromBody]IdInput input)
{
return \_demandPurchaseOrderService.CancelOrder(input);
}
/// <summary>
/// 采购单列表
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrders")]
public dynamic List([FromBody]PagedCondition pager, [FromBody]DemandPurchaseOrderInput input)
{
return \_demandPurchaseOrderService.List(input, pager);
}
/// <summary>
/// 采购单明细列表
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/details")]
public dynamic Details([FromBody]PagedCondition pager, [FromBody]DemandPurchaseOrderDetailInput input)
{
return \_demandPurchaseOrderService.Details(input, pager);
}
/// <summary>
/// 采购单详情导出
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/detailsExport")]
public dynamic DetailExport([FromBody]HeaderItemInput headers, [FromBody]DemandPurchaseOrderDetailInput input)
{
return \_demandPurchaseOrderService.DetailExport(input, headers);
}
/// <summary>
/// 采购单-换算单位列表
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("采购单换算单位")]
[HttpPost("/demand/purchaseOrder/units")]
public dynamic UnitList([FromBody]IdInput input)
{
return \_demandPurchaseOrderService.UnitList(input);
}
/// <summary>
/// 保存换算单位数据
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("采购单换算单位")]
[HttpPost("/demand/purchaseOrder/unitsSave")]
public dynamic SaveUnitList([FromBody]DemandPurchaseOrderUnitInput input)
{
return \_demandPurchaseOrderService.SaveUnitList(input);
}
/// <summary>
/// 采购单推送供应商信息
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/pushInfo")]
public dynamic PushInfo([FromBody]IdInput input)
{
return \_demandPurchaseOrderService.OnlinePushInfo(input);
}
/// <summary>
/// 确认采购供应商选择列表（待优化）
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/confirmStorages")]
public dynamic ConfirmStorages([FromBody]PagedCondition pager, [FromBody]DemandPurchaseOrderStorageInput input)
{
return \_demandPurchaseOrderService.OfflineConfirmStorages(input, pager);
}
/// <summary>
/// 采购单确认推送（选择SKU）
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/pushPriceInfo")]
public dynamic PushPriceInfo([FromBody]IdInput input)
{
return \_demandPurchaseOrderService.OfflinePushPriceInfo(input);
}
/// <summary>
/// 采购单供应商列表
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/storages")]
public dynamic ListStorage([FromBody]PagedCondition pager, [FromBody]DemandPurchaseOrderStorageInput input)
{
return \_demandPurchaseOrderService.ListStorage(input, pager);
}
/// <summary>
/// （线上）采购单-供应商报价列表
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/storageOffers")]
public dynamic OnlineOfferList([FromBody]IdInput input, [FromBody]PagedCondition pager)
{
return \_demandPurchaseOrderService.OnlineOfferList(input, pager);
}
/// <summary>
/// 确认采购
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("采购单确认采购")]
[HttpPost("/demand/purchaseOrder/confirm")]
public dynamic ConfirmPurchase([FromBody]ConfirmDemandPurchaseInput input)
{
return \_demandWholesaleOrderService.ConfirmDemandPurchase(input);
}
/// <summary>
/// （线上）采购单-推送（手动推送）
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看采购单")]
[HttpPost("/demand/purchaseOrder/push")]
public dynamic Push([FromBody]DemandPurchaseOrderPushInput input)
{
return \_demandPurchaseOrderService.Push(input);
}
/// <summary>
/// 获取 集采 配置
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpGet("/demand/purchaseOrder/config")]
public dynamic GetDemandPurchaseOrderDemandConfig()
{
return \_demandPurchaseOrderConfigService.GetDemandPurchaseOrderConfig();
}
/// <summary>
/// 设置 集采 配置
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/demand/purchaseOrder/updateConfig")]
public dynamic UpdateDemandPurchaseOrderDemandConfig([FromBody] DemandOrderConfigDto input)
{
return \_demandPurchaseOrderConfigService.UpdateDemandPurchaseOrderConfig(input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model.DTO;
using Wholesale.Services;
using Wholesale.Services.Demand;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 挂单-商家采购单
/// </summary>
public class DemandSellerPurchaseOrderController : BaseModule
{
private readonly IDemandPurchaseOrderService \_demandPurchaseOrderService;
private readonly IDemandPurchaseService \_demandPurchaseService;
public DemandSellerPurchaseOrderController(
IDemandPurchaseOrderService demandPurchaseOrderService,
IDemandPurchaseService demandPurchaseService)
{
\_demandPurchaseOrderService = demandPurchaseOrderService;
\_demandPurchaseService = demandPurchaseService;
}
/// <summary>
/// 商家采购单-列表
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看商家采购单")]
[HttpPost("/demand/seller/purchaseOrders")]
public dynamic List([FromBody]PagedCondition pager, [FromBody]DemandPurchaseOrderOfferInput input)
{
return \_demandPurchaseOrderService.SellerOfferList(pager, input);
}
/// <summary>
/// 商家采购单-明细列表
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看商家采购单")]
[HttpPost("/demand/seller/purchaseOrder/details")]
public dynamic Details([FromBody]PagedCondition pager, [FromBody]DemandSellerPurchaseOrderDetailInput input)
{
return \_demandPurchaseOrderService.SellerOfferDetails(input, pager);
}
/// <summary>
/// 商家采购单-供应商报价信息
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("商家采购单报价")]
[HttpPost("/demand/seller/purchaseOrder/offerInfo")]
public dynamic OfferInfo([FromBody]IdInput input)
{
return \_demandPurchaseOrderService.OfferInfo(input);
}
/// <summary>
/// 商家采购单-供应商报价
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("商家采购单报价")]
[HttpPost("/demand/seller/purchaseOrder/offer")]
public dynamic Offer([FromBody]DemandSellerPurchaseOrderOfferInput input)
{
return \_demandPurchaseOrderService.Offer(input);
}
/// <summary>
/// 商家采购单-弃单（放弃采购）
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("商家采购单弃单")]
[HttpPost("/demand/seller/purchaseOrder/giveup")]
[HttpPost("/demand/seller/purchaseOrder/cancel")]
public dynamic SellerGiveup([FromBody]IdInput input)
{
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_demandPurchaseService.WholesalerPurchaseCancel(input.Id);
}
}
}
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.DTO;
using Zhidian.Identity.Model;
using Zhidian.Identity.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class DepartmentController : BaseController
{
private readonly IDepartmentService \_dpService;
/\*
Post新增
Put编辑
Delete删除
Get获取信息
Get获取列表，后面路径带list
Get:commlist,获取可用部门信息，作为用户下拉选择使用
\*/
public DepartmentController(IDepartmentService dpService)
{
\_dpService = dpService;
}
/// <summary>
/// 新增部门信息
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("新增部门")]
[HttpPost]
[Route("/depart")]
public dynamic InsertDepartment([FromBody]DepartmentInput depart)
{
var validate = this.Validate(depart);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_dpService.InsertDepartment(depart);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<Department, DepartmentOutput>();
}
/// <summary>
/// 编辑部门信息
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("编辑部门")]
[HttpPut]
[Route("/depart")]
public dynamic UpdateDepartment([FromBody] DepartmentInput depart)
{
var validate = this.Validate(depart);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_dpService.UpdateDepartment(depart);
if (!result.IsSuccessed)
{
return result;
}
return result.Map<Department, DepartmentOutput>();
}
/// <summary>
/// 获取部门信息
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("查看部门")]
[HttpGet]
[Route("/depart")]
public dynamic GetDepartment(int id)
{
var result = \_dpService.GetDepartment(id);
if (!result.IsSuccessed)
{
return result;
}
return result;
}
/// <summary>
/// 获取部门信息列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看部门")]
[HttpGet]
[Route("/depart/list")]
public dynamic GetDepartmentList(PagedCondition pager, DepartmentInput dp)
{
Result<PagedList<DepartmentView>> result = \_dpService.GetDepartmentList(pager, dp);
return result.Map<DepartmentView, DepartmentViewOutput>();
}
/// <summary>
/// 获取部门信息列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet]
[Route("/depart/commlist")]
public dynamic GetCommDepartmentList()
{
Result<List<Department>> result = \_dpService.GetCommDepartmentList();
return result.Map<Department, DepartmentOutput>();
}
/// <summary>
/// 删除部门信息
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("删除部门")]
[HttpDelete]
[Route("/depart/{id}")]
public dynamic DeleteDepartment([FromBody] IdsInput module, string id)
{
if (id.IsNullOrEmpty())
{
id = module.ids;
}
var result = \_dpService.DeleteDepartment(id.ToIntList());
if (!result.IsSuccessed)
{
return result;
}
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Page;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class FileController : BaseController
{
private readonly IGeneratetFileService \_generatetFileService;
public FileController(IGeneratetFileService generatetFileService)
{
\_generatetFileService = generatetFileService;
}
/// <summary>
/// 重新下载
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
//[Auth("重新下载")]
[HttpPost]
[Route("/dwonload/retry")]
public dynamic Retry([FromBody]IdInput input)
{
var result = \_generatetFileService.RetryTask(input.Id);
return result;
}
/// <summary>
/// 获取文件下载列表
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
//[Auth("文件列表")]
[HttpPost]
[Route("/dwonload/list")]
public dynamic GetFiles([FromBody] PagedCondition pager, [FromBody] GetTaskListInput input)
{
var result = \_generatetFileService.GetTaskList(pager, input);
return result.Map<GeneratetFileTask, GeneratetFileTaskOutput>();
}
}
}
using System;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.DTO.Finance;
using Wholesale.Enums;
using Wholesale.Services;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Ormlite;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 财务管理模块
/// </summary>
public class FinanceController : BaseController
{
private readonly Lazy<IFinanceService> \_financeService;
private readonly Lazy<IGeneratetFileCommService> \_generatetFileCommService;
public FinanceController(Lazy<IFinanceService> financeService, Lazy<IGeneratetFileCommService> generatetFileCommService)
{
\_financeService = financeService;
\_generatetFileCommService = generatetFileCommService;
}
/// <summary>
/// 财务管理-待结算列表汇总
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务待结算")]
[HttpPost]
[Route("/finance/unliquidated/listsummary")]
public dynamic GetUnLiquidatedListSummary([FromBody] GetLiquidatedListInput input)
{
input.IsLiquidated = EBoolStatus.否;
var result = \_financeService.Value.GetLiquidatedListSummary(input);
return result;
}
/// <summary>
/// 财务管理-待结算列表
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务待结算")]
[HttpPost]
[Route("/finance/unliquidated/list")]
public dynamic GetUnLiquidatedList([FromBody] PagedCondition pager, [FromBody]GetLiquidatedListInput input)
{
input.IsLiquidated = EBoolStatus.否;
var result = \_financeService.Value.GetLiquidatedList(pager, input);
return result;
}
/// <summary>
/// 导出待结算列表
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务待结算")]
[HttpPost]
[Route("/finance/unliquidated/list/excel")]
public dynamic UnLiquidatedListToExcel([FromBody] HeaderItemInput headers, [FromBody]GetLiquidatedListInput input)
{
input.IsLiquidated = EBoolStatus.否;
var result = \_generatetFileCommService.Value.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.待结算列表导出,
TaskName = $"待结算列表导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"待结算列表导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 财务管理-待结算详情汇总
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务待结算")]
[HttpPost]
[Route("/finance/unliquidated/detailsummary")]
public dynamic GetUnLiquidatedDetailSummary([FromBody]GetLiquidatedDetailInput input)
{
input.IsLiquidated = EBoolStatus.否;
var result = \_financeService.Value.GetLiquidatedDetailSummary(input);
return result;
}
/// <summary>
/// 财务管理-待结算详情
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务待结算")]
[HttpPost]
[Route("/finance/unliquidated/detail")]
public dynamic GetUnLiquidatedDetails([FromBody] PagedCondition pager, [FromBody]GetLiquidatedDetailInput input)
{
input.IsLiquidated = EBoolStatus.否;
var result = \_financeService.Value.GetLiquidatedDetails(pager, input);
return result;
}
/// <summary>
/// 导出待结算明细
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务待结算")]
[HttpPost]
[Route("/finance/unliquidated/detail/excel")]
public dynamic UnLiquidatedDetailsToExcel([FromBody] HeaderItemInput headers, [FromBody]GetLiquidatedDetailInput input)
{
input.IsLiquidated = EBoolStatus.否;
var result = \_generatetFileCommService.Value.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.待结算明细导出,
TaskName = $"待结算明细导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"待结算明细导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 财务管理-已结算列表汇总
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务已结算")]
[HttpPost]
[Route("/finance/liquidated/listsummary")]
public dynamic GetLiquidatedListSummary([FromBody]GetLiquidatedListInput input)
{
