Skoruba IdentityServer4 Admin Documentation

Release dev-doc

Jan 13, 2019

Introduction

The administration for the IdentityServer4 and Asp.Net Core Identity.

CHAPTER 1

Overview

1.1 Solution structure

STS

`Skoruba.IdentityServer4.STS.Identity` Quickstart UI for the IdentityServer4 with Asp.Net Core Identity and EF Core storage

Admin UI

`Skoruba.IdentityServer4.Admin` ASP.NET Core MVC application that contains Admin UI

`Skoruba.IdentityServer4.Admin.BusinessLogic` project that contains Dtos, Repositories, Services and Mappers for the IdentityServer4

`Skoruba.IdentityServer4.Admin.BusinessLogic.Identity` project that contains Dtos, Repositories, Services and Mappers for the Asp.Net Core Identity

`Skoruba.IdentityServer4.Admin.BusinessLogic.Shared` project that contains shared Dtos and ExceptionHandling for the Business Logic layer of the IdentityServer4 and Asp.Net Core Identity

`Skoruba.IdentityServer4.Admin.EntityFramework` EF Core data layer that contains Entities for the IdentityServer4

`Skoruba.IdentityServer4.Admin.EntityFramework.Identity` EF Core data layer that contains Entities for the Asp.Net Core Identity

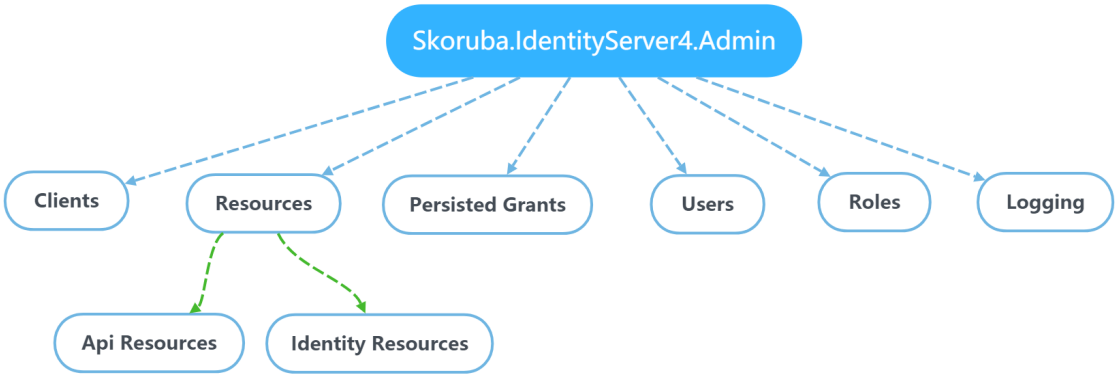
`Skoruba.IdentityServer4.Admin.EntityFramework.DbContexts` project that contains AdminDbContext for the administration

Tests

`Skoruba.IdentityServer4.Admin.IntegrationTests` xUnit project that contains the integration tests

`Skoruba.IdentityServer4.Admin.UnitTests` xUnit project that contains the unit tests

The admininistration contains the following sections



1.2 IdentityServer4

Clients

It is possible to define the configuration according the client type - by default the client types are used:

- Empty
- Web Application - Server side - Implicit flow
- Web Application - Server side - Hybrid flow
- Single Page Application - Javascript - Implicit flow
- Native Application - Mobile/Desktop - Hybrid flow
- Machine/Robot - Resource Owner Password and Client Credentials flow
- TV and Limited-Input Device Application - Device flow
- Actions: Add, Update, Clone, Remove
- Entities: - Client Cors Origins - Client Grant Types - Client IdP Restrictions - Client Post Logout Redirect Uris - Client Properties - Client Redirect Uris - Client Scopes - Client Secrets

API Resources

- Actions: Add, Update, Remove
- Entities: - Api Claims - Api Scopes - Api Scope Claims - Api Secrets

