Location Services

Introduction

- The location service gets its data from various sources such as GPS, Wi-Fi, and the cellular network.
- When developing a location aware application we should balance between the need for getting an accurate data and our wish to reduce the power consumption. The Wi-Fi and the cellular radio produce less accurate location information and use less power. The GPS receiver usually provide with a more precise location data and usually consume more power.

The `GeoCoordinateWatcher` Class

- We can use this class for getting location data during the execution of our application.

```
...
GeoCoordinateWatcher ob = new
    GeoCoordinateWatcher(GeoPositionAccuracy.Low);
…
```

- In order to use this class when developing your application make sure to add the `System.Device.Location` assembly reference to your project.

The GeoCoordinateWatcher

- The native code layer is responsible for evaluating the available location data sources and selecting the most suitable source based on the value we pass over when instantiating the `GeoCoordinateWatcher` class.
- When instantiating GeoCoordinateWatcher we should pass over one of two possible values:

```
GeoPositionAccuracy.Low
GeoPositionAccuracy.High
```

The `MovementTreshold` Property

- This property specifies the minimum change in position that must take place so that the `PositionChanged` event will be raised.
- We set this property with a value that specifies the change in metters. Once the specified change takes place a `PositionChanged` event is raised.

The `MovementTreshold` Property

- Microsoft recommends on setting the value of this property to be 20 metters at the minimum. According to Microsoft a smaller value might result in higher power consumption and in un accurate behavior.

...

```
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
...
```

The `PositionChanged` Event

- The `PositionChanged` event takes place each whenever the change is bigger than the `MovementTreshold` value.

...

```
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
locationWatcher.PositionChanged += MyLocationChangeMethod;
...
```

The `StatusChanged` Event

- When the status of the `GeoCoordinateWatcher` object changes (e.g. the GPS data isn't available) the StatusEvent takes place.

...

```
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
locationWatcher.PositionChanged += MyLocationChangeMethod;
locationWatcher.StatusChanged += MyLocationServiceStatusMethod;
...
```

The `Start` Method

- We start the `GeoCoordinateWatcher` by calling the asynchronous `Start()` method on it.

...

```
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
locationWatcher.PositionChanged += MyLocationChangeMethod;
locationWatcher.StatusChanged += MyLocationServiceStatusMethod;
locationWatcher.Start();
...
```

The `Permission` Property

- We can check the `Permission` in order to know whether the user chose to disable the location service.

```
...
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
if(locationManager.Permission==GeoPositionPermission.Denied)
{
    ...
}
...
```

Getting The Location Data

- The method we define to handle location data should be of the `EventHandler<GeoPositionStatusChangedEventArgs>` type.

```
...
void MyPositionChanged(GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    double latitude = e.Position.Location.Latitude;
    double longitude = e.Position.Location.Longitude;
}
...
```

the Location property is of the GeoCoordinate type

The `Stop` Method

- We stop the `GeoCoordinateWatcher` by calling the `Stop()` method on it.

```
...
locationWatcher.Stop();
...
```

- When there is no need in the location service we better call the `Stop()` method in order to maximize the battery life.

Location Services

Introduction

- The location service gets its data from various sources such as GPS, Wi-Fi, and the cellular network.
- When developing a location aware application we should balance between the need for getting an accurate data and our wish to reduce the power consumption. The Wi-Fi and the cellular radio produce less accurate location information and use less power. The GPS receiver usually provide with a more precise location data and usually consume more power.

The `GeoCoordinateWatcher` Class

- We can use this class for getting location data during the execution of our application.

```
...
GeoCoordinateWatcher ob = new
    GeoCoordinateWatcher(GeoPositionAccuracy.Low);
…
```

- In order to use this class when developing your application make sure to add the `System.Device.Location` assembly reference to your project.

The GeoCoordinateWatcher

- The native code layer is responsible for evaluating the available location data sources and selecting the most suitable source based on the value we pass over when instantiating the `GeoCoordinateWatcher` class.
- When instantiating GeoCoordinateWatcher we should pass over one of two possible values:

```
GeoPositionAccuracy.Low
GeoPositionAccuracy.High
```

The `MovementTreshold` Property

- This property specifies the minimum change in position that must take place so that the `PositionChanged` event will be raised.
- We set this property with a value that specifies the change in metters. Once the specified change takes place a `PositionChanged` event is raised.

The `MovementTreshold` Property

- Microsoft recommends on setting the value of this property to be 20 metters at the minimum. According to Microsoft a smaller value might result in higher power consumption and in un accurate behavior.

```
...
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
...
```

The `PositionChanged` Event

- The `PositionChanged` event takes place each whenever the change is bigger than the `MovementTreshold` value.

```
...
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
locationWatcher.PositionChanged += MyLocationChangeMethod;
...
```

The `StatusChanged` Event

- When the status of the `GeoCoordinateWatcher` object changes (e.g. the GPS data isn't available) the StatusEvent takes place.

```
...
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
locationWatcher.PositionChanged += MyLocationChangeMethod;
locationWatcher.StatusChanged += MyLocationServiceStatusMethod;
...
```

The `Start` Method

- We start the `GeoCoordinateWatcher` by calling the asynchronous `Start()` method on it.

```
...
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
locationWatcher.PositionChanged += MyLocationChangeMethod;
locationWatcher.StatusChanged += MyLocationServiceStatusMethod;
locationWatcher.Start();
...
```

The `Permission` Property

- We can check the `Permission` in order to know whether the user chose to disable the location service.

```
...
GeoCoordinateWatcher locationWatcher =
    new GeoCoordinateWatcher(GeoPositionAccuracy.Low);
locationWatcher.MovementThreshold = 20;
if(locationManager.Permission==GeoPositionPermission.Denied)
{
    ...
}
...
```

Getting The Location Data

- The method we define to handle location data should be of the `EventHandler<GeoPositionStatusChangedEventArgs>` type.

```
...
void MyPositionChanged(GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    double latitude = e.Position.Location.Latitude;
    double longitude = e.Position.Location.Longitude;
}
...
```

the Location property is of the GeoCoordinate type

The `Stop` Method

- We stop the `GeoCoordinateWatcher` by calling the `Stop()` method on it.

```
...
locationWatcher.Stop();
...
```

- When there is no need in the location service we better call the `Stop()` method in order to maximize the battery life.