



What's new with ABP 8.0?



Engincan VESKE

Software Engineer at Volosoft



Overall

<https://blog.abp.io/abp/announcing-abp-8-0-release-candidate>

- Upgrading to .NET 8.0
- Upgrading to Angular 17
- Dynamic Claims
- Bundling & Minification: CDN Support
- Read-Only Repositories
- New Features for Account Module
- Other News...
- What's New with ABP Commercial 8.0?

Upgrading to .NET 8.0

ABP v8.0 RC.1 has been shipped based on **.NET 8.0**. Therefore, if you are going to upgrade your existing projects to **.NET 8.0**, you need to make some changes to your project.

For example, you need to change the **Target Framework**, and update Microsoft's package versions and so on. For that purpose, you can check the [Microsoft's Migrate from ASP.NET Core 7.0 to 8.0 documentation](#).

Migration Guides

- [ABP Framework 7.x to 8.0](#)
- [ABP Commercial 7.x to 8.0](#)

Upgrading to Angular 17

Angular 17 [has been released on November 8](#) and ABP Framework & ABP Commercial startup templates are immediately migrated to **Angular 17!**

So, when you create a new solution with the Angular UI, you will take advantage of the new Angular with the new cutting-edge features and enhancements right from the start!

Dynamic Claims

The **Dynamic Claims** feature is used to dynamically generate claims for the user in each request. This was needed because claims-based authentication is using in the ASP.NET Core, and because it stores the claims in the cookie or token, and they are being static, they don't change until a next re-login.

In other words, **Dynamic Claims** allows you to **get the latest user claims** without need to re-login in each time.

It's already been documented: <https://docs.abp.io/en/abp/8.0/Dynamic-Claims>

Dynamic Claims

Configure the **AbpClaimsPrincipalFactoryOptions** and set the **IsDynamicClaimsEnabled** option as true 👉

```
public override void ConfigureServices(ServiceConfigurationContext context)
{
    context.Services.Configure<AbpClaimsPrincipalFactoryOptions>(options =>
    {
        options.IsDynamicClaimsEnabled = true;
    });
}
```

Add the **DynamicClaims** middleware into the request pipeline 👉

```
public override void OnApplicationInitialization(ApplicationInitializationContext context)
{
    // Add this line before UseAuthorization.
    app.UseDynamicClaims();
    app.UseAuthorization();
    //...
}
```

Bundling & Minification: CDN Support

```
Configure<AbpBundlingOptions>(options =>
{
    options.StyleBundles
        .Add( bundleName: "MyStyleBundles", configureAction: config: BundleConfiguration =>
            {
                config
                    .AddFiles("/styles/my-style-1.css")
                    .AddFiles("/styles/my-style-2.css")
                    .AddFiles("https://cdn.abp.io/bootstrap.css");
            });

    options.ScriptBundles
        .Add( bundleName: "MyScriptBundles", configureAction: config: BundleConfiguration =>
            {
                config
                    .AddFiles("/scripts/my-script-1.js")
                    .AddFiles("/scripts/my-script-2.js")
                    .AddFiles("https://cdn.abp.io/bootstrap.js");
            });
});
```

In this version on, ABP Framework's [Bundling System](#) provides **CDN Support** for bundling.

The bundling system automatically recognizes the **external/CDN files** and places them as script tags into the page along with the bundled inline CSS/JS files.

```
<link rel="stylesheet" href="/__bundles/MyStyleBundles.EA8C28419DCA43363E9670973D4C0D15.css?v=638331889644609730" />
<link rel="stylesheet" href="https://cdn.abp.io/bootstrap.css" />

<script src="/__bundles/MyScriptBundles.C993366DF8840E08228F3EE685CB08E8.js?v=638331889644937120"></script>
<script src="https://cdn.abp.io/bootstrap.js"></script>
```