Identity Resources

- Actions: Add, Update, Remove
- Entities: - Identity Claims

1.3 Asp.Net Core Identity

Users

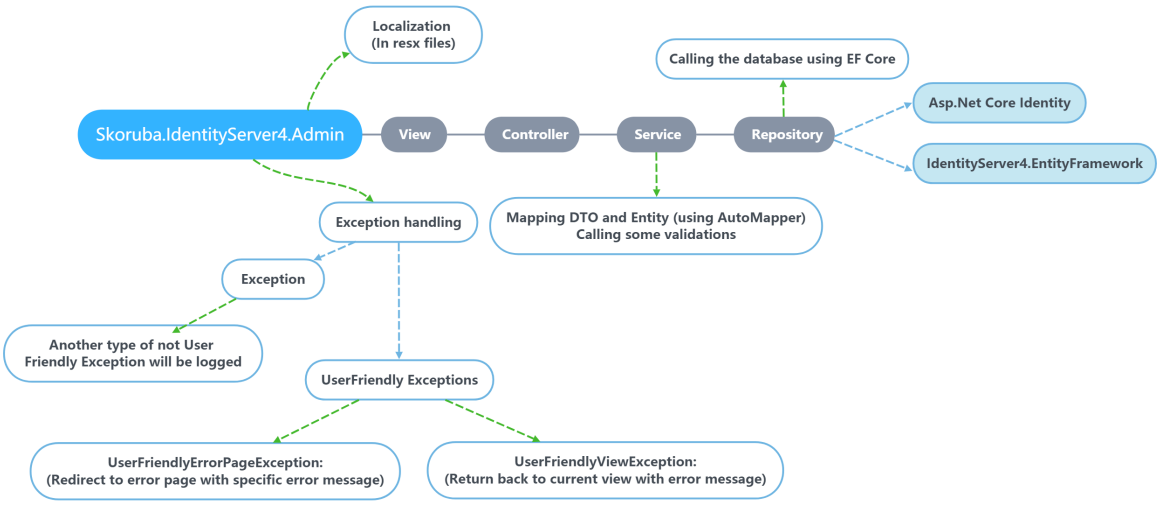
- Actions: Add, Update, Delete

- Entities: - User Roles - User Logins - User Claims

Roles

- Actions: Add, Update, Delete
- Entities: - Role Claims

1.4 Application Diagram



1.5 Template uses following list of nuget packages

- Available nuget packages

1.6 Authentication and Authorization

- Change the specific URLs and names for the IdentityServer and Authentication settings in `Constants/AuthenticationConsts` or *appsettings.json*
- `Constants/AuthorizationConsts.cs` contains configuration of constants connected with authorization - definition of the default name of admin policy
- In the controllers is used the policy which name is stored in - `AuthorizationConsts.AdministrationPolicy`. In the policy - *AuthorizationConsts.AdministrationPolicy* is defined required role stored in - *AuthorizationConsts.AdministrationRole*.
- With the default configuration, it is necessary to configure and run instance of IdentityServer4. It is possible to use initial migration for creating the client as it mentioned above

1.7 Localizations - labels, messages

- All labels and messages are stored in the resources `.resx` - locatated in `/Resources`
  - Client label descriptions from - http://docs.identityserver.io/en/release/reference/client.html
  - Api Resource label descriptions from - http://docs.identityserver.io/en/release/reference/api_resource.html
  - Identity Resource label descriptions from - http://docs.identityserver.io/en/release/reference/identity_resource.html