input.IsLiquidated = EBoolStatus.是;
var result = \_financeService.Value.GetLiquidatedListSummary(input);
return result;
}
/// <summary>
/// 财务管理-已结算列表
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务已结算")]
[HttpPost]
[Route("/finance/liquidated/list")]
public dynamic GetLiquidatedList([FromBody] PagedCondition pager, [FromBody]GetLiquidatedListInput input)
{
input.IsLiquidated = EBoolStatus.是;
var result = \_financeService.Value.GetLiquidatedList(pager, input);
return result;
}
/// <summary>
/// 导出已结算列表
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务已结算")]
[HttpPost]
[Route("/finance/liquidated/list/excel")]
public dynamic LiquidatedListToExcel([FromBody] HeaderItemInput headers, [FromBody]GetLiquidatedListInput input)
{
input.IsLiquidated = EBoolStatus.是;
var result = \_generatetFileCommService.Value.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.已结算列表导出,
TaskName = $"已结算列表导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"已结算列表导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 财务管理-已结算详情汇总
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务已结算")]
[HttpPost]
[Route("/finance/liquidated/detailsummary")]
public dynamic GetLiquidatedDetailSummary([FromBody]GetLiquidatedDetailInput input)
{
input.IsLiquidated = EBoolStatus.是;
var result = \_financeService.Value.GetLiquidatedDetailSummary(input);
return result;
}
/// <summary>
/// 财务管理-已结算详情
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务已结算")]
[HttpPost]
[Route("/finance/liquidated/detail")]
public dynamic GetLiquidatedDetails([FromBody] PagedCondition pager, [FromBody]GetLiquidatedDetailInput input)
{
input.IsLiquidated = EBoolStatus.是;
var result = \_financeService.Value.GetLiquidatedDetails(pager, input);
return result;
}
/// <summary>
/// 导出已结算明细
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("财务已结算")]
[HttpPost]
[Route("/finance/liquidated/detail/excel")]
public dynamic LiquidatedDetailsToExcel([FromBody] HeaderItemInput headers, [FromBody]GetLiquidatedDetailInput input)
{
input.IsLiquidated = EBoolStatus.是;
var result = \_generatetFileCommService.Value.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.已结算明细导出,
TaskName = $"已结算明细导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"已结算明细导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 结算中心
/// </summary>
/// <param name="pager"></param>
/// <param name="input"></param>
/// <returns></returns>
//[Auth("结算中心")]
[HttpPost("/finance/settle/center")]
public dynamic FinanceSettleCenter([FromBody] PagedCondition pager, [FromBody]FinanceSettleCenterInput input)
{
var result = \_financeService.Value.FinanceSettleCenter(pager, input);
return result;
}
}
}
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO.Freights;
using Wholesale.Services.Freights;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class FreightController : BaseController
{
private readonly IIdentityService dentity;
private readonly IFreightService \_service;
public FreightController(IFreightService service, IIdentityService \_identity)
{
\_service = service;
dentity = \_identity;
}
/// <summary>
/// 运费设置
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("运费设置")]
[HttpPost]
[Route("/freight")]
public dynamic InsertFreight([FromBody]FreightInput condition)
{
var result = \_service.InsertFreight(condition);
return result;
}
/// <summary>
/// 刷新运费Redis
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("运费设置")]
[HttpPost]
[Route("/freight/updateredis")]
public dynamic UpdateFreight(List<string> storageIds)
{
var result = \_service.UpdateFreight(storageIds);
return result;
}
/// <summary>
/// 获取运费
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("运费设置")]
[HttpGet]
[Route("/freight")]
public dynamic GetFreight()
{
var result = \_service.GetFreight();
return result;
}
/// <summary>
/// 获取当前登陆用户所管理的仓库的运费以及订单数量信息（供首页显示）
/// </summary>
/// <returns></returns>
[Auth("运费设置")]
[HttpGet]
[Route("/freight/storageinfo")]
public dynamic GetUserStorageInfo()
{
var result = \_service.GetUserStorageInfo();
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.DTO;
using Zhidian.Identity.Model;
using Zhidian.Identity.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class GroupsController : BaseController
{
private readonly IGroupsService \_gs;
public GroupsController(IGroupsService gs)
{
\_gs = gs;
}
/// <summary>
/// 新增用户分组
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("新增分组")]
[HttpPost]
[Route("group")]
public dynamic InsertGroups([FromBody]GroupsInput group)
{
var validateResult = this.Validate(group);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result<Groups> result = \_gs.InsertGroups(group);
if (result.IsSuccessed)
{
//转换成Dto Result<Dto>
return result.Map<Groups, GroupsOutput>();
}
return result;
}
/// <summary>
/// 编辑用户分组
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑分组")]
[HttpPut]
[Route("group")]
public dynamic UpdateGroups([FromBody]GroupsInput group)
{
var validateResult = this.Validate(group);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result<Groups> result = \_gs.UpdateGroups(group);
if (!result.IsSuccessed)
{
return result;
}
//转换成Dto Result<Dto>
return result.Map<Groups, GroupsOutput>();
}
/// <summary>
/// 获取用户分组信息
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看用户组")]
[HttpGet]
[Route("group")]
public dynamic GetGroups(int id)
{
Result<Groups> result = \_gs.GetGroup(id);
if (result.IsSuccessed)
{
return result.Map<Groups, GroupsOutput>();
}
return result;
}
/// <summary>
/// 获取用户分组列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看用户组")]
[HttpGet]
[Route("/group/list")]
public dynamic GetGroupList(PagedCondition pager, Groups group)
{
var valid = this.Validate(pager);
if (!valid.IsValid)
{
return valid.ToResult();
}
Result<PagedList<GroupView>> result = \_gs.GetGroupList(pager, group);
if (result.IsSuccessed)
{
return result.Map<GroupView, GroupViewOutput>();
}
return result;
}
/// <summary>
/// 删除用户分组信息
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("删除分组")]
[HttpDelete]
[Route("/group/{id}")]
public dynamic DeleteGroups([FromBody]IdsInput module, string id)
{
if (id.IsNullOrEmpty())
{
id = module.ids;
}
Result result = \_gs.DeleteGroups(id.ToIntList());
return result;
}
/// <summary>
/// 获取可用用户组
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看用户组")]
[HttpGet]
[Route("/group/commlist")]
public dynamic GetCommGroupsList()
{
var result = \_gs.GetCommGroupsList();
return result.Map<Groups, GroupsOutput>();
}
/// <summary>
/// 配置用户组所属角色
/// </summary>
/// <param name="perms"></param>
/// <returns></returns>
[Auth("用户组授权")]
[HttpPut]
[Route("/group/grouprole")]
public dynamic UpdateGroupRole([FromBody]PermissionsInput perms)
{
Result<Groups> result = \_gs.UpdateGroupRole(perms);
if (result.IsSuccessed) { return result.Map<Groups, GroupsOutput>(); }
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class HomeController : BaseController
{
//注意：这里是构造函数
public HomeController()
{
}
[HttpGet]
[Route("/check")]
public dynamic Check()
{
return "测试成功";
}
}
}
using System;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Enums;
using Wholesale.Services;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
public class InventoryController : BaseController
{
private readonly IInventoryService \_inventoryService;
private readonly IGeneratetFileCommService \_generatetFileService;
public InventoryController(IInventoryService inventoryService, IGeneratetFileCommService generatetFileService)
{
\_inventoryService = inventoryService;
\_generatetFileService = generatetFileService;
}
/// <summary>
/// 查询补货单详情
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
//[Auth("补货")]
[HttpGet]
[Route("/inventory/waitdetail")]
public dynamic ReplenishmentByShareStorage(string id)
{
if (string.IsNullOrWhiteSpace(id))
{
return new Result(ResultStatus.ValidError, "共享仓id参数没有或者错误");
}
//var pager = this.Bind<PagedCondition>();
//var validate = this.Validate(pager);
//if (!validate.IsValid)
//{
// return validate.ToResult();
//}
//Result<PagedList<WaitDetailOutput>> result = \_inventoryService.ReplenishmentByShareStorage(pkId, pager);
Result<SharedOrderDtoForUpdate> result = \_inventoryService.ReplenishmentByShareStorageNew(id);
return result;
//var map = result.Map<Result<List<WaitDetailOutput>>>();
//return map;
}
//[Auth("暂不补货")]
[HttpGet]
[Route("/inventory/notadd")]
public dynamic NotReplenishment(string id)
{
if (string.IsNullOrWhiteSpace(id))
{
return new Result(ResultStatus.ValidError, "id不能为空");
}
var result = \_inventoryService.NotReplenishment(id);
return result;
}
//[Auth("查看库存清单")]
[HttpPost]
[Route("/inventory/list")]
public dynamic GetShareInventoryListAndStatistics([FromBody]PagedCondition pager, [FromBody]InventoryInput search)
{
var validate = this.Validate(pager);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_inventoryService.GetShareInventoryListAndStatistics(pager, search);
return result;
}
//[Auth("查看库存清单")]
[HttpGet]
[Route("/inventory")]
public dynamic GetShareInventoryListAndStatisticsByStorageId(PagedCondition pager, InventoryDetailInput input,string id)
{
if (string.IsNullOrWhiteSpace(id))
{
return new Result(ResultStatus.ValidError, "id不能为空");
}
var result = \_inventoryService.GetShareInventoryListAndStatisticsByStorageId(id, pager, input);
//if (result.IsSuccessed)
//{
// var temp = result.Map<InventoryDetailDto, InventoryDetailOutput>();
// return temp;
//}
return result;
}
//[Auth("查看待补货清单")]
[HttpPost]
[Route("/inventory/waitorders")]
public dynamic GetWaitOrderListAndStatistics([FromBody]PagedCondition pager, [FromBody]InventoryWaitInput search)
{
var validate = this.Validate(pager);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = \_inventoryService.GetWaitOrderListAndStatistics(pager, search);
//if (result.IsSuccessed)
//{
// var temp = result.Map<InventoryWaitDto, InventoryWaitOuput>();
// return temp;
//}
return result;
}
/// <summary>
/// 导出共享仓库存清单Excel
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
//[Auth("导出共享仓库存清单")]
[HttpPost]
[Route("/inventory/ShareInventoryListToExcel")]
public dynamic GetShareInventoryListToExcel([FromBody]HeaderItemInput headers, [FromBody]InventoryInput input)
{
Result result = new Result();
result = \_generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.共享仓库存清单导出,
TaskName = $"共享仓库存清单导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"共享仓库存清单导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 导出共享仓库存清单Excel
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
//[Auth("导出共享仓库存详情")]
[HttpPost]
[Route("/inventory/ShareInventoryStatisticsListToExcel")]
public dynamic GetShareInventoryStatisticsListToExcel([FromBody]HeaderItemInput headers, [FromBody]InventoryDetailInput input)
{
Result result = new Result();
result = \_generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.共享仓库存清单明细导出,
TaskName = $"共享仓库存清单明细导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"共享仓库存清单明细导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 导出共享仓补货清单Excel
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
//[Auth("导出共享仓补货清单")]
[HttpPost]
[Route("/inventory/ShareInventoryWaitorderListToExcel")]
public dynamic GetShareInventoryWaitorderListToExcel([FromBody]HeaderItemInput headers, [FromBody]InventoryWaitInput input)
{
Result result = new Result();
result = \_generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.共享仓库存补货清单导出,
TaskName = $"共享仓库存补货清单导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"共享仓库存补货清单导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
namespace Wholesale.API.Controllers
{
/// <summary>
/// java组对接接口查看订单详情信息
/// </summary>
[Route("/javaorder")]
public class JavaOrderController : BaseModule
{
private readonly IWholesaleOrderService \_orderService;
public JavaOrderController(IWholesaleOrderService orderService) //: base("javaorder")
{
\_orderService = orderService;
}
/// <summary>
/// 根据订单编号获取订单详情
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[HttpPost("orderView")]
public dynamic GetOrderViewByNo([FromBody]IdInput input)
{
//var input = this.Bind<IdInput>();
return \_orderService.GetOrderViewByNo(input.Id);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services.Products;
namespace Wholesale.API.Controllers
{
public class JavaProductController : BaseModule
{
private readonly IProductService \_productService;
public JavaProductController(IProductService productService) //: base("")
{
this.\_productService = productService;
}
/// <summary>
/// 获取sku箱规换算数量
/// </summary>
/// <param name="skuCode"></param>
/// <returns></returns>
[HttpPost("/javaproduct/unitquantity")]
public dynamic GetProductUnitQuantity([FromBody]IdInput input)
{
//IdInput input = this.Bind<IdInput>();
return \_productService.GetProductUnitQuantity(input.Id);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO.Log;