Read-Only Repositories (AsNoTracking())

```
public class MyService
{
    private readonly IRepository<Book, Guid> _bookRepository;
    private readonly IReadOnlyRepository<Book, Guid> _bookReadOnlyRepository;

    public async Task MyMethod()
    {
        //ReadOnly Repository -> GetListAsync(), GetAsync() vs. read-only methods
        //not change tracking involved
        var books:List<Book> = await _bookReadOnlyRepository.GetListAsync();

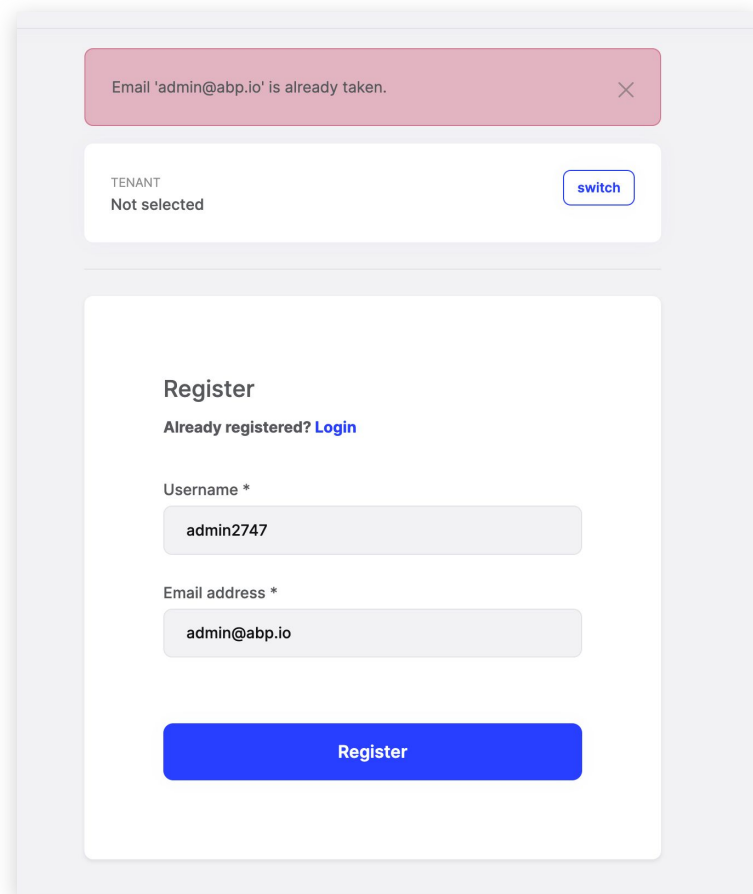
        //Repository -> All methods including InsertAsync(), UpdateAsync() etc.
        var newBook = await _bookRepository.InsertAsync(entity: new Book());
        var newBook2 = await _bookReadOnlyRepository.InsertAsync(new Book()); //not exists
    }
}
```

In this version, ABP Framework provides read-only repository interfaces (`IReadOnlyRepository<T>` or `IReadOnlyBasicRepository<T>`) to explicitly indicate that your purpose is to query data, but not change it.

Entity Framework Core read-only repository implementation uses [EF Core's No-Tracking](#) feature and that means the entities returned from the repository will not be tracked by the EF Core change tracker and this makes significant performance improvements.

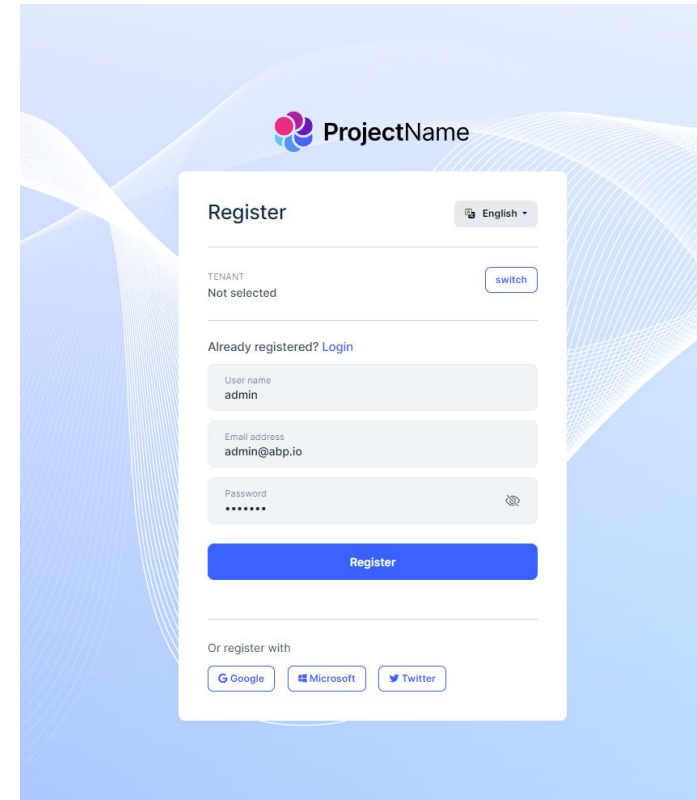
New Features for Account Module

Allowing to set username on social registration



Mockup of the Register form showing a validation error for email. The form includes a "TENANT" dropdown set to "Not selected" with a "switch" button. The "Register" section has a "Username *" field with "admin2747", an "Email address *" field with "admin@abp.io", and a "Register" button. A red error message at the top states: "Email 'admin@abp.io' is already taken."