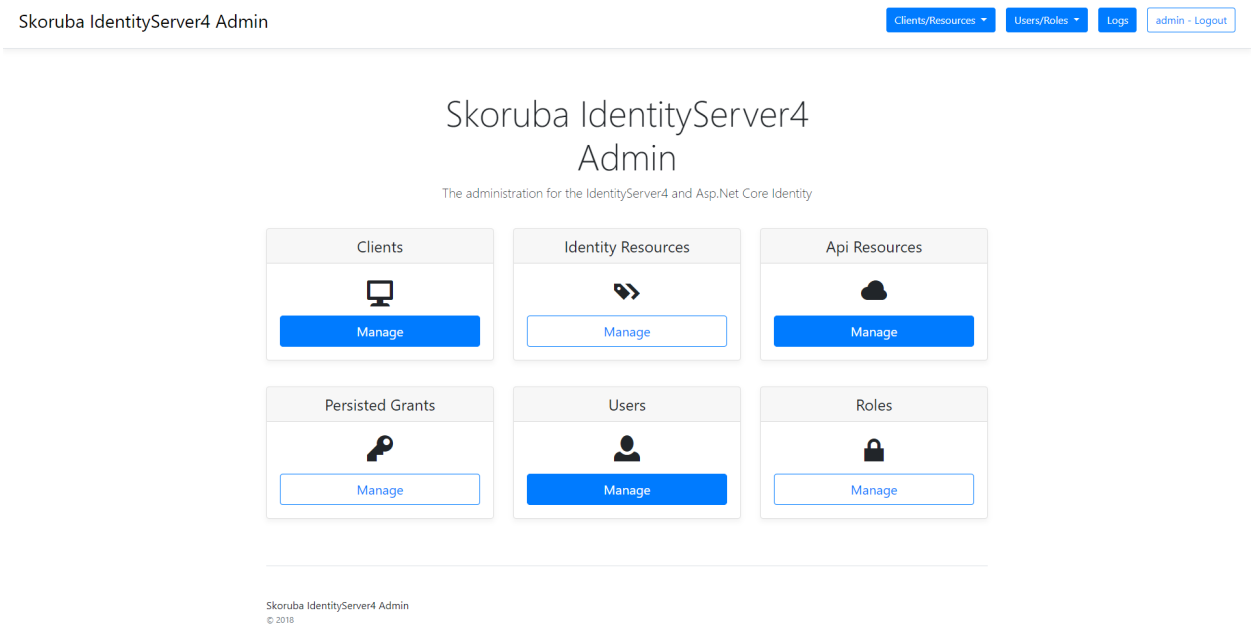
1.8 Tests

- The solution contains unit and integration tests.
- Stage environment is used for integration tests - `DbContext` contains setup for InMemory database - `Authentication` is setup for `CookieAuthentication` - with fake login url only for testing purpose - `AuthenticatedTestRequestMiddleware` - middleware for testing of authentication.
- If you want to use `Stage environment` for deploying - it is necessary to change these settings in `StartupHelpers.cs`.

