2.6. Timing

2.6.1. Introduction

While some applications target a single timezone, some others target to many different timezones. To satisfy such needs and centralize datetime operations, StudioX provides common infrastructure for datetime operations.

2.6.2. Clock

Clock is the main class used to work with DateTime values. It defines the following static properties or methods:

- **Now**: Gets current time according to current provider.
- **Kind**: Gets DateTimeKind of current provider.
- **SupportsMultipleTimezone**: Gets a value indicates that current provider can be used for applications need to multiple timezones.
- **Normalize**: Normalizes/converts given DateTime upon current provider.

So, instead of using DateTime.Now, we are using Clock.Now, which abstracts it:

```
DateTime now = Clock.Now;
```

Clock uses clock providers inside it. There are three types of built-in clock providers:

- **ClockProviders.Unspecified** (UnspecifiedClockProvider): This is the default clock provider and behaives just like **DateTime.Now**. It acts as you don't use Clock class at all.
- **ClockProviders.Utc** (UtcClockProvider): Works in UTC datetime. DateTime.UtcNow for Clock.Now. Normalize method converts a given datetime to utc datetime and set it's kind to DateTimeKind.UTC. It **supports multiple timezones**.
- **ClockProviders.Local** (LocalClockProvider): Works in Local computer's time. Normalize method converts a given datetime to local datetime and set it's kind to DateTimeKind.Local.

You can set Clock.Provider in order to use a different clock provider:

```
Clock.Provider = ClockProviders.Utc;
```

This is generally done at the beginning of an application (proper to do it Application_Start in a web application).

Client Side

Clock can be used on the client side by studiox.clock object in javascript. When you set Clock.Provider on the server side, StudioX automatically sets value of studiox.clock.provider on the client side.

2.6.3. Time Zones

StudioX defines a setting named StudioX.Timing.TimeZone (TimingSettingNames.TimeZone constant) for storing selected timezone of host, tenant and user. StudioX assumes that value of timezone setting is a valid Windows timezone id. It also defines a timezone mapping file to convert a Windows Timezone to IANA timezone since some of the common libraries are using IANA timezone id. UtcClockProvider must be used in order to support multiple timezones. Because if UtcClockProvider is used, all datetime values will be stored in UTC and all datetimes will be sent to clients in UTC format. Then on the client side we can convert UTC datetime to user's timezone by using user's current timezone setting.

Client Side

StudioX creates an javascript object named studiox.timing.timeZoneInfo which contains timezone information for current user. This information contains Windows and IANA timezone ids and some extra information for windows timezone info. This information can be used to make client side datetime convertions and showing a datetime to user in his/her timezone.

2.6.4. Binders and Converters

- StudioX automatically normalizes DateTimes received from clients in MVC, Web API and ASP.NET Core applications, based on the current clock provider.
- StudioX automatically normalized DateTimes received from database based on the current clock provider, when EntityFramework modules used.

If UTC clock provider is used, then all DateTimes stored in database assumed as UTC values, and all DateTimes received from clients assumed as UTC unless it's explicitly specified.