

xPushService

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

- all requirements to Implement Realtime Communications in a **xDashboard** based Project.

this module has following dependencies:

- xModels
- xCommons
- xDataService
- xIdentityService

all tools which provides realtime communications features in this module constructed based on SignalR library.

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

Implementation

Hub

it is a High Level Communication Endpoint which allow Bi-Directional Communications (Server => Client and Client => Server) using Realtime Connections. which introduced using **SignalR**.

XBaseHub

all Supported Hubs in **xDashboard** must inherit from this based abstract class, which provides base requirement which used in all Hubs.

XBaseWebRTCHub

a base abstract class which provides all requirements to establish WebRTC based Communications which called Signalling.

if you want to Implement WebRTC based Hubs for Audio/Vide/... establish connections. you have to extends from this base abstraction class.

XBaseEntityHub

an abstract class which Provides all requirements to Implement Entity Manipulations Notifications in your Projects.

```
public class XTestEntityHub : XBaseEntityHub<XTest, Guid>, ITestEntityHub
{
    public XTestEntityHub(ILogger<XBaseHub> logger) : base(logger)
    { }
}
```

XBaseDtoHub

an abstract class which Provides all requirements to Implement Dto Manipulations Notifications in your Projects.

```
public class XTestDtoHub : XBaseDtoHub<XTest, XTestDto, Guid>, ITestDtoHub
{
    public XTestDtoHub(ILogger<XTestDtoHub> logger) : base(logger)
    { }
}
```

Provider

a Provider is a Service which has access to Hub or Hubs for Realtime Communications in other words Push Notification Enabled Service.

XBaseEntityProvider

an abstract class which has Repository based Data Manipulation Implemented actions, which used EntityHub for Notifying Entity Changes.

```
public class XTestRepositoryProvider : XBaseEntityProvider<XTest, Guid,
XTestEntityHub>, ITestRepositoryProvider
{
    public XTestRepositoryProvider(
        ITestRepository repository,
        IHubContext<XTestEntityHub> hub,
        XDataServiceConfiguration dataConfiguration,
        IXIdentityProvider identityProvider = null
    ) : base(
        hub: hub,
        repository: repository,
        identityProvider: identityProvider,
        dataConfiguration: dataConfiguration
    )
    { }
}
```

XBaseDtoProvider

an abstract class which has Service based Data Manipulation Implemented actions, which used DtoHub for Notifying Dto Changes.

```
public class XTestServiceProvider : XBaseDtoProvider<XTest, XTestDto, Guid,
XTestDtoHub>, IXTestServiceProvider
{
    public XTestServiceProvider(
        IHubContext<XTestDtoHub> hub,
        IXTestRepositoryService service,
        XDataServiceConfiguration dataConfiguration,
        IXIdentityProvider identityProvider = null
    ) : base(
        hub: hub,
        service: service,
        identityProvider: identityProvider,
        dataConfiguration: dataConfiguration
    )
    { }
}
```

XBaseRepositoryProviderController

an abstract Controller class which has Identity Enabled Services for Authorization and Authentication based Endpoint Providing and also has all default Entity based Provider Service Actions implementation for using in Projects.

```
public class XTestRepositoryProviderController :
XBaseRepositoryProviderController<XTest, Guid, XTestEntityHub>,
IXBaseRepositoryProviderController<XTest, Guid, XTestEntityHub>
{
    public XTestRepositoryProviderController(
        ILogger<XTestRepositoryProviderController> logger,
        XAppConfiguration appConfiguration,
        IXIdentityProvider identityProvider,
        XValidationProvider validationProvider,
        IXTestRepositoryProvider provider,
        Func<IQueryable<XTest>, IOrderedQueryable<XTest>> defaultOrderBuilder,
        Func<IQueryable<XTest>, IIncludableQueryable<XTest, object>>
defaultIncludeBuilder
    ) : base(
        logger,
        appConfiguration,
        identityProvider,
        validationProvider,
        provider,
        defaultOrderBuilder,
        defaultIncludeBuilder
    )
}
```

```
{ }
}
```