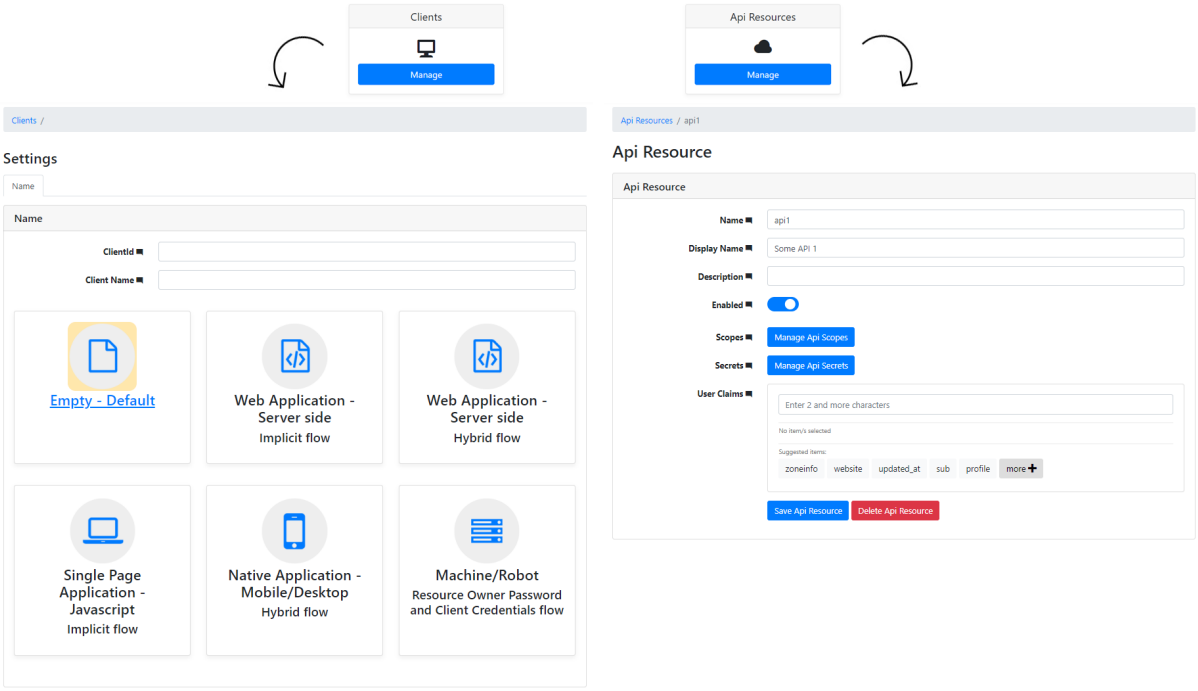
CHAPTER 2

Administration UI preview

- This administration uses bootstrap 4



- Forms


Clients
Manage
Api Resources
Manage
Clients /
Settings
Name
Name
ClientId
Client Name
Empty - Default
Web Application - Server side
Implicit flow
Web Application - Server side
Hybrid flow
Single Page Application - Javascript
Implicit flow
Native Application - Mobile/Desktop
Hybrid flow
Machine/Robot
Resource Owner Password and Client Credentials flow
Api Resources / api1
Api Resource
Api Resource
Name
api1
Display Name
Some API 1
Description
Enabled
Scopes
Manage Api Scopes
Secrets
Manage Api Secrets
User Claims
Enter 2 and more characters
No item/s selected
Suggested items:
zoneinfo
website
updated_at
sub
profile
more
Save Api Resource
Delete Api Resource

CHAPTER 3

Installation

3.1 Requirements

- Install the latest .NET Core 2.x SDK (using older versions may lead to 502.5 errors when hosted on IIS or application exiting immediately after starting when self-hosted)

3.2 Installation methods

Cloning

```
git clone https://github.com/skoruba/IdentityServer4.Admin
```

Installation via dotnet new template

- Install the dotnet new template:

:: dotnet new -i Skoruba.IdentityServer4.Admin.Templates::1.0.0-beta5-update1

- Create new project:

:: dotnet new skoruba.is4admin –name MyProject –title MyProject –adminrole MyRole –adminclientid MyClientId

Project template options:

```
--name: [string value] for project name
--title: [string value] for title and footer of the administration in UI
--adminrole: [string value] for name of admin role, that is used to authorize the␣
↪administration
--adminclientid: [string value] for client name, that is used in the IdentityServer4␣
↪configuration
```

3.3 Installation of the Client Libraries

:: cd src/Skoruba.IdentityServer4.Admin npm install

cd src/Skoruba.IdentityServer4.STS.Identity npm install

3.4 Running in Visual Studio

- Set Startup projects: - Skoruba.IdentityServer4.Admin - Skoruba.IdentityServer4.STS.Identity

CHAPTER 4

Migration and seeding

4.1 EF Core & Data Access

- Run entity framework migrations - for instance from Visual Studio command line (Nuget package manager):

:: Add-Migration DbInit -context AdminDbContext -output Data/Migrations Update-Database -context AdminDbContext

- Or via `dotnet CLI`:

:: dotnet ef migrations add DbInit -c AdminDbContext -o Data/Migrations dotnet ef database update -c AdminDbContext

Migrations are not a part of the repository - they are ignored in `.gitignore`.

We suggest to use seed data:

- In `Program.cs` -> `Main`, uncomment `DbMigrationHelpers.EnsureSeedData(host)` or use dotnet CLI `dotnet run /seed`
- The `Clients` and `Resources` files in `Configuration/IdentityServer` are the initial data, based on a sample from IdentityServer4
- The `Users` file in `Configuration/Identity` contains the default admin username and password for the first login

CHAPTER 5

Changing database engine

5.1 Create the migration of database

:: Add-Migration Initial -context AdminDbContext -output Data/Migrations Update-Database -context AdminDbContext

5.2 Using other database engines

PostgreSQL

Install following NuGet package:

:: Npgsql.EntityFrameworkCore.PostgreSQL.Design

Find `RegisterDbContexts` function in `Helpers\StartupHelpers.cs`

:: services.AddDbContext<AdminDbContext>(options => options.UseSqlServer(configuration.GetConnectionString(ConfigurationCon
optionsSql => optionsSql.MigrationsAssembly(migrationsAssembly)));

and change `UseSqlServer` to `UseNpgsql`.

