Main types of development for DigitalEdge solution.

INTRODUCTION

This document presents and explains standard paths for developing new, or modifying existing, functionalities on DigitalEdge web application.
Intermediate knowledge about MVC 5 framework and toolset is required, and assumed by the document creator.

UI FLOW INTRODUCTION (DIGITAL EDGE PROCESS)

Ui Flows in DigitalEdge is a set of complete functionalities which enables the application to track stages of multistep flows which asks the user, through multiple states, to fill in the data needed for a single or multiple actions he requested to perform.

Most common flows are structured like this:

Enter data screen/s – user is asked to input all the data required for the action he is performing. User can be presented with multiple steps/stages/states if the data he is asked to enter is too numerous to fit on one screen, or there are multiple paths in the flow, which is dependent on the data he inputted.

Review entered data screen – this is step two of the flow, where the user can review the data he filled out, and check if it's alright. He can then go back and edit the data or continue to the next step. If additional authorization is requested for that action, **subprocess** of authorization will be triggered, where user will be asked to use another factor of authentication. **Subprocesses** are explained later in this document

Result/status and additional actions screen – User is informed about the competition of his action, and presented with additional options. Those options can be related to the action he just performed, like "repeat" or "add geolocation to transfer", or it can be related to his next point of navigation on the application, such as "go to homepage" or "view transfer list"

UI FLOW MODEL AND DEFINITION (DIGITAL EDGE PROCESS)

There are two main types of UI FLOWs in Digital Edge solution.

Coded UI Flow and Generic UI Flow. Classes used for each are named:

```
CodedUIProcess<T> : Process

GenericUIProcess : Process
```

Generic UI flow is using the same structure as Coded UI flow, but the process of development is different.

Generic UI flow is loaded from the database, whole definition is being injected into process object in memory and then executed as-is. Generic flow definition is created and edited through "Digital Edge Studio" tool which allows the flow creator to use visual elements, through drag&drop mechanism, to organize and define the whole process.

In this document, we will talk only about Coded UI flows, and the documentation about Generic Ui flows should be contained in a separate document.

Coded UI flow is self-explanatory. It is a whole process definition written in code.

The main building block of processes is its definition. How everything will look and work is defined in one file. Digital Edge solution has a whole infrastructure developed to support this definition.

As seen above, Coded UI Process derives from the class "Process" and this is where most of the process definition lies. Let's look at the structure of the "Process" class:

`UiContentPlacement.Placement`	`Property:Enum`	Where this process will be displayed. Processes can be placed Inline or in a modal popup (wide, narrow or side popup)
`string OnDoneUtility`	`Property: String`	Which "Utility" (JavaScript function) will be executed after the flow is done
`int ProcessId`	`Property:Int`	Process identifier
`string WorkItemId`	`Property:String`	Unique flow identfier
`string ContainerId`	`Property:String`	Id of the html element which will contain the process markup
`MessagesDisplay MessagesDisplay`	`Property:MessagesDisplay`	Used for setting the way of displaying messages, warnings during the UI Flow execution

`bool UsesFlowHistory`	`Property:Boolean`	UiFlow system has an automated way of remembering the history of state changes. Enabling this will enable history tracking so using “back button” is seamlessly easy to implement. This is not always possible, so the developer can decide whether or not to use this functionality
`string ShortName`	`Property:String`	Short name of the process, which will be translated on the UI. This should be populated with the localization key which has his localization strings defined.
`string Description`	`Property:String`	Description of the, which will be translated on the UI. This should be populated with the localization key which has his localization strings defined.
`RouteValues RouteValues`	`Property:RouteValues`	Whats the url of the process. Controller, Area and Action should be defined here
`Messages Messages`	`Property:Messages`	Object that contains all the messages that should be displayed to the user during the process
`List<State> ProcessStates`	`Property:State`	List of states objects. State is another complex object which defines one screen, one step of the process/flow
`List<Form> Forms`	`Property:Form`	List of forms. Every state has its form, which is used to display input fields to the user and bind it to the process object, more precisely, to the process workitem object, through the binding paths
`ProcessActivity OnInitActivity`	`Property:ProcessActivity`	Definition of the microsoft foundation activity (xaml) which will be executed on process initialization (when