using Wholesale.Services.Log;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Log.Models;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("/log")]
public class LogController : BaseModule
{
private readonly ILogService \_logService;
public LogController(ILogService logService) //: base("/log")
{
\_logService = logService;
}
[Auth("查看系统日志")]
[HttpGet("SearchLog")]
public dynamic SearchLog([FromQuery]PagedCondition page, [FromQuery]LogSearchInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<LogSearchInput>();
var vResult = this.Validate(page);
if (!vResult.IsValid)
{
return vResult.ToResult();
}
page.PageIndex++;
var result = \_logService.SearchLog(page, input);
return result.Map<LogItem, LogListDto>();
}
[Auth("查看操作日志")]
[HttpGet("SearchOperation")]
public dynamic SearchOperation([FromQuery]PagedCondition page, [FromQuery]OperationSearchInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<OperationSearchInput>();
var vResult = this.Validate(page);
if (!vResult.IsValid)
{
return vResult.ToResult();
}
page.PageIndex++;
var result = \_logService.SearchOperation(page, input);
return result.Map<OperationItem, OperationListDto>();
}
}
}
using Microsoft.AspNetCore.Mvc;
using System;
using System.IO;
using System.Threading.Tasks;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Identity.DTO;
using Zhidian.Identity.Model;
using Zhidian.Identity.Services;
using Zhidian.Web;
using Zhidian.Web.Infrastructure;
namespace Wholesale.API.Controllers
{
public class LoginController : Controller //: NancyModule
{
private readonly IIdentityService identity;
private readonly IIdentityWholesaleService wholeIdentity;
public LoginController(IIdentityService \_identity, IIdentityWholesaleService \_wholeIdentity)
{
identity = \_identity;
wholeIdentity = \_wholeIdentity;
}
/// <summary>
/// 登录时，检测用户是否需要输入验证码
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpGet("login/verifycode/need")]
public dynamic NeedVerifyCode(string mno)
{
//var userName = Request.Query.mno;
var result = wholeIdentity.NeedVerifyCode(mno);
return result;
}
/// <summary>
/// 登录
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("login")]
public dynamic Login([FromBody]LoginInput login)
{
//var login = this.BindAndValidate<LoginInput>();
var valite = Validator.Validate(login);// this.Validate(login);
if (!valite.IsValid)
{
return valite.ToResult();
}
string key = login.CodeKey;
if (string.IsNullOrWhiteSpace(key))
{
//Request.Cookies.TryGetValue("valikey", out key);
Zhidian.Common.Web.HttpContext.Current.Request.Cookies.TryGetValue("valikey", out key);
login.CodeKey = key ?? Guid.NewGuid().ToShortString();
}
var result = wholeIdentity.LoginWholesale(login);
if (result.IsSuccessed)
{
return result.Map<PUser, PUserOutput>();
}
return result;
}
/// <summary>
/// 退出用户登录
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("login/loginout")]
public dynamic LoginOut()
{
var result = identity.LoginOut();
return result;
}
/// <summary>
/// 获取登录用户
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet("login")]
public dynamic GetLogin()
{
var result = identity.GetCurrentUser();
return result;
}
/// <summary>
/// 找回密码
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost("findpassword")]
public dynamic FindPassword([FromBody]Zhidian.Java.DTO.FindPwdInput input)
{
//var input = this.Bind<Zhidian.Java.DTO.FindPwdInput>();
return wholeIdentity.FindPassword(input);
}
/// <summary>
/// 获取短信验证码（找回密码）
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost("findpassword/smscode")]
public dynamic GetFindPasswordSmsCode([FromBody]Zhidian.Java.DTO.GetSMSInput input)
{
//var input = this.Bind<Zhidian.Java.DTO.GetSMSInput>();
return wholeIdentity.GetFindPasswordSMSCode(input);
}
/// <summary>
/// 获取图形验证码（找回密码）
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpGet("findpassword/imgcode")]
public async Task GetFindPasswordImgCode([FromQuery]Zhidian.Java.DTO.GetImageCode input)
{
var img = wholeIdentity.GetFindPasswordImgCode(input);
var ms = new MemoryStream();
img.Save(ms, System.DrawingCore.Imaging.ImageFormat.Jpeg);
var buffer = ms.ToArray();
Response.ContentType = "image/jpeg";
await Response.Body.WriteAsync(buffer, 0, buffer.Length);
}
/// <summary>
/// 校验短信验证码（找回密码）
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpPost("findpassword/smscode/check")]
public dynamic CheckFindPasswordSmsCode([FromBody]Zhidian.Java.DTO.CheckSMSInput input)
{
//var input = this.Bind<Zhidian.Java.DTO.CheckSMSInput>();
return wholeIdentity.CheckFindPasswordSMSCode(input);
}
}
}
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Enums;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("material")]
public class MaterialController : BaseModule
{
private readonly IMaterialOrderService \_materialOrderService;
private readonly IMaterialSkuService \_materialSkuService;
private readonly IMaterialOrderPushService \_pushService;
public MaterialController(
IMaterialOrderService materialOrderService,
IMaterialSkuService materialSkuService,
IMaterialOrderPushService pushService) //: base("/material")
{
\_materialOrderService = materialOrderService;
\_materialSkuService = materialSkuService;
\_pushService = pushService;
}
/// <summary>
/// 获取原料订单
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看原料订单")]
[HttpPost("order/detail")]
public dynamic GetMaterialOrderById([FromBody]IdInput idInput)
{
//IdInput idInput = this.Bind<IdInput>();
return \_materialOrderService.GetMaterialOrderById(idInput.Id);
}
/// <summary>
/// 获取原料单列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
///
[Auth("查看原料订单")]
[HttpPost("order/list")]
public dynamic GetMaterialOrders([FromBody]PagedCondition page, [FromBody]MaterialOrderSearchInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<MaterialOrderSearchInput>();
var validatePage = this.Validate(page);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_materialOrderService.GetMaterialOrders(page, input);
}
/// <summary>
/// 原料订单发货
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("原料订单发货")]
[HttpPost("order/send")]
public dynamic UpdateMaterialOrderSend([FromBody]MaterialOrderSendInput input)
{
//var input = this.Bind<MaterialOrderSendInput>();
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_materialOrderService.UpdateMaterialOrderSend(input);
}
/// <summary>
/// 获取收货单
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("receipt/view")]
public dynamic GetReceiptOrder([FromBody]PagedCondition page, [FromBody]MaterialReceiptOrderInput receipt)
{
//var receipt = this.Bind<MaterialReceiptOrderInput>();
//var page = this.Bind<PagedCondition>();
return \_materialOrderService.GetReceiptOrder(receipt, page);
}
/// <summary>
/// 订原料单导出任务
/// </summary>
[HttpPost("export")]
public dynamic OrderExportTask([FromBody]MaterialOrderSearchInput input, [FromBody]HeaderItemInput header)
{
//var input = this.Bind<MaterialOrderSearchInput>();
//var header = this.Bind<HeaderItemInput>();
return \_materialOrderService.OrderExportTask(input, header);
}
/// <summary>
/// 上游订单-列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看上游订单")]
[HttpPost("up/list")]
public dynamic UpOrderList([FromBody]PagedCondition pager, [FromBody]MaterialUpOrderInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialUpOrderInput>();
return \_materialOrderService.UpOrderList(input, pager);
}
/// <summary>
/// 上游订单-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("上游订单导出")]
[HttpPost("up/export")]
public dynamic UpOrderExportTask([FromBody]MaterialUpOrderInput input, [FromBody]HeaderItemInput headers)
{
//var input = this.BindAndValidate<MaterialUpOrderInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
return \_materialOrderService.UpOrderExportTask(input, headers);
}
/// <summary>
/// 上游订单详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看上游订单")]
[HttpPost("up/detail")]
[HttpPost("up/print")]
public dynamic UpOrderDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.UpOrderDetail(idInput.Id);
}
/// <summary>
/// 上游订单手动推单详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("上游订单手动推单")]
[HttpPost("up/push/detail")]
public dynamic UpOrderOrderPushDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.UpOrderOrderPushDetail(idInput.Id);
}
/// <summary>
/// 上游订单手动推单
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("上游订单手动推单")]
[HttpPost("up/push")]
public dynamic UpOrderOrderPush([FromBody]MaterialUpOrderPushOrderInput input)
{
//var input = this.Bind<MaterialUpOrderPushOrderInput>();
var data = \_materialOrderService.UpOrderOrderPush(input);
if (!data.IsSuccessed)
return data;
return \_pushService.ControllPushOrder(data.Data);
}
/// <summary>
/// 上游订单-发货列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看上游订单发货明细")]
[HttpPost("up/delivery/list")]
public dynamic UpOrderDeliveryList([FromBody]PagedCondition pager, [FromBody]MaterialUpOrderDeliveryInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialUpOrderDeliveryInput>();
return \_materialOrderService.UpOrderDeliveryList(input, pager);
}
/// <summary>
/// 上游订单-发货明细
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看上游订单发货明细")]
[HttpPost("up/delivery/detail")]
public dynamic UpOrderDeliveryDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.UpOrderDeliveryDetail(idInput.Id);
}
/// <summary>
/// 上游订单-完成订单
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("上游订单完成订单")]
[HttpPost("up/complete")]
public dynamic UpOrderComplete([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.UpOrderComplete(idInput.Id);
}
/// <summary>
/// 下游订单列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看下游订单")]
[HttpPost("down/list")]
public dynamic DownOrderList([FromBody]PagedCondition pager, [FromBody]MaterialDownOrderInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialDownOrderInput>();
return \_materialOrderService.DownOrderList(input, pager);
}
/// <summary>
/// 下游订单-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("下游订单导出")]
[HttpPost("down/export")]
public dynamic DownOrderExportTask([FromBody]MaterialDownOrderInput input, [FromBody]HeaderItemInput headers)
{
//var input = this.BindAndValidate<MaterialDownOrderInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
return \_materialOrderService.DownOrderExportTask(input, headers);
}
/// <summary>
/// 下游订单详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看下游订单")]
[HttpPost("down/detail")]
[HttpPost("down/print")]
public dynamic DownOrderDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.DownOrderDetail(idInput.Id);
}
/// <summary>
/// 下游订单详情（原料商家-煤场）
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("order/print/detail")]
public dynamic DownOrderDetailByMaterial([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.DownOrderDetail(idInput.Id, false);
}
/// <summary>
/// 下游订单-收货列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看下游订单收货明细")]
[HttpPost("down/receiving/list")]
public dynamic DownOrderReceivingList([FromBody]PagedCondition pager, [FromBody]MaterialDownOrderReceivingInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialDownOrderReceivingInput>();
if (input == null) input = new MaterialDownOrderReceivingInput();
input.DownStatus = EMaterialReceiptUpStatus.已收货;
input.UpStatus = EMaterialReceiptUpStatus.已收货;
return \_materialOrderService.DownOrderReceivingList(input, pager);
}
/// <summary>
/// 下游订单-收货列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看下游订单收货明细")]
[HttpPost("down/receiving/receipt")]
public dynamic DownOrderReceivingListByWaitDelivery([FromBody]PagedCondition pager, [FromBody]MaterialDownOrderReceivingInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialDownOrderReceivingInput>();
if (input == null) input = new MaterialDownOrderReceivingInput();
input.DownStatus = EMaterialReceiptUpStatus.待收货;
input.UpStatus = EMaterialReceiptUpStatus.已收货;
return \_materialOrderService.DownOrderReceivingList(input, pager);
}
/// <summary>
/// 下游订单-收货明细
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看下游订单收货明细")]
[HttpPost("down/receiving/detail")]
public dynamic DownOrderReceivingDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.DownOrderReceivingDetail(idInput.Id);
}
/// <summary>
/// 下游订单-收货明细（原料商家-煤场）
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("receipt/list/detail")]
[HttpPost("receipt/print/detail")]
public dynamic DownOrderReceivingDetailByMaterial([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.DownOrderReceivingDetail(idInput.Id, false);
}
/// <summary>
/// 下游订单-确认收货
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("下游订单确认收货")]
[HttpPost("down/receiving/confirm")]
public dynamic DownOrderReceivingConfirm([FromBody]IdArrInput input)
{
//var input = this.Bind<IdArrInput>();
return \_materialOrderService.DownOrderReceivingConfirm(input.Ids);
}
/// <summary>
/// 下游订单-完成订单
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("下游订单完成订单")]
[HttpPost("down/complete")]