XBaseServiceProviderController

an abstract Controller class which has Identity Enabled Services for Authorization and Authentication based Endpoint Providing and also has all default Dto based Provider Service Actions implementation for using in Projects.

```
public class XTestServiceProviderController :
XBaseServiceProviderController<XTest, XTestDto, Guid, XTestDtoHub>,
IXBaseServiceProviderController<XTest, XTestDto, Guid, XTestDtoHub>
{
    public XTestServiceProviderController(
        ILogger<XTestServiceProviderController> logger,
        XAppConfiguration appConfiguration,
        IXIdentityProvider identityProvider,
        XValidationProvider validationProvider,
        IXTestServiceProvider provider,
        Func<IQueryable<XTest>, IOrderedQueryable<XTest>> defaultOrderBuilder,
        Func<IQueryable<XTest>, IIncludableQueryable<XTest, object>>
defaultIncludeBuilder
    ) : base(
        logger,
        appConfiguration,
        identityProvider,
        validationProvider,
        provider,
        defaultOrderBuilder,
        defaultIncludeBuilder
    )
    { }
}
```

Register PushService

after preparation of all requirements, final steps is Register DataService.there are two main step:

- **DI Registration:** in this phase, all requirements for Enabling Push Notifications Registered in DI Container.
- **Middleware Usage:** in this phase, all registered Hubs enabled using it's Provided routes and then accessible through their Paths.

Configure Service

this service Configured using this Configuration Model:

```

public partial class XPushServiceConfiguration {
    /// <summary>
    /// Base route of WebSocket Server ...
    /// </summary>
    /// <value>string</value>
    public string BaseRoute { get; set; }

    /// <summary>
    /// determines log level of signalR ...
    /// </summary>
    /// <value></value>
    public object ConnectionLogLevel { get; set; }

    /// <summary>
    /// when client disconnected it's automatically try to reconnect, this
    /// determines max number of try to connect ...
    /// </summary>
    /// <value></value>
    public int PushConnectionMaxRetry { get; set; }

    /// <summary>
    /// add support for message protocol ...
    /// </summary>
    /// <value></value>
    public bool AddSupportMessageProtocol { get; set; }

    /// <summary>
    /// the delay between two connection try ..
    /// </summary>
    /// <value></value>
    public int PushConnectionReconnectDelay { get; set; }
}

```

you can directly instanced this class and fill it and use it in Service Registration phase for Configuring Service. or Configure Service using App Setting Filling like this:

```

...
"PushServiceConfiguration": {
    "BaseRoute": "hubs",
    "AddSupportMessageProtocol": true
},
...

```

for Register Service and see different Methods:

```

...
/// <summary>
/// Register Module Provided Service on DI

```

```

/// </summary>
/// <param name="services"></param>
/// <param name="config"></param>
public static void AddXPushService(
    this IServiceCollection services,
    IConfiguration config,
    ServiceLifetime lifeTime = ServiceLifetime.Scoped
);

/// <summary>
/// Register Module Provided Service on DI
/// </summary>
/// <param name="services"></param>
/// <param name="config"></param>
public static void AddXPushService(
    this IServiceCollection services,
    XPushServiceConfiguration config,
    ServiceLifetime lifeTime = ServiceLifetime.Scoped
);
...

```

and for enable Hubs Middle wares you Have to Introduce your Hubs through an Instance of xPushServiceHelper class and provides it to Middleware Registrations:

```

/// <summary>
/// Use Module Middlewares on Application Builder
/// </summary>
/// <param name="app"></param>
/// <param name="helper"></param>
public static void UseXPushService(
    this IApplicationBuilder app,
    XPushServiceHelper helper
);

```

full Service Registration:

```

public class Startup
{
    //
    public IConfiguration Configuration { get; }

    public Startup(IConfiguration configuration)
    {
        ...
        Configuration = configuration;
        ...
    }

    public void ConfigureServices(IServiceCollection services)

```

```
{
    ...
    //
    // Push Service ...
    services.AddXPushService(
        config: configuration,
        lifeTime: ServiceLifetime.Scoped
    );

    //
    // Register Provider Services ...
    // since Hubs required Identity Provider, you Have to Register
    IdentityService as Requirements ...
    // a Provider Service, Provides Repository (Entity) or Service (Dto) based
    // Data Manipulation Actions by Supporting Hub Contexts Events ...
    services.AddScoped<IXTestServiceProvider, XTestServiceProvider>();
    services.AddScoped<IXTestRepositoryProvider, XTestRepositoryProvider>();
    ...
}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    ...
    //
    // Instance Service Helper ...
    var pushHelper = new XPushServiceHelper();

    //
    // Introduce Hubs and their Child paths ...
    pushHelper.AddHub<XTestDtoHub>("testDto");
    pushHelper.AddHub<XTestEntityHub>("testEntity");

    //
    // Use Push Service Middlewares ...
    app.UseXPushService(pushHelper);
    ...
}
}
```

Maintainer

Hadi Khazaei asl

<https://www.saherelm.ir>

hadi_khazaei_asl@yahoo.com