Don't forget to update your connection string in appsettings.json and (re)generate migrations for new database

SQLite

Install following NuGet package:

:: Microsoft.EntityFrameworkCore.Sqlite.Design

Find `RegisterDbContexts` function in `Helpers\StartupHelpers.cs`

:: services.AddDbContext<AdminDbContext>(options => options.UseSqlServer(configuration.GetConnectionString(ConfigurationCon
optionsSql => optionsSql.MigrationsAssembly(migrationsAssembly)));

and change `UseSqlServer` to `UseSqlite`.

Note: Don't forget to update your connection string in appsettings.json and (re)generate migrations for new database

MySQL and MariaDB

Install the following NuGet package:

:: Pomelo.EntityFrameworkCore.MySql

Find `RegisterDbContexts` function in `Helpers\StartupHelpers.cs`

:: services.AddDbContext<AdminDbContext>(options => options.UseSqlServer(configuration.GetConnectionString(ConfigurationCon
optionsSql => optionsSql.MigrationsAssembly(migrationsAssembly)));

and change `UseSqlServer` to `UseMySql`.

Find `Properties` in `Skoruba.IdentityServer4.Admin.EntityFramework\Entities\Log.cs`

:: [Column(TypeName = "xml")] public string Properties { get; set; }

and remove the `[Column]` attribute. As MySQL and MariaDB don't know about a XML data type.

Note: Don't forget to update your connection string in appsettings.json and (re)generate migrations for new database

CHAPTER 6

How to use existing IdentityServer4 instance

- You can use one or more `DbContexts` for the administration.
- The configuration of DbContexts is in the `Startup.cs`:

1. Single DbContext:

:: services.AddAdminServices<AdminDbContext>();

`AddAdminServices` expects one generic param:

- `TAdminDbContext` - It requires to implement interfaces `IAdminConfigurationDbContext`, `IAdminPersistedGrantDbContext`, `IAdminLogDbContext`

2. Multiple DbContexts:

- It is possible to overload this method:

:: services.AddAdminServices<TConfigurationDbContext, TPersistedGrantDbContext, TLogDbContext>();

`AddAdminServices` expects following generic params:

- `TConfigurationDbContext` - DbContext for the configuration data - It requires to implement interface `IAdminConfigurationDbContext`
- `TPersistedGrantDbContext` - DbContext for the operational data - It requires to implement interface `IAdminPersistedGrantDbContext`
- `TLogDbContext` - for the logs - DbContext for the operational data - It requires to implement interface `IAdminLogDbContext`

CHAPTER 7

Using with existing ASP.NET Identity deployment

7.1 How to configure Asp.Net Core Identity - database, primary key data type

By default, it's used as the primary key `int`, but it's possible to change it:

How to configure DbContext for ASP.NET Core Identity

- You can setup the `DbContext` - in `Startup.cs`:

::

services.AddAdminAspNetIdentityServices<AdminDbContext, UserDto<int>, int, RoleDto<int>, int, int, int, UserIdentity, UserIdentityRole, int, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken>();

- Method `AddAdminAspNetIdentityServices` expects the generic param `TAdminDbContext` that inherits from `IdentityDbContext` and implements interface `IAdminPersistedGrantIdentityDbContext` (for operation data connected with Asp.Net Core Identity)

How to configure Identity primary key data type in ASP.NET Core Identity

- By default, it's used `int` as the primary key, but you can change to `Guid` or `string`.

