

xStringService

it is a Part of xDashboard on SaherElm IT Center which provides:

- all requirements to Implement String Resource Features in a **xDashboard** based Project.

this module has following dependencies :

- xDataService
- xPushService
- xIdentityService

for configure and use this Module refer to DI.XDIHelperExtension.cs file.

String Resource

an string resource is a Key/Value based structure which used in Multi Language Applications for accessing Translated of Spcified Values by Key, in Specified Languages. this architecture is very useful for implementing other services in an Application. such as Terms and Conditions.

Data Persists

all of Provided features of this Module is works based on Data Persistence. so data models is Very Important. in this section all Data models explained.

data manipulation done using **xDataService** provided Features.

- **XString**: an Entity Model for Mapping to Tables.
- **XStringDto**: a Data Transfer Object for Representing a Data Model.
- **XStringEntityConfiguration**: a Configuration file for XString Entity.
- **XStringEntityHub**: a Hub Implementation for XString Entity Model.
- **XStringDtoHub**: a Hub Implementation for XStringDto Model.
- **XStringEntityRegistrar**: an Implementation class for Dynamically Register XString Entity in a DbContext.
- **IXStringRepositoryEvents**: an Interface for Repository Events Describing of XString Entity.
- **XStringRepositoryEvents**: an Implementation for Repository Events of XString Entity.
- **IXStringKeyGenerator**: an Interface for Describing Key Generator Actions for XString Entity.
- **XStringKeyGenerator**: an Implementation for Describing Key Generator Actions for XString Entity.
- **IXStringRepository**: an Interface for Describing XString Repository Actions.
- **XStringEFRepository**: an Implementation of EF Core based XString Repository Actions.
- **XStringMongoRepository**: an Implementation of MongoDB based XString Repository Actions.
- **XStringMongoRepository**: an Implementation of MongoDB based XString Repository Actions.
- **XStringInMemoryRepository**: an Implementation of InMemory based XString Repository Actions.
- **IXStringSeeder**: an Interface for Describe XString DbSeeder actions.
- **XStringSeederBase**: an abstraction of Implementation of XString DbSeeder actions.
- **IXStringRepositoryProvider**: an Interface for Describing XString Repository Provider (using Hubs) actions.
- **XStringRepositoryProvider**: an Implementation of XString Repository Provider (using Hubs) actions.

- **IXStringRepositoryService**: an Interface for Describing XString Repository Service Actions (using XStringDto).
- **XStringRepositoryService**: an Implementation of XString Repository Service Actions (using XStringDto).
- **IXStringServiceProvider**: an Interface for Describing XString Repository Service Actions Provider (using Hubs and XStringDto).
- **XStringServiceProvider**: an Implementation of XString Repository Service Actions Provider (using Hubs and XStringDto).
- **XStringGraphType**: introduce XString Data model for GraphQL using.
- **XStringGraphQuery**: introduce Repository Implementation of XString model actions as Query Actions for GraphQL using.
- **XStringGraphSchema**: introduce XString Query Schema for GraphQL using.
- **IXStringGraphQLTypeHelper**: an Interface for Describing XString GraphQL Type Helper Provided Actions.
- **IXStringGraphQLTypeHelper**: an Implementation of XString GraphQL Type Helper Provided Actions.

data persistence almost is a Down Layer works which handled using Service itself. but in some Used cases of Applications Business Logics, for Manipulating resources you can use above Registered Services by Injecting them.

Providers

this module actually used to act as a base Provider for implementing Features based on it. for this purpose, provides some Features which described in this section.

String Resource Provider

some services used XString model for Provide Specially Concept's of Actions. for example Terms and Conditions Services use XString Persisted Resource for Specified Purpose.

this Module provides some sort of tools for this purpose. which described in this section.

Data Transfer Models

this service used some Data Transfer Models for Data Providing. which Described in this section.

```
public class XLocaleResourceDto : XBaseDto
{
    public string Language { get; set; }
    public string Value { get; set; }
}

public class XResourceDto : XBaseDto
{
    public string Resource { get; set; }
    public IEnumerable<XLocaleResourceDto> Locales { get; set; } = new
    HashSet<XLocaleResourceDto>();
}
```