public dynamic DownOrderComplete([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.DownOrderComplete(idInput.Id);
}
/// <summary>
/// 商品列表-关联商品-列表
/// </summary>
/// <returns></returns>
[Auth("关联商家商品")]
[HttpPost("sku/relation/list")]
public dynamic SkuRelationList([FromBody]MaterialSkuRelationSearchInput input)
{
//var input = this.BindAndValidate<MaterialSkuRelationSearchInput>();
return \_materialSkuService.SkuRelationList(input);
}
/// <summary>
/// 商品列表-关联商品-商品搜索列表
/// </summary>
/// <returns></returns>
[Auth("关联商家商品")]
[HttpPost("sku/relation/sku/list")]
public dynamic SkuRelationSkuList([FromBody]PagedCondition pager, [FromBody]MaterialSkuRelationSkuSearchInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialSkuRelationSkuSearchInput>();
return \_materialSkuService.SkuRelationList(input, pager);
}
/// <summary>
/// 商品列表-关联商品-历史记录
/// </summary>
/// <returns></returns>
[Auth("关联商家商品")]
[HttpPost("sku/relation/history/list")]
public dynamic SkuRelationHistoryList([FromBody]MaterialSkuRelationSearchInput input)
{
//var input = this.BindAndValidate<MaterialSkuRelationSearchInput>();
return \_materialSkuService.SkuRelationHistoryList(input);
}
/// <summary>
/// 商品列表-验证产品对应的订单是否流程中
/// </summary>
/// <returns></returns>
[Auth("关联商家商品")]
[HttpPost("sku/validate")]
public dynamic ValidateSkuIsWF([FromBody]MaterialSkuRelationSearchInput input)
{
//var input = this.BindAndValidate<MaterialSkuRelationSearchInput>();
return \_materialSkuService.ValidateSkuIsWF(input.SkuCode);
}
/// <summary>
/// 商品列表-关联商品-绑定
/// </summary>
/// <returns></returns>
[Auth("商家商品绑定")]
[HttpPost("sku/relation/bind")]
public dynamic SkuRelationBind([FromBody]MaterialSkuRelationInput input)
{
//var input = this.BindAndValidate<MaterialSkuRelationInput>();
return \_materialSkuService.SkuRelationBind(input);
}
/// <summary>
/// 商家绑定对应的商家下拉框列表
/// </summary>
/// <returns></returns>
[HttpPost("up/storages")]
[HttpPost("down/storages")]
[HttpPost("sku/relation/sku/storages")]
public dynamic StoragesRelation()
{
return \_materialSkuService.StoragesRelation();
}
/// <summary>
/// 手动创建下游订单
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("order/down/add")]
public dynamic DownOrderAdd([FromBody]MaterialOrderPushInput input)
{
//var input = this.Bind<MaterialOrderPushInput>();
var validate = this.Validate(input);
if (!validate.IsValid)
{
return validate.ToResult();
}
return \_pushService.ControllPushOrder(input);
}
/// <summary>
/// 订单增加打印次数
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("order/print/")]
[HttpPost("up/print/add")]
[HttpPost("down/print/add")]
public dynamic OrderAddPrint([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.OrderAddPrint(idInput.Id);
}
/// <summary>
/// 配送单增加打印次数
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("receipt/print")]
public dynamic ReceiptAddPrint([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.ReceiptAddPrint(idInput.Id);
}
/// <summary>
/// 下游订单（煤厂）批量发货
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("批量发货")]
[HttpPost("down/import")]
public dynamic ReceiptOrderImport()
{
var httpFile = HttpContext.Request.Form.Files.FirstOrDefault();
//var httpFile = Request.Files.FirstOrDefault();
return \_materialOrderService.ReceiptOrderImport(httpFile.OpenReadStream(), httpFile.FileName);
}
[Auth("原料订单发货")]
[HttpPost("order/control/send/list")]
public dynamic OrderSendList([FromBody]NoInput input)
{
//var input = this.Bind<NoInput>();
return \_materialOrderService.MaterialReceipSendOrderList(input);
}
[Auth("原料订单发货")]
[HttpPost("order/control/send")]
public dynamic OrderSend([FromBody]MaterialReceiptSendOrderInput input)
{
//var input = this.Bind<MaterialReceiptSendOrderInput>();
return \_materialOrderService.MaterialReceipSendOrder(input);
}
/// <summary>
/// 下游订单-发货明细列表（煤场）
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("receipt/list")]
public dynamic DownOrderReceivingListByMaterial([FromBody]PagedCondition pager, [FromBody]MaterialDownOrderReceivingInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialDownOrderReceivingInput>();
return \_materialOrderService.DownOrderReceivingList(input, pager, false);
}
[Auth("原料订单发货")]
[HttpPost("driver/no")]
public dynamic MaterialDriverNo([FromBody]PagedCondition pager)
{
//var pager = this.BindAndValidate<PagedCondition>();
return \_materialOrderService.MaterialDriverNo(pager);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.DTO.MaterialSettle;
using Wholesale.Services;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.Services;
using Zhidian.Ormlite.OrmLite;
namespace Wholesale.API.Controllers
{
[Route("/materialsettleorder")]
public class MaterialSettleController : BaseModule
{
private readonly IIdentityService dentity;
private readonly IMaterialSettleService \_service;
private readonly IMaterialOrderService \_materialOrderService;
public MaterialSettleController(
IMaterialSettleService service,
IIdentityService \_identity,
IMaterialOrderService materialOrderService) // : base("/materialsettleorder")
{
\_service = service;
dentity = \_identity;
\_materialOrderService = materialOrderService;
}
/// <summary>
/// 结算单列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("list")]
public dynamic GetSettleList([FromBody]PagedCondition pager, [FromBody]MaterialSettleOrderSearch search)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var search = this.BindAndValidate<MaterialSettleOrderSearch>();
return \_service.GetSettleList(pager, search);
}
[HttpPost("totalAmount")]
public dynamic GetSettleOrderTotalAmont([FromBody]MaterialSettleOrderSearch search)
{
//var search = this.BindAndValidate<MaterialSettleOrderSearch>();
return \_service.GetSettleOrderTotalAmont(search);
}
/// <summary>
/// 结算单详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpPost("detail")]
public dynamic GetSettleDetail([FromBody]MaterialSettleInput input)
{
return \_service.GetSettleDetail(input);
}
/// <summary>
/// 标记已付款
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("标记结算单已付款")]
[HttpPost("orderpay")]
public dynamic OrderPay([FromBody]MaterialSettleInput input)
{
return \_service.OrderPay(input);
}
/// <summary>
/// 结算单导出任务
/// </summary>
[HttpPost("export")]
public dynamic SettleExportTask([FromBody]MaterialSettleOrderSearch input, [FromBody] HeaderItemInput header)
{
//var input = this.Bind<MaterialSettleOrderSearch>();
//var header = this.Bind<HeaderItemInput>();
return \_service.SettleExportTask(input, header);
}
/// <summary>
/// 上游结算单-列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看上游结算单")]
[HttpPost("up/list")]
public dynamic UpOrderList([FromBody]PagedCondition pager, [FromBody] MaterialSettleUpOrderSearchInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialSettleUpOrderSearchInput>();
return \_service.UpOrderList(input, pager);
}
/// <summary>
/// 上游结算单-导出
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("上游结算单导出")]
[HttpPost("up/export")]
public dynamic UpOrderExportTask([FromBody]MaterialSettleUpOrderSearchInput input, [FromBody]HeaderItemInput headers)
{
//var input = this.BindAndValidate<MaterialSettleUpOrderSearchInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
return \_service.UpOrderExportTask(input, headers);
}
/// <summary>
/// 上游订单-发货明细
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("上游结算单明细")]
[HttpPost("up/detail")]
public dynamic UpOrderDeliveryDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.UpSettleReceiptDetail(idInput.Id);
}
/// <summary>
/// 下游结算单-列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看下游结算单")]
[HttpPost("down/list")]
public dynamic DownOrderList([FromBody]PagedCondition pager, [FromBody] MaterialSettleDownOrderSearchInput input)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var input = this.BindAndValidate<MaterialSettleDownOrderSearchInput>();
return \_service.DownOrderList(input, pager);
}
/// <summary>
/// 下游结算单-导出
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("下游结算单导出")]
[HttpPost("down/export")]
public dynamic DownOrderExportTask([FromBody]HeaderItemInput headers, [FromBody] MaterialSettleUpOrderSearchInput input)
{
//var input = this.BindAndValidate<MaterialSettleUpOrderSearchInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
return \_service.DownOrderExportTask(input, headers);
}
/// <summary>
/// 下游订单-收货明细
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("下游结算单明细")]
[HttpPost("down/detail")]
public dynamic DownOrderReceivingDetail([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_materialOrderService.DownOrderReceivingDetail(idInput.Id);
}
/// <summary>
/// 标记付款
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("上游结算单标记付款")]
[HttpPost("up/payment")]
public dynamic UpOrderIsTagPayment([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_service.OrderIsTagPayment(idInput.Id);
}
/// <summary>
/// 标记付款
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("下游结算单标记付款")]
[HttpPost("down/payment")]
public dynamic DownOrderIsTagPayment([FromBody]IdInput idInput)
{
//var idInput = this.Bind<IdInput>();
return \_service.OrderIsTagPayment(idInput.Id);
}
}
}
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.DTO;
using Zhidian.Identity.Enum;
using Zhidian.Identity.Model;
using Zhidian.Identity.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("/user")]
public class MemberController : BaseModule
{
private readonly IWholesaleUserService us;
private readonly IUserService iuser;
private readonly IIdentityService identity;
/\*
Post新增
Put编辑
Delete删除
Get获取信息
Get获取列表，后面路径带list
\*/
public MemberController(IWholesaleUserService \_us, IIdentityService \_identity, IUserService \_iuser) // : base("/user")
{
us = \_us;
iuser = \_iuser;
identity = \_identity;
}
[Auth("用户新增")]
[HttpPost()]
public dynamic InsertUser([FromBody]UserInput u)
{
//var u = this.Bind<UserInput>();
var validateResult = this.Validate(u);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result<User> result = us.InsertUser(u);
if (!result.IsSuccessed)
{
return result;
}
//转换成Dto Result<Dto>
return result.Map<User, WholesaleUserOutput>();
}
[Auth("用户编辑")]
[HttpPut()]
public dynamic UpdateUser([FromBody]UserInput u)
{
//var u = this.Bind<UserInput>();
var validateResult = this.Validate(u);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result<User> result = us.UpdateUser(u);
//转换成Dto Result<Dto>
return result.Map<User, WholesaleUserOutput>();
}
/// <summary>
/// 管理员重置密码
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("重置密码")]
[HttpPut("restpwd")]
public dynamic RestPassword([FromBody]RestPasswordInput u)
{
//var u = this.Bind<RestPasswordInput>();
var validateResult = this.Validate(u);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result result = us.RestPassword(u);
return result;
}
/// <summary>
/// 当前用户修改密码
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("修改密码")]
[HttpPut("updatepwd")]
public dynamic UpdatePassword([FromBody]RestPasswordInput u)
{
//var u = this.Bind<RestPasswordInput>();
var validateResult = this.Validate(u);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result result = us.UpdatePassword(u);
//转换成Dto Result<Dto>
return result;
}
[Auth("删除用户")]
[HttpDelete("{id}")]
public dynamic DeleteUser([FromBody]IdsInput module, string id)
{
//string id = parameters.id;
if (id.IsNullOrEmpty())
{
//var module = this.BindAndValidate<IdsInput>();
id = module.ids;
}
Result result = us.DeleteUser(id.ToIntList());
return result;
}
/// <summary>
/// 查看用户详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看用户")]
[HttpGet()]
public dynamic GetWholesaleUser(int id)
{
//int id = Request.Query.id;
var result = us.GetWholesaleUser(id);
return result;
}
[Auth("查看用户")]
[HttpGet("list")]
[HttpGet("commlist")]
public dynamic GetUserList([FromQuery]PagedCondition pager, [FromQuery]User u)
{
pager.PageIndex++;
//var pager = this.BindAndValidate<PagedCondition>();
//var u = this.BindAndValidate<User>();
Result<PagedList<UserView>> result = iuser.GetUserList(pager, u);
return result.Map<UserView, UserViewOutput>();
}
/// <summary>
/// 获取公共选择的用户信息
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看用户")]
public dynamic GetCommUserList(PagedCondition pager, User u)
{
//var pager = this.BindAndValidate<PagedCondition>();
//var u = this.BindAndValidate<User>();
Result<PagedList<User>> result = iuser.GetCommUserList(pager, u);
return result.Map<User, UserOutput>();
}
[Auth("用户授权")]
[HttpGet("userrolelist")]
public dynamic GetUserRoleList(int userId)
{
//var userId = Request.Query.userId;
Result<List<UserRoleView>> result = iuser.GetUserRoleList(userId);
return result;
}
[Auth("用户授权")]
[HttpGet("usermodulelist")]
public dynamic GetUserModuleList(int userId)
{
//var userId = Request.Query.userId;
Result<List<UserModuleView>> result = iuser.GetUserModuleList(userId);
return result;
}
[Auth("用户授权")]
[HttpGet("usergrouplist")]
public dynamic GetUserGroupsList(int userId)
