Chapter 31: MessagingCenter

Xamarin.Forms has a built-in messaging mechanism to promote decoupled code. This way, view models and other components do not need to know each other. They can communicate by a simple messaging contract.

There a basically two main ingredients for using the `MessagingCenter`.

Subscribe; listen for messages with a certain signature (the contract) and execute code when a message is received. A message can have multiple subscribers.

Send; sending a message for subscribers to act upon.

Section 31.1: Simple example

Here we will see a simple example of using the MessagingCenter in Xamarin.Forms.

First, let's have a look at subscribing to a message. In the `FooMessaging` model we subscribe to a message coming from the `MainPage`. The message should be "Hi" and when we receive it, we register a handler which sets the property `Greeting`. Lastly `this` means the current `FooMessaging` instance is registering for this message.

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage> (this, "Hi", (sender) => {
            this.Greeting = "Hi there!";
        });
    }
}
```

To send a message triggering this functionality, we need to have a page called `MainPage`, and implement code like underneath.

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage> (this, "Hi");
    }
}
```

In our `MainPage` we have a button with a handler that sends a message. `this` should be an instance of `MainPage`.

Section 31.2: Passing arguments

You can also pass arguments with a message to work with.

We will use the classed from our previous example and extend them. In the receiving part, right behind the `Subscribe` method call add the type of the argument you are expecting. Also make sure you also declare the arguments in the handler signature.

```
public class FooMessaging
{
```

```
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage, string> (this, "Hi", (sender, arg) => {
            this.Greeting = arg;
        });
    }
}
```

When sending a message, make sure to include the argument value. Also, here you add the type right behind the `Send` method and add the argument value.

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage, string> (this, "Hi", "Hi there!");
    }
}
```

In this example a simple string is used, but you can also use any other type of (complex) objects.

Section 31.3: Unsubscribing

When you no longer need to receive messages, you can simply unsubscribe. You can do it like this:

```
MessagingCenter.Unsubscribe<MainPage> (this, "Hi");
```

When you are supplying arguments, you have to unsubscribe from the complete signature, like this:

```
MessagingCenter.Unsubscribe<MainPage, string> (this, "Hi");
```