IXBaseStringResourceProvider

an interface for Describe Provided Actions of Resource based Service.

```

/// <summary>
/// a Base Class for Handling Resources Based on Specified Types on
/// String Resources Use ...
/// such as:
/// - Terms and Conditions;
/// - Contents;
/// - News;
/// - etc ...
/// </summary>
public interface IXBaseStringResourceProvider
{
    //
    #region Props ...
    string Prefix { get; }
    IXStringServiceProvider Provider { get; }
    XAppConfiguration AppConfiguration { get; }
    #endregion

    //
    #region Actions ...
    /// <summary>
    /// Prepare Resource Title by Given Data ...
    /// </summary>
    /// <param name="suffix"></param>
    /// <returns></returns>
    string GetResourceTitle(string suffix);

    /// <summary>
    /// Add Specified Locales ...
    /// </summary>
    /// <param name="suffix"></param>
    /// <param name="locales"></param>
    /// <param name="connectionId"></param>
    /// <param name="cancellationToken"></param>
    /// <returns></returns>
    Task<IEnumerable<XStringDto>> Add(
        string suffix,
        IEnumerable<XLocaleResourceDto> locales,
        string connectionId = null,
        CancellationToken cancellationToken = default
    );

    /// <summary>
    /// Add Or Update Specified Locals ...
    /// </summary>
    /// <param name="resource"></param>
    /// <param name="locales"></param>
    /// <param name="connectionId"></param>

```

```
/// <param name="cancellationToken"></param>
/// <returns></returns>
Task<IEnumerable<XStringDto>> AddOrUpdate(
    string resource,
    IEnumerable<XLocaleResourceDto> locales,
    string connectionId = null,
    CancellationToken cancellationToken = default
);

/// <summary>
/// Get Specified Locale ...
/// </summary>
/// <param name="suffix"></param>
/// <param name="language"></param>
/// <param name="cancellationToken"></param>
/// <returns></returns>
Task<XLocaleResourceDto> GetLocale(
    string suffix,
    string language = null,
    CancellationToken cancellationToken = default
);

/// <summary>
/// Add Specified Resource ...
/// </summary>
/// <param name="suffix"></param>
/// <param name="item"></param>
/// <param name="connectionId"></param>
/// <param name="cancellationToken"></param>
/// <returns></returns>
Task<XResourceDto> AddResource(
    string suffix,
    XResourceDto item,
    string connectionId = null,
    CancellationToken cancellationToken = default
);

/// <summary>
/// Get Resources of Specified Resource ...
/// </summary>
/// <param name="suffix"></param>
/// <param name="cancellationToken"></param>
/// <returns></returns>
Task<XResourceDto> GetResource(
    string suffix,
    CancellationToken cancellationToken = default
);

/// <summary>
/// Remove Specified Suffix Resources ...
/// </summary>
/// <param name="suffix"></param>
/// <param name="connectionId"></param>
/// <param name="cancellationToken"></param>
```

```

    /// <returns></returns>
    Task Remove(
        string suffix,
        string connectionId = null,
        CancellationToken cancellationToken = default
    );

    /// <summary>
    /// Remove Specified Resource ...
    /// </summary>
    /// <param name="resource"></param>
    /// <param name="language"></param>
    /// <param name="connectionId"></param>
    /// <param name="cancellationToken"></param>
    /// <returns></returns>
    Task Remove(
        string resource,
        string language,
        string connectionId = null,
        CancellationToken cancellationToken = default
    );
    #endregion
}

```

XBaseStringResourceProvider

an implementation of Provided Actions of Resource based Service.

```

    /// <summary>
    /// a Base Class for Handling Resources Based on Specified Types on
    /// String Resources Use ...
    /// such as:
    /// - Terms and Conditions;
    /// - Contents;
    /// - News;
    /// - etc ...
    /// </summary>
    public abstract class XBaseStringResourceProvider : IXBaseStringResourceProvider
    {
        //
        #region Props ...
        public string Prefix { get; }
        public IXStringServiceProvider Provider { get; }
        public XAppConfiguration AppConfiguration { get; }
        #endregion

        //
        #region Constructor ...
        public XBaseStringResourceProvider(
            string prefix,
            IXStringServiceProvider provider,