{
//var userId = Request.Query.userId;
Result<List<UserGroupView>> result = iuser.GetUserGroupsList(userId);
return result;
}
[Auth("用户授权")]
[HttpPut("userrole")]
public dynamic UpdateUserRole([FromBody]PermissionsInput perms)
{
//var perms = this.Bind<PermissionsInput>();
Result<User> result = iuser.UpdateUserRole(perms);
if (result.IsSuccessed)
{
return result.Map<User, UserOutput>();
}
return result;
}
[Auth("用户授权")]
[HttpPut("usermodule")]
public dynamic UpdateUserModule([FromBody]PermissionsInput perms)
{
//var perms = this.Bind<PermissionsInput>();
Result<User> result = iuser.UpdateUserModule(perms);
if (result.IsSuccessed)
{
return result.Map<User, UserOutput>();
}
return result;
}
[Auth("用户授权")]
[HttpPut("usergroup")]
public dynamic UpdateUserGroup([FromBody]PermissionsInput perms)
{
//var perms = this.Bind<PermissionsInput>();
Result<User> result = iuser.UpdateUserGroup(perms);
if (result.IsSuccessed)
{
return result.Map<User, UserOutput>();
}
return result;
}
/// <summary>
/// 获取登录用户
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet("logininfo")]
public dynamic GetLogin()
{
//var result = identity.GetLoinInfo();
//update 20180705 使用重写方法
var result = us.GetLoinInfo();
return result;
}
/// <summary>
/// 获取用户类型
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet("usertype")]
public dynamic GetUserType()
{
var result = iuser.GetUserType();
return result.Map<EUserType, JsonEnum>();
}
}
}
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Identity.DTO;
using Zhidian.Identity.Model;
using Zhidian.Identity.Services;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("/module")]
public class MenuController : BaseModule
{
private readonly IModuleService ms;
private readonly IIdentityService dentity;
/\*
Post新增
Put编辑
Delete删除
Get获取信息
Get获取列表，后面路径带list
\*/
public MenuController(IModuleService \_ms, IIdentityService \_identity) //: base("/module")
{
ms = \_ms;
dentity = \_identity;
}
[Auth("新增模块")]
[HttpPost()]
public dynamic InsertModule([FromBody]ModuleInput module)
{
//var module = this.BindAndValidate<ModuleInput>();
var validateResult = this.Validate(module);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result<Module> result = ms.InsertModule(module);
if (result.IsSuccessed)
{
return result.Map<Module, ModuleOutput>();
}
//转换成Dto Result<Dto>
return result;
}
[Auth("编辑模块")]
[HttpPut()]
public dynamic UpdateModule([FromBody]ModuleInput module)
{
//var module = this.BindAndValidate<ModuleInput>();
var validateResult = this.Validate(module);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
Result<Module> result = ms.UpdateModule(module);
if (result.IsSuccessed) { return result.Map<Module, ModuleOutput>(); }
return result;
}
[Auth("删除模块")]
[HttpDelete()]
[HttpDelete("{id}")]
public dynamic DeleteModule([FromBody]Zhidian.Identity.DTO.IdsInput module, string id)
{
//string id = parameters.id;
if (id.IsNullOrEmpty())
{
//var module = this.BindAndValidate<Zhidian.Identity.DTO.IdsInput>();
id = module.ids;
}
Result result = ms.DeleteModule(id.ToIntList());
return result;
}
[Auth("查看模块")]
[HttpGet()]
public dynamic GetModule(int id)
{
//var id = Request.Query.id;
Result<Module> result = ms.GetModule(id);
if (result.IsSuccessed)
{
return result.Map<Module, ModuleOutput>(); ;
}
return result;
}
[Auth("查看模块")]
[HttpGet("list")]
public dynamic GetModuleList([FromQuery]PagedCondition pager)
{
//var pager = this.BindAndValidate<PagedCondition>();
var validateResult = this.Validate(pager);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
pager.PageIndex++;
Result<PagedList<ModuleView>> result = ms.GetModuleList(pager);
return result.Map<ModuleView, ModuleViewOutput>();
}
/// <summary>
/// 获取菜单（用于父菜单选择使用）
/// </summary>
/// <returns></returns>
[HttpGet("menulist")]
public dynamic GetMenuList()
{
Result<List<Module>> result = ms.GetMenuList();
return result.Map<Module, ModuleOutput>();
}
/// <summary>
/// 获取当前用户下的菜单
/// </summary>
/// <returns></returns>
[HttpGet("usermenu")]
public dynamic GetUserModuleList()
{
Result<IList<UserModuleDTO>> result = dentity.GetUserModuleList();
return result;
}
/// <summary>
/// 获取可用模块
/// </summary>
/// <returns></returns>
[HttpGet("commlist")]
public dynamic GetCommModuleList()
{
Result<List<ModuleView>> result = ms.GetCommModuleList();
return result.Map<ModuleView, ModuleViewOutput>();
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Model;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Page;
namespace Wholesale.API.Controllers
{
[Route("message")]
public class MessageController : BaseModule
{
private readonly IMessageService \_messageService;
public MessageController(IMessageService messageService) //: base("message")
{
this.\_messageService = messageService;
}
/// <summary>
/// 获取用户当天的消息（最近一天的消息）
/// </summary>
/// <returns></returns>
[HttpPost("daylist")]
public dynamic GetDayMessageList([FromBody]MessageSearchInput search)
{
//var search = this.Bind<MessageSearchInput>();
var result = \_messageService.GetDayMessageList(search);
if (result.IsSuccessed)
{
return result.Map<MessageText, MessageTextOutput>();
}
return result;
}
/// <summary>
/// 获取当前用户消息列表
/// </summary>
/// <param name="page"></param>
/// <returns></returns>
[HttpPost("list")]
public dynamic GetMessageList([FromBody]PagedCondition page, [FromBody]MessageSearchInput search)
{
//var page = this.Bind<PagedCondition>();
//var search = this.Bind<MessageSearchInput>();
var result = \_messageService.GetMessageList(page, search);
if (result.IsSuccessed)
{
return result.Map<MessageText, MessageTextOutput>();
}
return result;
}
/// <summary>
/// 将消息设置为已读状态
/// </summary>
/// <param name="Ids">消息</param>
/// <returns></returns>
[HttpPost("updateread")]
public dynamic UpdateIsRead([FromBody]IdInput input)
{
//IdInput input = this.Bind<IdInput>();
var result = \_messageService.UpdateIsRead(input);
return result;
}
/// <summary>
/// 获取当前用户未读消息总数量
/// </summary>
/// <returns></returns>
[HttpGet("messagecount")]
public dynamic GetMessageCount()
{
var result = \_messageService.GetMessageCount();
return result;
}
/// <summary>
/// 获取用户各类型未读消息数量
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[HttpGet("unreadmessage")]
public dynamic GetUserUnreadMessage()
{
var result = \_messageService.GetUserUnreadMessage();
return result;
}
/// <summary>
/// 获取用户新消息
/// </summary>
/// <param name="num"></param>
/// <returns></returns>
[HttpGet("newmessage")]
public dynamic NewMessage(int num)
{
//int num = Request.Query.num;
var result = \_messageService.NewMessage(num);
if (result.IsSuccessed)
{
return result.Map<MessageText, MessageTextOutput>();
}
return result;
}
}
}
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.DTO.Order;
using Wholesale.DTO.Refunds;
using Wholesale.Enums;
using Wholesale.Model;
using Wholesale.Services;
using Wholesale.Services.Order;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("order")]
public class OrderController : BaseModule
{
private readonly IWholesaleOrderService order;
private readonly IRefundsService refundsService;
private readonly ISystemOrderService isystem;
private readonly IGeneratetFileCommService generatetFileService;
private readonly Lazy<IStewardWealService> \_stewardWealService;
private readonly IOrderImportService \_orderImportService;
private readonly IOrderInsteadCustomersService \_orderInsteadGuestService;
//无货源商家结算单统计
private readonly INotSupplyGoodsStorageService \_notSupplyGoodsStorageService;
public OrderController(
IWholesaleOrderService \_order,
IRefundsService \_refundsService,
ISystemOrderService \_isystem,
IGeneratetFileCommService \_generatetFileService,
Lazy<IStewardWealService> stewardWealService,
IOrderImportService orderImportService,
IOrderInsteadCustomersService orderInsteadGuestService,
INotSupplyGoodsStorageService notSupplyGoodsStorageService)
{
this.order = \_order;
refundsService = \_refundsService;
this.isystem = \_isystem;
generatetFileService = \_generatetFileService;
\_stewardWealService = stewardWealService;
\_orderImportService = orderImportService;
\_orderInsteadGuestService = orderInsteadGuestService;
\_notSupplyGoodsStorageService = notSupplyGoodsStorageService;
}
/// <summary>
/// 获取订单导入批次列表
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("批次管理")]
[HttpPost("batch/list")]
public dynamic GetBatchList([FromBody]GetBatchListInput input, [FromBody]PagedCondition pager)
{
return \_orderImportService.GetBatchListForWeb(input, pager);
}
/// <summary>
/// 重发提成
/// </summary>
/// <returns></returns>
[UnAuth]
[HttpPost("resendweal")]
public dynamic ResendWeal([FromBody]ResendWealInput input)
{
//var input = this.Bind<ResendWealInput>();
var result = order.ResendWeal(input);
return result;
}
/// <summary>
/// 取消维护人预期提成
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[UnAuth]
[HttpGet("stewardweal/cancel")]
public dynamic CancelStewardWeal(string no)
{
var orderNo = no;// this.Request.Query.no;
var result = \_stewardWealService.Value.CancelExpected(orderNo);
return result;
}
/// <summary>
/// 联营订单，待收货，录入差异（发货后）
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[Auth("查看批发订单")]
[HttpGet("deliverafter/searchdifference")]
public dynamic SearchDifference([FromQuery]IdInput input)
{
//var input = this.Bind<IdInput>();
var result = order.SearchDifference(input.Id, EOrdeFlowSource.发货后);
return result;
}
/// <summary>
/// 联营订单，待收货，录入差异（发货前）
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[Auth("查看批发订单")]
[HttpGet("deliverbefore/searchdifference")]
public dynamic SearchDifference2([FromQuery]IdInput input)
{
//var input = this.Bind<IdInput>();
var result = order.SearchDifference(input.Id, EOrdeFlowSource.发货前);
return result;
}
/// <summary>
/// 普通订单退款
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[Auth("查看批发订单")]
[HttpPost("regularrefund")]
public dynamic RegularRefund([FromBody]RegularRefundInput input)
{
//var input = this.Bind<RegularRefundInput>();
var result = refundsService.RegularRefund(input);
return result;
}
/// <summary>
/// 获取仓库批发订单列表
/// </summary>
/// <param name="pager"></param>
/// <param name="search">查询条件</param>
/// <returns></returns>
[Auth("查看批发订单")]
[HttpPost("list")]
public dynamic GetWholesaleStorageOrderList([FromBody]PagedCondition pager, [FromBody]WholesaleSearch search)
{
//var pager = this.Bind<PagedCondition>();
//var search = this.Bind<WholesaleSearch>();
var validate = this.Validate(pager);
if (!validate.IsValid)
{
return validate.ToResult();
}
var result = order.GetWholesaleStorageOrderList(pager, search);
if (result.IsSuccessed)
{
result.Data.List.Each(item => PackOrderView(item));
return result.Map<WholesaleStorageOrderView, WholesaleStorageOrderViewOutput>();
}
return result;
}
/// <summary>
/// 获取仓库订单详情
/// </summary>
/// <param name="Id"></param>
/// <returns></returns>
[Auth("查看批发订单")]
[HttpGet()]
public dynamic GetWholesaleStorageOrderById(string id)
{
//string id = this.Request.Query.id;
var result = order.GetWholesaleStorageOrderById(id);
if (result.IsSuccessed)
{
PackOrderView(result.Data);
return result.Map<WholesaleStorageOrderView, WholesaleStorageOrderViewOutput>();
}
return result;
}
private void PackOrderView(WholesaleStorageOrderView orderView)
{
if (orderView.OrderStatus == EOrderStatus.交易关闭)
{
orderView.RefundsDeliverAfter = 0M;
orderView.RefundsDeliverBefore = 0M;
}
}
/// <summary>
/// 获取打印批发订单
/// </summary>
/// <param name="Id"></param>
/// <returns></returns>
[Auth("打印批发订单")]
[HttpGet("print")]
public dynamic GetWholesaleStorageOrderPrintById(string id)
{
//string id = this.Request.Query.id;
var result = order.WholesaleStorageOrderPrint(id);
if (result.IsSuccessed)
{
return result.Map<WholesaleStorageOrderView, WholesaleStorageOrderViewOutput>();
}
return result;
}
/// <summary>
/// 批发订单送货(点击送货直接标识订单为送货完成)
/// </summary>
/// <param name="Id">仓库订单标识</param>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost("delivery")]
public dynamic WholseOrderDelivery([FromBody]IdInput input)
{
//IdInput input = this.Bind<IdInput>();
var result = order.WholseOrderDelivery(input.Id);
return result;
}
/// <summary>
/// 退单操作
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("退单")]
[HttpPost("refund")]
public dynamic WebRefundOrder([FromBody]WebCancelOrderInput input)
{
//WebCancelOrderInput input = this.Bind<WebCancelOrderInput>();
var result = order.WebRefundOrder(input);
return result;
}
/// <summary>
/// 订单退款第三方支付的费用
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("第三方退款")]
[HttpPost("refundpay")]
public dynamic WebRefundThirdPay([FromBody]WebCancelOrderInput input)
{
//WebCancelOrderInput input = this.Bind<WebCancelOrderInput>();
var result = order.WebRefundThirdPay(input);
return result;
}
/// <summary>
/// 推送库存至综合仓
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost("zhcstock")]
public dynamic PushZhcStock([FromBody]IdInput input)
{