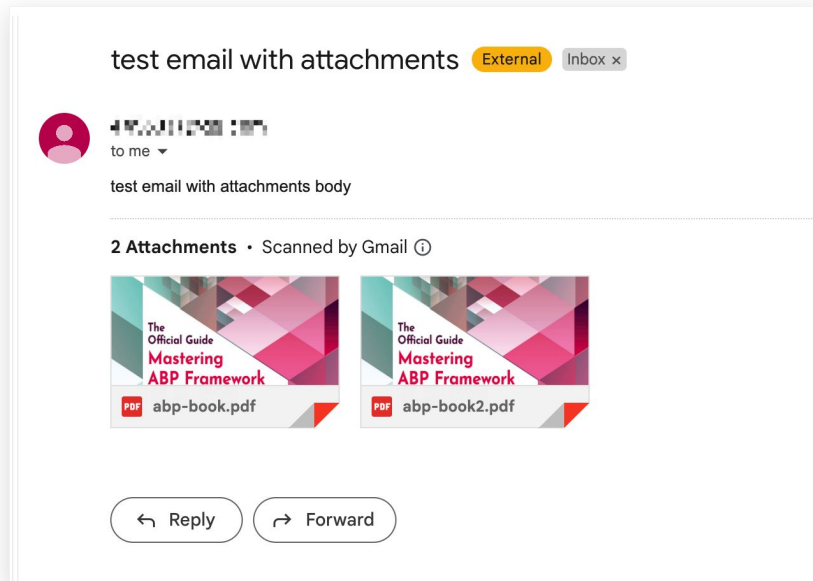
Adding "register with" option (social registration)



Mockup of the Register form with social registration options. The form includes a "TENANT" dropdown set to "Not selected" with a "switch" button. The "Register" section has fields for "User name" (admin), "Email address" (admin@abp.io), and "Password" (masked with dots). Below these is a "Register" button. At the bottom, there is a section "Or register with" with buttons for "Google", "Microsoft", and "Twitter". The form is titled "Register" and has a language selector set to "English".

Other News

- **LDAP over SSL (LDAPS)** setting has been added and recommended to establish a secure connection.
- **Object Mapping Enhancements** (supports mapping collection of objects for custom object mappers)
- **Email Sending Enhancements** (allowing sending attachments with **ISender.QueueAsync()** method)



Once you implement `IObjectMapper<User, UserDto>`, ABP can automatically convert a collection of `User` objects to a collection of `UserDto` objects. The following generic collection types are supported:

- `IEnumerable<T>`
- `ICollection<T>`
- `Collection<T>`
- `IList<T>`
- `List<T>`
- `T[]` (array)

What's New with ABP Commercial 8.0?

- Try to get profile picture from social/external logins
- Switch from Ocelot to YARP for the API Gateway (Microservice Solution Template)
- Password complexity indicators for MVC & Blazor UIs
- Read-only view for Identity/Users page
- Export & Import Users as Excel/CSV
- Suite: Generating Master/Detail Forms

Suite: Generating Master/Detail Forms

ABP Suite allows you to create a master-detail relationship with a few clicks. It generates the necessary code for the master and detail tables, including the foreign key relationship between the two tables.

The screenshot displays the 'Orders' management interface in the ABP Suite. The interface is divided into two main sections: the 'Master / Parent Grid' and the 'Child / Detail Grid'.

Master / Parent Grid: This table lists individual orders. It has columns for 'ACTIONS', 'NAME', and 'CODE'. Two orders are visible: 'ORD-2' with code '10644' and 'ORD-1' with code '10643'. The 'ORD-1' row is highlighted in blue.

Child / Detail Grid: This table shows the 'Order Lines' for the selected order (ORD-1). It has columns for 'ACTIONS', 'QUANTITY', 'TOTAL PRICE', and 'PRODUCT'. Five order lines are listed, each with a quantity and a total price, and a link to the product details.

ACTIONS	NAME	CODE
Actions	ORD-2	10644
Actions	ORD-1	10643

ACTIONS	QUANTITY	TOTAL PRICE	PRODUCT
Actions	2	\$ 28.59	Book - Social Animal: A Story of How Success Happens
Actions	1	\$ 21.09	Book - Mastering ABP Framework: Build maintainable .NET solutions by implementing software development best practices
Actions	1	\$ 44.99	Book - Domain-Driven Design: Tackling Complexity in the Heart of Software
Actions	4	\$ 120.00	Book - Clean Code: A Handbook of Agile Software Craftsmanship
Actions	2	\$ 40.20	Book - Refactoring: Improving the Design of Existing Code



THANKS FOR WATCHING