How to use for example ‘‘Guid‘‘

1. Change `int` to `Guid` in `Startup.cs`:

Original:

::

services.AddAdminAspNetIdentityServices<AdminDbContext, UserDto<int>, int, RoleDto<int>, int, int, int, UserIdentity, UserIdentityRole, int, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken>();

New:

::

services.AddAdminAspNetIdentityServices<AdminDbContext, UserDto<Guid>, Guid, RoleDto<Guid>, Guid, Guid, Guid
UserIdentity, UserIdentityRole, Guid, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken>();

2. Change `int` to `Guid` in all files in folder - `Skoruba.IdentityServer4.Admin.EntityFramework/Entities/Identity`:

For example - `UserIdentity.cs`:

Original:

:: public class UserIdentity : IdentityUser<int> {

}

New:

:: public class UserIdentity : IdentityUser<Guid> {

}

- Change `int` to `Guid` in other files in this folder - `Skoruba.IdentityServer4.Admin.EntityFramework/Entities/Identity`

3. Change `int` to `Guid` in all files in folder - `Skoruba.IdentityServer4.Admin/Views/Identity`:

For example - `Role.cshtml`:

Original:

:: @model Skoruba.IdentityServer4.Admin.BusinessLogic.Dtos.Identity.RoleDto<int> // ... @if (!EqualityComparer<int>.Default.Equals(Model.Id, default(int)))

New:

:: @model Skoruba.IdentityServer4.Admin.BusinessLogic.Dtos.Identity.RoleDto<Guid> // ... @if (!EqualityComparer<Guid>.Default.Equals(Model.Id, default(Guid)))

- Change `int` to `Guid` in other files in this folder - `Skoruba.IdentityServer4.Admin/Views/Identity`

4. Change `int` to `Guid` in `AdminDbContext` - `Skoruba.IdentityServer4.Admin.EntityFramework/DbContexts`:

Original:

::

public class AdminDbContext [IdentityDbContext<UserIdentity, UserIdentityRole, int, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken>,] IAdminConfigurationDbContext, IAdminLogDbContext, IAdminPersistedGrantIdentityDbContext

New:

::

public class AdminDbContext [IdentityDbContext<UserIdentity, UserIdentityRole, Guid, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken>,] IAdminConfigurationDbContext, IAdminLogDbContext, IAdminPersistedGrantIdentityDbContext

5. Change `int` to `Guid` in `GrantController` - `Skoruba.IdentityServer4.Admin/Controllers`:

Original:

:: public class GrantController : BaseController {

private readonly IPersistedGrantService<AdminDbContext, UserIdentity, UserIdentityRole, int, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken> _persistedGrantService; private readonly IStringLocalizer<GrantController> _localizer;

public GrantController(IPersistedGrantService<AdminDbContext, UserIdentity, UserIdentityRole, int, UserIdentity
ILogger<ConfigurationController> logger, IStringLocalizer<GrantController> localizer) : base(logger)

{ _persistedGrantService = persistedGrantService; _localizer = localizer;

}

}

New:

:: public class GrantController : BaseController {

private readonly IPersistedGrantService<AdminDbContext, UserIdentity, UserIdentityRole, Guid, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken> _persistedGrantService; private readonly IStringLocalizer<GrantController> _localizer;

public GrantController(IPersistedGrantService<AdminDbContext, UserIdentity, UserIdentityRole, Guid, UserIdent
ILogger<ConfigurationController> logger, IStringLocalizer<GrantController> localizer) : base(logger)

{ _persistedGrantService = persistedGrantService; _localizer = localizer;

}

}

6. Change `int` to `Guid` in `IdentityController - Skoruba.IdentityServer4.Admin/Controllers`:

Original:

:: public class IdentityController : BaseIdentityController<AdminDbContext, UserDto<int>, int, RoleDto<int>, int, int, int, UserIdentity, UserIdentityRole, int, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken> {

public IdentityController(IIdentityService<AdminDbContext, UserDto<int>, int, RoleDto<int>, int, int, int, UserIde
: base(identityService, logger, localizer)

}

New:

:: public class IdentityController : BaseIdentityController<AdminDbContext, UserDto<Guid>, Guid, RoleDto<Guid>, Guid, Guid, Guid, UserIdentity, UserIdentityRole, Guid, UserIdentityUserClaim, UserIdentityUserRole, UserIdentityUserLogin, UserIdentityRoleClaim, UserIdentityUserToken> {

public IdentityController(IIdentityService<AdminDbContext, UserDto<Guid>, Guid, RoleDto<Guid>, Guid, Guid,
: base(identityService, logger, localizer)

}

CHAPTER 8

PostgreSQL on Ubuntu tutorial

8.1 Introduction

Tutorial covers configuration of Admin on Ubuntu 18.04 with fresh instance of PostgreSQL database.