//IdInput input = this.Bind<IdInput>();
var result = order.PushZhcStock(input);
return result;
}
/// <summary>
/// 批量推送库存至综合仓
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost("batchstock")]
public dynamic BatchPushZhcStock([FromBody]List<string> input)
{
//var input = this.Bind<List<string>>();
var result = order.BatchPushZhcStock(input);
return result;
}
/// <summary>
/// 批量推送商品至商品库
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost("batchproduct")]
public dynamic BatchPushZhcStockProduct([FromBody]List<string> input)
{
//var input = this.Bind<List<string>>();
var result = order.BatchPushZhcStockProduct(input);
return result;
}
/// <summary>
/// 关闭订单，用户发货前强制关闭订单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("关闭订单")]
[HttpPost("close")]
public dynamic Close([FromBody]CancelInput input)
{
//CancelInput input = this.Bind<CancelInput>();
var result = order.Close(input);
return result;
}
[Auth("联营订单")]
[HttpPost("unionorder")]
public dynamic GetUnionOrderList([FromBody]PagedCondition page, [FromBody]WholesaleSearch input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<WholesaleSearch>();
var result = order.GetUnionOrderList(page, input);
return result;
}
[Auth("直营订单")]
[HttpPost("directorder")]
public dynamic GetDirectOrderList([FromBody]PagedCondition pager, [FromBody]WholesaleSearch input)
{
input.DirectStatus = Zhidian.Ormlite.EBoolStatus.是;
var result = order.GetWholesaleStorageOrderList(pager, input);
if (result.IsSuccessed)
{
result.Data.List.Each(item => PackOrderView(item));
return result.Map<WholesaleStorageOrderView, WholesaleStorageOrderViewOutput>();
}
return result;
}
/// <summary>
/// 上传送货单图片/销售合同图片
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("直营订单")]
[HttpPost("images")]
public dynamic UploadOrderImages([FromBody]UploadOrderImagesInput input)
{
return order.UploadOrderImagesForWebApi(input);
}
/// <summary>
/// 查看送货单图片/销售合同图片
/// </summary>
/// <param name="paramer"></param>
/// <returns></returns>
[Auth("直营订单")]
[HttpPost("seeimages")]
public dynamic SeeOrderImages([FromBody]SeeOrderImagesInput input)
{
return order.SeeOrderImages(input);
}
/// <summary>
/// 订单收益明细
/// </summary>
/// <returns></returns>
[Auth("批发发货")]
[HttpGet("IncomeDetail")]
public dynamic IncomeDetail([FromQuery]PagedCondition page, [FromQuery]IncomeDetailSearchInput input)
{
page.PageIndex++;
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<IncomeDetailSearchInput>();
var result = order.IncomeDetailWithoutWeal(page, input);
return result;
}
/// <summary>
/// 获取预期明细列表
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
[HttpGet("expectedetail")]
public dynamic GetExpectedDetail([FromQuery]PagedCondition page, [FromQuery]IncomeDetailSearchInput input)
{
page.PageIndex++;
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<IncomeDetailSearchInput>();
var result = order.GetExpectedDetail(page, input);
return result;
}
/// <summary>
/// 批发商预期、可提提成列表
/// </summary>
[HttpGet("commissiondetail")]
public dynamic GetCommissionDetail([FromQuery]PagedCondition page, [FromQuery]IncomeDetailSearchInput input)
{
page.PageIndex++;
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<IncomeDetailSearchInput>();
if (input.ClassifyType == EClassifyType.预期)
return order.GetExpectedCommissionDetail(page, input);
else
return order.GetCanCommissionDetail(page, input);
}
/// <summary>
/// 联营订单按商品维度打印
/// </summary>
/// <param name="Id"></param>
/// <returns></returns>
[Auth("打印批发订单")]
[HttpPost("printsku")]
public dynamic PrintOrderSkuDetail([FromBody]IdListInput input)
{
//var input = this.Bind<IdListInput>();
var result = order.PrintOrderSkuDetail(input);
return result;
}
/// <summary>
/// 联营订单按订单维度打印
/// </summary>
/// <param name="Id"></param>
/// <returns></returns>
[Auth("打印批发订单")]
[HttpPost("printorder")]
public dynamic PrintOrderDetail([FromBody]IdListInput input)
{
//var input = this.Bind<IdListInput>();
var result = order.PrintOrderDetail(input);
if (result.IsSuccessed)
{
return result.Map<WholesaleStorageOrderView, WholesaleStorageOrderViewOutput>();
}
return result;
}
/// <summary>
/// 直营订单-送货单打印
/// </summary>
/// <param name="Id"></param>
/// <returns></returns>
[Auth("打印批发订单")]
[HttpPost("directorder/images/print")]
public dynamic ImagesPrint([FromBody]ImagesPrintInput input)
{
var result = order.ImagesPrint(input);
return result;
}
/// <summary>
/// 推送结算系统
/// </summary>
/// <param name="param"></param>
/// <returns></returns>
[HttpPost("pushsettle")]
public dynamic PushSettle([FromBody]IdInput input)
{
//var input = this.Bind<IdInput>();
var result = isystem.PushSettle(input.Id);
return result;
}
/// <summary>
/// 导出订单
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("导出订单")]
[HttpPost("excel")]
public dynamic ExportToExcel([FromBody]WholesaleSearch input)
{
Result result = new Result();
if (!input.StartTime.HasValue || input.StartTime.Value == DateTime.MaxValue || input.StartTime.Value == DateTime.MinValue)
{
result.Errored("请选择开始日期");
return result;
}
if (!input.EndTime.HasValue || input.EndTime.Value == DateTime.MaxValue || input.EndTime.Value == DateTime.MinValue)
{
result.Errored("请选择结束日期");
return result;
}
if (input.StartTime.Value.CompareTo(input.EndTime.Value) > 0)
{
result.Errored("开始日期不能大于结束日期");
return result;
}
if (input.StartTime < input.EndTime.Value.AddDays(-60).Date)
{
result.Errored("最多只能导出60天的订单");
return result;
}
result = generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = "[]",
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.订单导出Excel,
TaskName = $"订单导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"订单导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 雅士导出订单
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("导出订单")]
[HttpPost("yashi/excel")]
public dynamic ExportToExcelYashi([FromBody]WholesaleSearch input)
{
Result result = new Result();
if (!input.StartTime.HasValue || input.StartTime.Value == DateTime.MaxValue || input.StartTime.Value == DateTime.MinValue)
{
result.Errored("请选择开始日期");
return result;
}
if (!input.EndTime.HasValue || input.EndTime.Value == DateTime.MaxValue || input.EndTime.Value == DateTime.MinValue)
{
result.Errored("请选择结束日期");
return result;
}
if (input.StartTime.Value.CompareTo(input.EndTime.Value) > 0)
{
result.Errored("开始日期不能大于结束日期");
return result;
}
if (input.StartTime < input.EndTime.Value.AddDays(-60).Date)
{
result.Errored("最多只能导出60天的订单");
return result;
}
result = generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = "[]",
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.雅士订单导出,
TaskName = $"订单导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"订单导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 导出联营订单
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("导出订单")]
[HttpPost("unionorder/excel")]
public dynamic UnionOrderToExcel()
{
//var input = this.Bind<WholesaleSearch>();
var input = new WholesaleSearch()
{
StartTime = DateTime.Now.AddDays(-60).Date
};
input.OrderSource = (int)EOrderSource.联营订单;
Result result = new Result();
if (!input.StartTime.HasValue)
{
result.Errored("请选择开始日期");
return result;
}
if (input.StartTime < DateTime.Now.AddDays(-60).Date)
{
result.Errored("只能导出最近60天的订单");
return result;
}
result = generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = "[]",
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.订单导出Excel,
TaskName = $"联营订单导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"联营订单导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 导出直营订单
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("导出订单")]
[HttpPost("directorder/excel")]
public dynamic DirectOrderToExcel()
{
//var input = this.Bind<WholesaleSearch>();
var input = new WholesaleSearch()
{
StartTime = DateTime.Now.AddDays(-60).Date
};
input.DirectStatus = Zhidian.Ormlite.EBoolStatus.是;
Result result = new Result();
if (!input.StartTime.HasValue)
{
result.Errored("请选择开始日期");
return result;
}
if (input.StartTime < DateTime.Now.AddDays(-60).Date)
{
result.Errored("只能导出最近60天的订单");
return result;
}
result = generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = "[]",
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.订单导出Excel,
TaskName = $"直营订单导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"直营订单导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
/// <summary>
/// 预订活动修复订单商品数量
/// </summary>
/// <returns></returns>
[HttpPost("bookingup")]
public dynamic BookingBuyQuantity()
{
return order.BookingBuyQuantity();
}
/// <summary>
/// 打印次数自增长
/// </summary>
/// <param name="paramr"></param>
/// <returns></returns>
[Auth("打印批发订单")]
[HttpPost("printcount")]
public dynamic SelfGrowingPrintCount([FromBody]IdInput input)
{
//IdInput input = this.Bind<IdInput>();
return order.SelfGrowingPrintCount(input);
}
[Auth("标记订单已付款")]
[HttpPost("confirmPay")]
public dynamic ConfirmPay([FromBody]IdInput input)
{
//IdInput input = this.Bind<IdInput>();
return order.ConfirmPay(input);
}
[Auth("打印批发订单")]
[HttpPost("/order/directorder/print/count")]
public dynamic InvoicePrintCount([FromBody]ImagesPrintInput input)
{
return order.InvoicePrintCount(input);
}
[Auth("查看批发订单")]
[HttpPost("/order/refund/detail")]
public dynamic RefundDetail([FromBody]NoInput input)
{
return refundsService.RefundDetail(input);
}
[Auth("查看批发订单")]
[HttpPost("/order/refund/import/list")]
public dynamic RefundList([FromBody]IdInput input, [FromQuery]PagedCondition pager)
{
return refundsService.RefundList(pager, input);
}
/// <summary>
/// 获取订单物流轨迹
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("批发发货")]
[HttpPost("/order/logisticsTrajectory")]
public dynamic GetLogisticsTrajectory([FromBody] IdInput idInput)
{
var result = order.GetOrderLogisticsById(idInput);
if (result.IsSuccessed)
{
return result.Map<OrderDeliveryDto, OrderDeliveryOutput>();
}
return result;
}
/// <summary>
/// 代客下单-预下单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/shop/order/preview")]
public dynamic PreViewOrder([FromBody]OrderInsteadCustomersPreViewInput input)
{
return \_orderInsteadGuestService.PreViewOrder(input);
}
/// <summary>
/// 代客下单-确认订单
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/shop/order/buyConfirm")]
public dynamic BuyConfirm([FromBody]OrderInsteadCustomersConfirmInput input)
{
return \_orderInsteadGuestService.BuyConfirm(input);
}
/// <summary>
/// 代客下单-支付邀请二维码
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/shop/order/payQRCode")]
public dynamic GetPayQRCode([FromBody]IdInput input)
{
return \_orderInsteadGuestService.GetPayQRCode(input.Id);
}
#region 无货源待结算、已结算、 阶梯提成
/// <summary>
/// 统计无货源批发商订单
/// </summary>
/// <returns></returns>
[HttpPost("/order/notGoodsStorageOrderStatistic")]
public dynamic GetSettleOrder([FromBody]NotSupplyGoodsStorageOrderInput input, [FromBody]PagedCondition paged)
{
return \_notSupplyGoodsStorageService.GetSettleOrder(input, paged);
}
/// <summary>
/// 统计无货源批发商订单详情
/// </summary>
/// <returns></returns>
[HttpPost("/order/notGoodsStorageOrderStatisticDetail")]
public dynamic GetSettleOrder([FromBody]NotSupplyGoodsStorageOrderDetailInput input, [FromBody]PagedCondition paged)
{
return \_notSupplyGoodsStorageService.GetSettleOrderDetail(input, paged);
}
/// <summary>
/// 统计无货源批发商阶梯奖励
/// </summary>
/// <returns></returns>
[HttpPost("/order/notGoodsStorageLadderAwardStatic")]
public dynamic GetLadderAward([FromBody]NotSupplyGoodsStorageLadderAwardInput input, [FromBody]PagedCondition paged)
{
return \_notSupplyGoodsStorageService.GetLadderAward(input, paged);
}
/// <summary>
/// 统计无货源批发商阶梯奖励详情
/// </summary>
/// <returns></returns>
[HttpPost("/order/notGoodsStorageLadderAwardDetail")]
public dynamic GetLadderAwardDetail([FromBody]NotSupplyGoodsStorageLadderAwardDetailInput input, [FromBody]PagedCondition paged)
{
return \_notSupplyGoodsStorageService.GetSettledLadderAward(input, paged);
}
#endregion
/// <summary>
/// 取消订单审核
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[HttpPost("/order/reviewCancel")]
public dynamic ReviewCancel([FromBody]ReviewCancelInput input)
{
return order.ReviewCancel(input);
}
}
}
using Zhidian.Identity.Services;
using Microsoft.AspNetCore.Mvc;
namespace Wholesale.API.Modules
{
public class PartModule : BaseModule
{
private readonly IRoleService rs;
public PartModule(IRoleService \_rs) //: base("/part")
{
rs = \_rs;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Web.Attributes;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("partner")]
public class PartnerController : BaseModule
{
private readonly IPartnerService \_service;
public PartnerController(IPartnerService service) //: base("/partner")
{
\_service = service;
}
/// <summary>
/// 管理员-合伙人-阶梯奖励
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/partner/ladder")]