```

```

        XAppConfiguration appConfiguration
    )
    {
        //
        Prefix = prefix;
        Provider = provider;
        AppConfiguration = appConfiguration;
    }
#endregion

//
#region Actions ...
/// <summary>
/// Prepare Resource Title by Given Data ...
/// </summary>
/// <param name="suffix"></param>
/// <returns></returns>
public string GetResourceTitle(string suffix)
{
    //
    string result = "";

    //
    if (suffix.IsNullOrEmpty())
    {
        return result;
    }

    //
    result = $"{Prefix}_{suffix}";

    //
    return result;
}

/// <summary>
/// Add Specified Locales ...
/// </summary>
/// <param name="suffix"></param>
/// <param name="locales"></param>
/// <param name="connectionId"></param>
/// <param name="cancellationToken"></param>
/// <returns></returns>
public async Task<IEnumerable<XStringDto>> Add(
    string suffix,
    IEnumerable<XLocaleResourceDto> locales,
    string connectionId = null,
    CancellationToken cancellationToken = default
)
{
    //
    // Validate ...
    var has =
        locales.HasChild() &&

```

```

        !suffix.IsNullOrEmpty();
    if (!has)
    {
        XException.InvalidArgs.Throw();
    }

    //
    // Prepare Resource ID ...
    string resourceID = GetResourceTitle(suffix);
    var result = await locales.SelectAsync(async l =>
    {
        //
        // Normalize Language ...
        if (l.Language.IsNullOrEmpty())
        {
            l.Language = AppConfiguration.DefaultLanguage;
        }

        //
        // Prepare Model for Add ...
        var model = new XStringDto
        {
            Language = l.Language,
            TranslatedValue = l.Value,
            ResourceTitle = resourceID,
        };
        model = await Provider.AddAsync(
            item: model,
            saveChanges: true,
            connectionId: connectionId,
            cancellation_token: cancellationToken
        );
        return model;
    });

    //
    return result;
}

/// <summary>
/// Add Or Update Specified Locals ...
/// </summary>
/// <param name="resource"></param>
/// <param name="locales"></param>
/// <param name="connectionId"></param>
/// <param name="cancellation_token"></param>
/// <returns></returns>
public async Task<IEnumerable<XStringDto>> AddOrUpdate(
    string resource,
    IEnumerable<XLocaleResourceDto> locales,
    string connectionId = null,
    Cancellation_token cancellationToken = default
)
{

```

```

        //
        // Validate ...
        bool isValid =
            locales.HasChild() &&
            !resource.IsNullOrEmpty() &&
            locales.All(l => !l.Value.IsNullOrEmpty() &&
!l.Language.IsNullOrEmpty());
        if (!isValid)
        {
            XException.InvalidArgs.Throw();
        }

        //
        // Check Resource ...
        isValid = resource.Contains(Prefix);
        if (!isValid)
        {
            XException.NotAllowed.Throw();
        }

        //
        var result = await locales.SelectAsync(async l =>
            await Provider.AddOrUpdateByLocaleResource(
                item: l,
                resource: resource,
                connectionId: connectionId,
                cancellationToken: cancellationToken
            )
        );

        //
        return result;
    }

    /// <summary>
    /// Get Specified Locale ...
    /// </summary>
    /// <param name="suffix"></param>
    /// <param name="language"></param>
    /// <param name="cancellationToken"></param>
    /// <returns></returns>
    public async Task<XLocaleResourceDto> GetLocale(
        string suffix,
        string language = null,
        CancellationToken cancellationToken = default
    )
    {
        //
        // Normalize ...
        if (language.IsNullOrEmpty())
        {
            language = AppConfiguration.DefaultLanguage;
        }
    }