8.2 Prerequisites

.NET Core 2.2 SDK

Instructions from: https://dotnet.microsoft.com/download/linux-package-manager/ubuntu18-04/sdk-2.2.101

:: wget -q https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb sudo dpkg -i packages-microsoft-prod.deb

sudo add-apt-repository universe sudo apt-get install apt-transport-https sudo apt-get update sudo apt-get install dotnet-sdk-2.2

PostgreSQ

Instructions from: https://linuxize.com/post/how-to-install-postgresql-on-ubuntu-18-04/

:: sudo apt update sudo apt install postgresql postgresql-contrib

Throughout tutorial we will use PostgreSQL running on localhost and default port 5432 with username/password combination postgres/postgres. You can update connection strings with your own or if you want to follow everything exactly then after installing PostgreSQL you will need to change password of *postgres* user:

:: sudo -u postgres psql ALTER USER postgre WITH PASSWORD 'postgres';

IDE

Console commands will be used throghout the tutorial but for code editing it is recommended to use dedicated IDE. I've got good experience with [Visual Studio Code](https://code.visualstudio.com/) and [Rider](https://www.jetbrains.com/rider/).

8.3 Cloning Admin

:: git clone https://github.com/skoruba/IdentityServer4.Admin

8.4 Adjustments for PostgreSQL

By default everything is configured for Microsoft SQL Server, but fortunately it's pretty easy to change.

Replace connection strings First change connection strings in `src/Skoruba.IdentityServer4.Admin/appsettings.json` and `src/Skoruba.IdentityServer4.STS.Identity/appsettings.json` and replace them with following connection string:

:: Server=localhost; User Id=postgres; Database=is4admin; Port=5432; Password=postgres; SSL Mode=Prefer; Trust Server Certificate=true

Install required packages

Next we need to install PostgreSQL support for EntityFramework Core in Skoruba.IdentityServer4.Admin and Skoruba.IdentityServer4.STS.Identity in order to do that run in each project's directory:

:: dotnet add src/Skoruba.IdentityServer4.Admin package Npgsql.EntityFrameworkCore.PostgreSQL dotnet add src/Skoruba.IdentityServer4.Admin package Npgsql.EntityFrameworkCore.PostgreSQL.Design dotnet add src/Skoruba.IdentityServer4.STS.Identity package Npgsql.EntityFrameworkCore.PostgreSQL dotnet add src/Skoruba.IdentityServer4.STS.Identity package Npgsql.EntityFrameworkCore.PostgreSQL.Design

Replace UseSqlServer with UseNpgsql

In `src/Skoruba.IdentityServer4.Admin` and `src/Skoruba.IdentityServer4.STS.Identity` in `Helpers/StartupHelpers.cs` replace all occurences of `UseSqlServer` with `UseNpgsql`. This will inform EntityFramework that PostgreSQL will be used instead of SQL Server.

8.5 Final setup

**Generate initial migrations **

:: dotnet ef migrations add DbInit -c AdminDbContext -o Data/Migrations dotnet ef database update -c AdminDbContext

** Run STS and Admin**

First run STS in `src/Skoruba.IdentityServer4.STS.Identity` launch:

:: dotnet run

Admin also needs to seed the database so seperate terminal in `src/Skoruba.IdentityServer4.Admin` we add additional seed parameter:

:: dotnet run /seed

After that we should have STS listening on http://localhost:5000 and Admin on http://localhost:9000. We can go to the latter and we should be redirected to our STS for authentication (admin/Pa$$word123 is the default combination).

8.6 Final thoughts

There are many more steps required before IS4 and Admin panel are sufficiently hardened to be used in production scenario. Please bear in mind that this tutorial serves only as a quickstart.