public dynamic GetManagePartnerLadder([FromBody]PagedCondition page, [FromBody]ManagePartnerLadderInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<ManagePartnerLadderInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManagePartnerLadder(page, input);
}
/// <summary>
/// 管理员-合伙人-提成奖励
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/partner/commission")]
public dynamic GetManagePartnerCommission([FromBody]PagedCondition page, [FromBody]ManagePartnerCommissionInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<ManagePartnerCommissionInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManagePartnerCommission(page, input);
}
/// <summary>
/// 管理员-合伙人-提成奖励-详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/partner/commission/detail")]
public dynamic GetManagePartnerCommissionDetail([FromBody]PagedCondition page, [FromBody]ManagePartnerCommissionDetailInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<ManagePartnerCommissionDetailInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManagePartnerCommissionDetail(page, input);
}
/// <summary>
/// 管理员-合伙人-阶梯奖励-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/partner/ladder/export")]
public dynamic GetManagePartnerLadderExport([FromBody]HeaderItemInput headers, [FromBody]ManagePartnerLadderInput input)
{
//var input = this.BindAndValidate<ManagePartnerLadderInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
return \_service.GetManagePartnerLadderExport(headers, input);
}
/// <summary>
/// 管理员-合伙人-提成奖励-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/partner/commission/export")]
public dynamic GetManagePartnerCommissionExport([FromBody]HeaderItemInput headers, [FromBody]ManagePartnerCommissionInput input)
{
//var input = this.BindAndValidate<ManagePartnerCommissionInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
return \_service.GetManagePartnerCommissionExport(headers, input);
}
/// <summary>
/// 管理员-合伙人-提成奖励-详情-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/partner/commission/detail/export")]
public dynamic GetManagePartnerCommissionDetailExport([FromBody]HeaderItemInput headers, [FromBody]ManagePartnerCommissionDetailInput input)
{
//var input = this.BindAndValidate<ManagePartnerCommissionDetailInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManagePartnerCommissionDetailExport(headers, input);
}
/// <summary>
/// 管理员-批发商-阶梯奖励
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/wholesaler/ladder")]
public dynamic GetManageWholesalerLadder([FromBody]PagedCondition page, [FromBody]ManageWholesalerLadderInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<ManageWholesalerLadderInput>();
return \_service.GetManageWholesalerLadder(page, input);
}
/// <summary>
/// 管理员-批发商-提成奖励
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/wholesaler/commission")]
public dynamic GetManageWholesalerCommission([FromBody]PagedCondition page, [FromBody]ManageWholesalerCommissionInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<ManageWholesalerCommissionInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManageWholesalerCommission(page, input);
}
/// <summary>
/// 管理员-批发商-阶梯奖励-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/wholesaler/ladder/export")]
public dynamic GetManageWholesalerLadderExport([FromBody]HeaderItemInput headers, [FromBody]ManageWholesalerLadderInput input)
{
//var input = this.BindAndValidate<ManageWholesalerLadderInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManageWholesalerLadderExport(headers, input);
}
/// <summary>
/// 管理员-批发商-提成奖励-导出
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("")]
[HttpPost("manage/wholesaler/commission/export")]
public dynamic GetManageWholesalerCommissionExport([FromBody]HeaderItemInput headers, [FromBody]ManageWholesalerCommissionInput input)
{
//var input = this.BindAndValidate<ManageWholesalerCommissionInput>();
//var headers = this.BindAndValidate<HeaderItemInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetManageWholesalerCommissionExport(headers, input);
}
/// <summary>
/// 批发商-奖励收入-预期
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("合伙人收入")]
[HttpPost("wholesaler/expectantreward")]
public dynamic GetWholesalerExpectedReward([FromBody]PagedCondition page, [FromBody]WholesalerRewardInput input)
{
input.IsFromWeb = true;
var result = \_service.GetWholesalerExpectedReward(page, input);
return result.Map<WholesalerExpectedRewardDto, WholesalerExpectedRewardOutput>();
}
/// <summary>
/// 批发商-奖励收入
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("合伙人收入")]
[HttpPost("wholesaler/reward")]
public dynamic GetWholesalerReward([FromBody]PagedCondition page, [FromBody]WholesalerRewardInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<WholesalerRewardInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetWholesalerReward(page, input);
}
/// <summary>
/// 批发商-奖励收入-详情
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("合伙人收入")]
[HttpPost("wholesaler/reward/detail")]
public dynamic GetWholesalerRewardDetail([FromBody]PagedCondition page, [FromBody]IdInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<IdInput>();
return \_service.GetWholesalerRewardDetail(page, input);
}
/// <summary>
/// 批发商-提成收入
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("合伙人收入")]
[HttpPost("wholesaler/commission")]
public dynamic GetWholesalerCommission([FromBody]PagedCondition page, [FromBody]WholesalerCommissionInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<WholesalerCommissionInput>();
var validatePage = this.Validate(input);
if (!validatePage.IsValid)
{
return validatePage.ToResult();
}
return \_service.GetWholesalerCommission(page, input);
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services;
using Zhidian.Common.Web.Attributes;
namespace Wholesale.API.Controllers
{
public class PaymentAccountController : BaseModule
{
private readonly IPaymentAccountService \_service;
public PaymentAccountController(IPaymentAccountService service) // : base()
{
this.\_service = service;
}
/// <summary>
/// 删除支付号与账号关联(同时传入支付号与账号，支付号以及账号下的数据都标记删除)
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[Auth("删除账号下支付号")]
[HttpPost("paymentaccount/del")]
public dynamic PaymentAccountDelete([FromBody]PaymentAccountInput input)
{
//var input = this.Bind<PaymentAccountInput>();
var result = \_service.Delete(input);
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Services.Places;
using Zhidian.Common.Utility.Page;
namespace Wholesale.API.Controllers
{
public class PlaceController : BaseModule
{
IPlaceService placeService;
public PlaceController(IPlaceService placeService) //: base("/place")
{
this.placeService = placeService;
}
[HttpGet("place/provincelist")]
public dynamic GetProvinceList()
{
return placeService.GetProvinceList();
}
[HttpGet("place/citylist/{code}")]
public dynamic GetCityList(string code)
{
string provinceCode = code;// arg.code;
return placeService.GetCityList(provinceCode);
}
[HttpGet("place/arealist/{code}")]
public dynamic GetAreaList(string code)
{
string cityCode = code;// arg.code;
return placeService.GetAreaList(cityCode);
}
[HttpGet("place/streetlist/{code}")]
public dynamic GetStreetList(string code)
{
string areaCode = code;// arg.code;
return placeService.GetStreetList(areaCode);
}
[HttpGet("place/location")]
public dynamic GetGeoLocation(string address)
{
return placeService.GetGeoLocation(address, null);
}
/// <summary>
/// 获取区域划分-销售范围
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpGet("place/zone")]
public dynamic GetPlaceZone()
{
return placeService.GetPlaceZone();
}
/// <summary>
/// 获取获取城市-服务费
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpGet("place/cityfee")]
public dynamic GetPlaceCityList([FromQuery]PagedCondition pager, [FromQuery]PlaceCity input)
{
pager.PageIndex++;
return placeService.GetPlaceCityList(pager, input);
}
}
}
using System;
using System.IO;
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.DTO.Copartners;
using Wholesale.DTO.Products;
using Wholesale.DTO.Products.Search;
using Wholesale.DTO.Products.Update;
using Wholesale.Enums;
using Wholesale.Model;
using Wholesale.Services;
using Wholesale.Services.Products;
using Zhidian.Common.AtuoMapper;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Common.Utility.Results;
using Zhidian.Common.Web.Attributes;
using Zhidian.Ormlite;
using Zhidian.Ormlite.OrmLite;
using Zhidian.Web;
namespace Wholesale.API.Controllers
{
[Route("/product")]
public class ProductController : BaseModule
{
private readonly ICategoryService \_categoryService;
private readonly IProductService \_productService;
private readonly IUnitService \_unitService;
private readonly IGeneratetFileCommService \_generatetFileService;
private readonly IBatchImportProductService \_batchImportProductService;
public ProductController(
ICategoryService categoryService, IProductService productService, IUnitService unitService,
IGeneratetFileCommService generatetFileService,
IBatchImportProductService batchImportProductService
)
{
\_categoryService = categoryService;
\_productService = productService;
\_unitService = unitService;
\_generatetFileService = generatetFileService;
\_batchImportProductService = batchImportProductService;
}
#region sku编辑页
/// <summary>
/// 修改sku参数
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑商品")]
[HttpPut("updatesku")]
public dynamic UpateSku([FromBody]UpdateSku input)
{
//var input = this.Bind<UpdateSku>();
var validateResult = this.Validate(input);
if (!validateResult.IsValid)
{
return validateResult.ToResult();
}
return \_productService.UpdateSku(input);
}
/// <summary>
/// 批量修改sku分类
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("编辑商品")]
[HttpPost("/skus/updatecategory")]
public dynamic BatchUpateSkuCategory([FromBody]BatchUpateSkuCategoryInput input)
{
return \_productService.BatchUpateSkuCategory(input);
}
/// <summary>
/// 修改商品图片
/// </summary>
[Auth("编辑商品")]
[HttpPost("/product/update/image")]
public dynamic UpdateImage([FromBody]AppUpdateImageInput input)
{
return \_productService.UpdateImage(input);
}
/// <summary>
/// 修改sku前，获取一个sku
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看商品")]
[HttpPost("/product/viewsku/description")]
public dynamic GetSku([FromBody]SkuCodeInput input)
{
return \_productService.ViewDescription(input);
}
/// <summary>
/// 修改商品图片
/// </summary>
[Auth("编辑商品")]
[HttpPost("/product/update/description")]
public dynamic UpdateDescription([FromBody]UpdateDescriptionDto input)
{
return \_productService.UpdateDescription(input);
}
/// <summary>
/// 修改sku前，获取一个sku
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看商品")]
[HttpGet("viewsku")]
[HttpGet("sku")]
public dynamic GetSku(string skuCode)
{
//string skuCode = Request.Query.skuCode;
if (skuCode.IsNullOrEmpty())
Result.Error.ValidError("skuCode参数不能为空");
return \_productService.ViewSku(skuCode);
}
#endregion
#region 商品编辑页
/// <summary>
/// 商品编辑页，加载商品信息
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看商品")]
[HttpGet("view")]
public dynamic GetProduct(string skuCode)
{
//string skuCode = Request.Query.skuCode;
if (skuCode.IsNullOrEmpty())
Result.Error.ValidError("skuCode参数不能为空");
return \_productService.ViewProduct(skuCode);
}
#endregion
#region 商品列表页
/// <summary>
/// 获取商品列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看商品")]
[HttpGet("Search")]
public dynamic Search(PagedCondition page, ProductSearchInput input)
{
//var page = this.Bind<PagedCondition>();
//var input = this.Bind<ProductSearchInput>();
var vResult = this.Validate(input);
if (!vResult.IsValid)
{
return vResult.ToResult();
}
page.PageIndex++;
return \_productService.Search(page, input);
}
/// <summary>
/// 获取商品列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("查看商品")]
[HttpGet("shopDecoration/search")]
public dynamic SearchV2(PagedCondition page, ProductSearchInput input)
{
var vResult = this.Validate(input);
if (!vResult.IsValid)
{
return vResult.ToResult();
}
return \_productService.ShopDecorationSearch(page, input);
}
/// <summary>
/// 获取下拉列表仓库名称
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("查看商品")]
[HttpGet("shopDecoration/storagenames")]
public dynamic GetDDLShopNames([FromBody] CopartnerBindStorageInput input)
{
return \_productService.GetDDLStorageNames(input);
}
#endregion
#region 商品发布页
/// <summary>
/// 商品发布
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[Auth("新增商品")]
[HttpPost("publish")]
public dynamic Publish([FromBody]PublishProductInput input)
{
//var input = this.Bind<PublishProductInput>();
var vResult = this.Validate(input);
if (!vResult.IsValid)
{
return vResult.ToResult();
}
return \_productService.PublishFromWeb(input);
}
#endregion
#region 辅助方法
/// <summary>
/// 商品发布时，搜索类别列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet("searchCategories")]
public dynamic SearchCategories(string Key)
{
string key = Key;// Request.Query.Key;
var result = \_categoryService.GetCategoriesTree();
if (!result.IsSuccessed)
{