```

```
//  
// Validate ...  
if (suffix.IsNullOrEmpty() || language.IsNullOrEmpty())  
{  
    XException.InvalidArgs.Throw();  
}  
  
//  
// Prepare Resource Title ...  
string resource = GetResourceTitle(suffix);  
  
//  
// Check Resource Exists ...  
var isExists = await Provider.IsResourceExists(  
    resource: resource,  
    language: language,  
    cancellationToken: cancellationToken  
);  
if (!isExists)  
{  
    XException.NotFound.Throw();  
}  
  
//  
// Retrieve and Validate Dto Model ...  
var model = await Provider.Get(  
    resource: resource,  
    language: language,  
    cancellationToken: cancellationToken  
);  
if (model.IsNullOrEmpty())  
{  
    XException.ActionFailed.Throw();  
}  
  
//  
// Prepare Result ...  
var result = model  
    .ToXLocaleResourceDto();  
return result;  
}  
  
/// <summary>  
/// Add Specified Resource ...  
/// </summary>  
/// <param name="suffix"></param>  
/// <param name="item"></param>  
/// <param name="connectionId"></param>  
/// <param name="cancellationToken"></param>  
/// <returns></returns>  
public async Task<XResourceDto> AddResource(  
    string suffix,  
    XResourceDto item,  
    string connectionId = null,
```

```

        CancellationToken cancellationToken = default
    )
    {
        //
        // Validate ...
        if (suffix.IsNullOrEmpty() ||
            item.IsNull() ||
            !item.Locales.HasChild() ||
            !item.Locales.All(l => !l.Value.IsNullOrEmpty()))
        {
            XException.InvalidArgs.Throw();
        }

        //
        // Prepare Resource ID ...
        var addedLocals = await Add(
            suffix: suffix,
            locales: item.Locales,
            connectionId: connectionId,
            cancellationToken: cancellationToken
        );
        if (!addedLocals.HasChild())
        {
            XException.ActionFailed.Throw();
        }

        //
        var result = await GetResource(
            suffix: suffix,
            cancellationToken: cancellationToken
        );
        return result;
    }

    /// <summary>
    /// Get Resources of Specified Resource ...
    /// </summary>
    /// <param name="suffix"></param>
    /// <param name="cancellationToken"></param>
    /// <returns></returns>
    public async Task<XResourceDto> GetResource(
        string suffix,
        CancellationToken cancellationToken = default
    )
    {
        //
        // Validate ...
        if (suffix.IsNullOrEmpty())
        {
            XException.InvalidArgs.Throw();
        }

        //
        // Prepare Resource ID ...

```

```
        string resourceID = GetResourceTitle(suffix);

        //
        // Prepare ...
        var result = await Provider.GetResource(
            resource: resourceID,
            cancellation_token: cancellationToken
        );
        return result;
    }

    /// <summary>
    /// Remove Specified Suffix Resources ...
    /// </summary>
    /// <param name="suffix"></param>
    /// <param name="connectionId"></param>
    /// <param name="cancellationToken"></param>
    /// <returns></returns>
    public async Task Remove(
        string suffix,
        string connectionId = null,
        CancellationToken cancellationToken = default
    )
    {
        //
        // Validate ...
        if (suffix.IsNullOrEmpty())
        {
            XException.InvalidArgs.Throw();
        }

        //
        var resourceID = GetResourceTitle(suffix);
        await Provider.RemoveResource(
            resource: resourceID,
            connectionId: connectionId,
            cancellationToken: cancellationToken
        );
    }

    /// <summary>
    /// Remove Specified Resource ...
    /// </summary>
    /// <param name="resource"></param>
    /// <param name="language"></param>
    /// <param name="connectionId"></param>
    /// <param name="cancellationToken"></param>
    /// <returns></returns>
    public async Task Remove(
        string resource,
        string language,
        string connectionId = null,
        CancellationToken cancellationToken = default
    )
```

```
{
    //
    // Validate ...
    bool isValid =
        !resource.IsNullOrEmpty() &&
        !language.IsNullOrEmpty();
    if (!isValid)
    {
        XException.InvalidArgs.Throw();
    }

    //
    // Check resource ID Contains Prefix ...
    isValid = resource.Contains(Prefix);
    if (!isValid)
    {
        XException.NotAllowed.Throw();
    }

    //
    // Check Exists ...
    var isExists = await Provider.IsResourceExists(
        resource: resource,
        language: language,
        cancellationToken: cancellationToken
    );
    if (!isExists)
    {
        XException.NotFound.Throw();
    }

    //
    // Remove Resource ...
    await Provider.Remove(
        resource: resource,
        language: language,
        connectionId: connectionId,
        cancellationToken: cancellationToken
    );
}
#endregion
}
```

Implementation

you have to follow these steps for using this Module.

- **DI Registration:** in this phase, all Provided Services Registered in DI.
- **Middleware Usage:** in this phase, Service Middlewares Use to Handle Features.

```
public class Startup
{
    //
    public void ConfigureServices(IServiceCollection services)
    {
        ...
        //
        // Register String Service ...
        services.AddXStringService<XApiDbContext>(
            lifeTime: lifeTime,
            repositoryType: xDataService.Constants.XRepositoryType.EF
        );
        ...
    }

    //
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        ...
        //
        // Using String Service ...
        app.UseXStringService(
            isDevelopmentEnvironment: isDevelopmentEnvironment
        );
        ...
    }
}
```

Maintainer

Hadi Khazaei asl

<https://www.saherelm.ir>

hadi_khazaei_asl@yahoo.com