return result;
}
key = key.Trim();
var tree = result.Data;
var postResult = Result<object>.Success;
var list = tree.SelectMany(it =>
it.Children.SelectMany(it1 => it1.Children.Where(child => child.Name.Contains(key))))
.Select(it => new
{
Name = it.ToString(),
No = it.No,
CategoryId1 = it.Parent.Parent.Id,
CategoryId2 = it.Parent.Id,
CategoryId3 = it.Id,
});
postResult.Data = list;
return postResult;
}
/// <summary>
/// 商品发布时，获取类别列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet("categories")]
public dynamic GetCategories(string ParentId, string ParentNo, string StorageId, int CategoryLevel)
{
string parentId = ParentId;// Request.Query.ParentId;
string parentNo = ParentNo;// Request.Query.ParentNo;
string storageId = StorageId;
int categoryLevel = CategoryLevel;
if (!string.IsNullOrWhiteSpace(parentNo))
{
//根据编码
int[] noArr = parentNo.Split(',').Select(it => it.ToInt()).ToArray();
return \_productService.GetCategoriesByNo(noArr);
}
else
{
//根据id
string[] ids = string.IsNullOrWhiteSpace(parentId)
? new string[] { "0" }
: parentId.Split(',');
return \_productService.GetCategories(ids, storageId, categoryLevel);
}
}
/// <summary>
/// 商品发布时，获取单位列表
/// </summary>
/// <param name="parameters"></param>
/// <returns></returns>
[HttpGet("units")]
public dynamic GetUnits()
{
var result = \_unitService.SearchUnits();
if (!result.IsSuccessed)
return result;
return result.Map<Unit, UnitDto>();
}
#endregion
#region 图库方法
[HttpGet("searchCommodity")]
public dynamic SearchCommodity(string Key, [FromQuery]PagedCondition pager)
{
pager.PageIndex++;
string key = Key;// Request.Query.Key;
//var pager = this.Bind<PagedCondition>();
string json = \_productService.SearchCommodityByTuku(pager, key);
return Content(json);
}
[HttpGet("getCommodity")]
public dynamic GetCommodity(string Id)
{
string key = Id; //Request.Query.Id;
string json = \_productService.GetCommodityByTuku(key);
return Content(json);
}
#endregion
[UnAuth]
[HttpGet("description")]
public dynamic GetDescription(string skuCode)
{
//string skuCode = Request.Query.skuCode;
return \_productService.GetDescription(skuCode);
}
/// <summary>
/// 导出
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("导出商品")]
[HttpPost("excel")]
public dynamic Derivation([FromBody]ProductSearchInput input)
{
//var input = this.Bind<ProductSearchInput>();
//var headers = this.Bind<HeaderItemInput>();
var workbook = \_productService.Derivation(input);
MemoryStream ms = new MemoryStream();
workbook.Write(ms);
ms.Seek(0, SeekOrigin.Begin);
return File(ms, "application/vnd.ms-excel", $"{"商品导出\_"}{ DateTime.Now.ToString("yyyyMMdd\_HHmmss")}{".xls"}");
}
/// <summary>
/// 商品导出
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("导出商品")]
[HttpPost("exportexcel")]
public dynamic ExportExcel([FromBody]ProductSearchInputV2 input)
{
//var input = this.Bind<ProductSearchInputV2>();
//var headers = this.Bind<HeaderItemInput>();
var result = \_generatetFileService.AddTask(new AddGeneratetFileInput
{
Headers = "{}",
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.商品导出Excel,
TaskName = $"商品导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"商品导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
[HttpGet("status")]
public dynamic GetProductStatus(string StorageId, string SkuCode)
{
string storageId = StorageId;// Request.Query.StorageId;
string skuCode = SkuCode;// Request.Query.SkuCode;
var result = \_productService.GetProductStatus(storageId, skuCode);
return result;
}
/// <summary>
/// 根据分类id获取设置的规格
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[HttpGet("category/spec")]
public dynamic GetSpecByCategoryId(string categoryId, string categoryNo)
{
//var categoryId = Request.Query.categoryId;
//var categoryNo = Request.Query.categoryNo;
if (!string.IsNullOrWhiteSpace(categoryNo))
{
return \_productService.GetSpecByCategoryNo(categoryNo.ToInt());
}
else
{
return \_productService.GetSpecByCategoryId(categoryId);
}
}
/// <summary>
/// 批量上架
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("编辑商品")]
[HttpPost("onsale")]
public dynamic LetOnSale([FromBody]SkuCodesInput ids)
{
return \_productService.LetOnSale(ids.SkuCodes);
}
/// <summary>
/// 批量下架
/// </summary>
/// <param name="arg"></param>
/// <returns></returns>
[Auth("编辑商品")]
[HttpPost("offsale")]
public dynamic LetOffSale([FromBody]SkuCodesInput ids)
{
return \_productService.LetOffSale(ids.SkuCodes);
}
/// <summary>
/// 无货源批发商改销售价
/// </summary>
/// <param name="input"></param>
/// <returns></returns>
[Auth("编辑商品")]
[HttpPost("nostock/saleprice")]
public dynamic NoStockUpdateSalePrice([FromBody]NoStockSalePriceInput input)
{
return \_productService.NoStockUpdateSalePrice(input);
}
#region sku价格管理
/// <summary>
/// 获取三级分类扣点
/// </summary>
[HttpPost("CategoriesOffset")]
public dynamic CategoriesOffset([FromBody]string smallCategory)
{
//var model = this.BindToDynamic();
//string smallCategory = model.smallCategory;
return \_productService.CategoriesOffset(smallCategory);
}
#endregion
[Auth("新增商品")]
[HttpPost("BatchImportProducts")]
public dynamic BatchImportProducts()
{
var file = HttpContext.Request.Form.Files[0];
var result = \_batchImportProductService.ImportProductExcel(file.OpenReadStream(), file.FileName);
return result;
}
[Auth("删除商品")]
[HttpPost("/product/del")]
public dynamic DeleteSku([FromBody]SkuCodeInput input)
{
return \_productService.DeleteSku(input);
}
}
}
using System;
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO;
using Wholesale.Enums;
using Wholesale.Services;
using Wholesale.Services.Provider;
using Zhidian.Common.Utility.Extensions;
using Zhidian.Common.Utility.Page;
using Zhidian.Identity.Services;
using Zhidian.Ormlite.OrmLite;
namespace Wholesale.API.Controllers
{
/// <summary>
/// 供应商管理
/// </summary>
[Route("/provider")]
public class ProviderController : BaseModule
{
private readonly IProviderService \_providerService;
private readonly IdentityService \_identityService;
private readonly IGeneratetFileCommService \_generatetFileCommService;
public ProviderController(IProviderService providerService, IdentityService \_identityService, IGeneratetFileCommService \_generatetFileCommService) //: base("/provider")
{
this.\_providerService = providerService;
this.\_identityService = \_identityService;
this.\_generatetFileCommService = \_generatetFileCommService;
}
/// <summary>
/// 是否存在
/// </summary>
/// <returns></returns>
[HttpGet("existProvider")]
public dynamic ExistProvider([FromQuery]ProviderInputDto providerInputDto)
{
//ProviderInputDto providerInputDto = this.Bind<ProviderInputDto>();
return \_providerService.ExistProvider(providerInputDto);
}
/// <summary>
/// 添加供应商
/// </summary>
/// <param name="providerInputDto"></param>
/// <returns></returns>
[HttpPost("addProvider")]
public dynamic AddProvider([FromBody]ProviderInputDto providerInputDto)
{
//ProviderInputDto providerInputDto = this.Bind<ProviderInputDto>();
return \_providerService.AddProvider(providerInputDto);
}
/// <summary>
/// 删除
/// </summary>
/// <param name="providerInputDto"></param>
/// <returns></returns>
[HttpPost("deleteProvider")]
public dynamic DeleteProvider([FromBody]ProviderInputDto providerInputDto)
{
//ProviderInputDto providerInputDto = this.Bind<ProviderInputDto>();
return \_providerService.DeleteProvider(providerInputDto);
}
/// <summary>
/// 修改
/// </summary>
/// <param name="providerInputDto"></param>
/// <returns></returns>
[HttpPost("updateProvider")]
public dynamic UpdateProvider([FromBody]ProviderInputDto providerInputDto)
{
//ProviderInputDto providerInputDto = this.Bind<ProviderInputDto>();
return \_providerService.UpdateProvider(providerInputDto);
}
/// <summary>
/// 获取
/// </summary>
/// <param name="providerInputDto"></param>
/// <returns></returns>
[HttpGet("getProviders")]
public dynamic GetProviders([FromQuery] ProviderSearchInput search, [FromQuery]PagedCondition paged)
{
paged.PageIndex++;
var user = \_identityService.GetCurrentUserOrThrow();
//var paged = this.Bind<PagedCondition>();
//var search = this.Bind<ProviderSearchInput>();
search.StorageId = user.StroageId;
return \_providerService.GetProviders(paged, search);
}
/// <summary>
/// 根据Id获取供应商基本信息
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpGet("getProviderById")]
public dynamic GetProviderById([FromQuery]ProviderInputDto providerInputDto)
{
//ProviderInputDto providerInputDto = this.Bind<ProviderInputDto>();
return \_providerService.GetProviderById(providerInputDto);
}
/// <summary>
/// 导出供应商列表
/// </summary>
/// <param name="args"></param>
/// <returns></returns>
[HttpPost("exportPorviders")]
public dynamic ExportProviderToExcel([FromBody]ProviderSearchInput input, [FromBody]HeaderItemInput headers)
{
var user = \_identityService.GetCurrentUserOrThrow();
//var input = this.Bind<ProviderSearchInput>();
//var headers = this.Bind<HeaderItemInput>();
var result = \_generatetFileCommService.AddTask(new AddGeneratetFileInput
{
Headers = headers.ToJson(),
Args = input.ToJson(),
DwonloadFileWay = input.DwonloadFileWay,
TaskType = ETaskType.供应商列表导出,
TaskName = $"供应商列表导出{DateTime.Now.ToString("yyyyMMdd")}",
FileName = $"供应商列表导出{DateTime.Now.ToString("yyyyMMddHHmmss")}.xls",
});
return result;
}
}
}
using Microsoft.AspNetCore.Mvc;
using Wholesale.DTO.Purchase;
using Wholesale.Services.Purchase;
using Zhidian.Common.Utility.Page;
using Zhidian.Ormlite.OrmLite;
namespace Wholesale.API.Controllers
{
[Route("/Purchase")]
public class PurchaseController : BaseModule
{
private readonly IPurchaseOrderService \_purchaseOrderService;
public PurchaseController(IPurchaseOrderService purchaseOrderService) //: base("/Purchase")
{
this.\_purchaseOrderService = purchaseOrderService;
}
/// <summary>
/// 添加采购单
/// </summary>
[HttpPost("AddOrder")]
public dynamic AddOrder([FromBody]PurchaseOrderInput input)
{
//var input = this.Bind<PurchaseOrderInput>();
return \_purchaseOrderService.PurchaseOrderAdd(input);
}
/// <summary>
/// 添加退货单
/// </summary>
[HttpPost("AddReturnOrder")]
public dynamic AddReturnOrder([FromBody]PurchaseReturnOrderInput input)
{
//var input = this.Bind<PurchaseReturnOrderInput>();
return \_purchaseOrderService.PurchaseReturnOrderAdd(input);
}
/// <summary>
/// 选择采购sku列表
/// </summary>
[HttpGet("SkuList")]
public dynamic PurchaseSkuList([FromQuery]PurchaseSkuListInput input, [FromQuery]PagedCondition page)
{
page.PageIndex++;
//var input = this.Bind<PurchaseSkuListInput>();
//var page = this.Bind<PagedCondition>();
return \_purchaseOrderService.PurchaseSkuList(page, input);
}
[HttpGet("PurchaseOrderSkuList")]
public dynamic PurchaseOrderSkuList(string OrderId, [FromQuery]PagedCondition page)
{
page.PageIndex++;
//var page = this.Bind<PagedCondition>();
string orderId = OrderId;// this.Request.Query.OrderId;
return \_purchaseOrderService.PurchaseOrderSkuList(page, orderId);
}
/// <summary>
/// 查询采购单列表
/// </summary>
[HttpGet("OrderList")]
public dynamic PurchaseOrderList([FromQuery]PagedCondition page, [FromQuery]PurchaseOrderSearchInput input)
{
page.PageIndex++;
//var input = this.Bind<PurchaseOrderSearchInput>();
//var page = this.Bind<PagedCondition>();
return \_purchaseOrderService.PurchaseOrderList(page, input);
}
/// <summary>
/// 采购单详细
/// </summary>
[HttpGet("OrderDetail")]
public dynamic PurchaseOrderDetail(string OrderId)
{
string orderId = OrderId;// this.Request.Query.OrderId;
return \_purchaseOrderService.PurchaseOrderDetail(orderId);
}
/// <summary>
/// 查询采购退货单列表
/// </summary>
[HttpGet("ReturnOrderList")]
public dynamic PurchaseReturnOrderList([FromQuery]PurchaseOrderSearchInput input, [FromQuery]PagedCondition page)
{
page.PageIndex++;
//var input = this.Bind<PurchaseOrderSearchInput>();
//var page = this.Bind<PagedCondition>();
return \_purchaseOrderService.PurchaseReturnOrderList(page, input);
}
/// <summary>
/// 退货单详情
/// </summary>
[HttpGet("ReturnOrderDetail")]
public dynamic PurchaseReturnOrderDetail(string ReOrderId)
{
string reOrderId = ReOrderId;// this.Request.Query.ReOrderId;
return \_purchaseOrderService.PurchaseReturnOrderDetail(reOrderId);
}
/// <summary>
/// 采购入库单列表导出任务
/// </summary>
[HttpPost("OrderList/ExportTask")]
public dynamic PurchaseOrderListExportTask([FromBody]HeaderItemInput header, [FromBody]PurchaseOrderSearchInput input)
{
//var input = this.Bind<PurchaseOrderSearchInput>();
//var header = this.Bind<HeaderItemInput>();
return \_purchaseOrderService.PurchaseOrderListExportTask(input, header);
}
/// <summary>
/// 采购退货单列表导出任务
/// </summary>
[HttpPost("ReturnOrderList/ExportTask")]
public dynamic PurchaseReturnOrderListExportTask([FromBody]HeaderItemInput header, [FromBody]PurchaseOrderSearchInput input)
{
//var input = this.Bind<PurchaseOrderSearchInput>();
//var header = this.Bind<HeaderItemInput>();
return \_purchaseOrderService.PurchaseReturnOrderListExportTask(input, header);
}
}
}**