Elements of the Uniface IDE: Editors and Worksheets

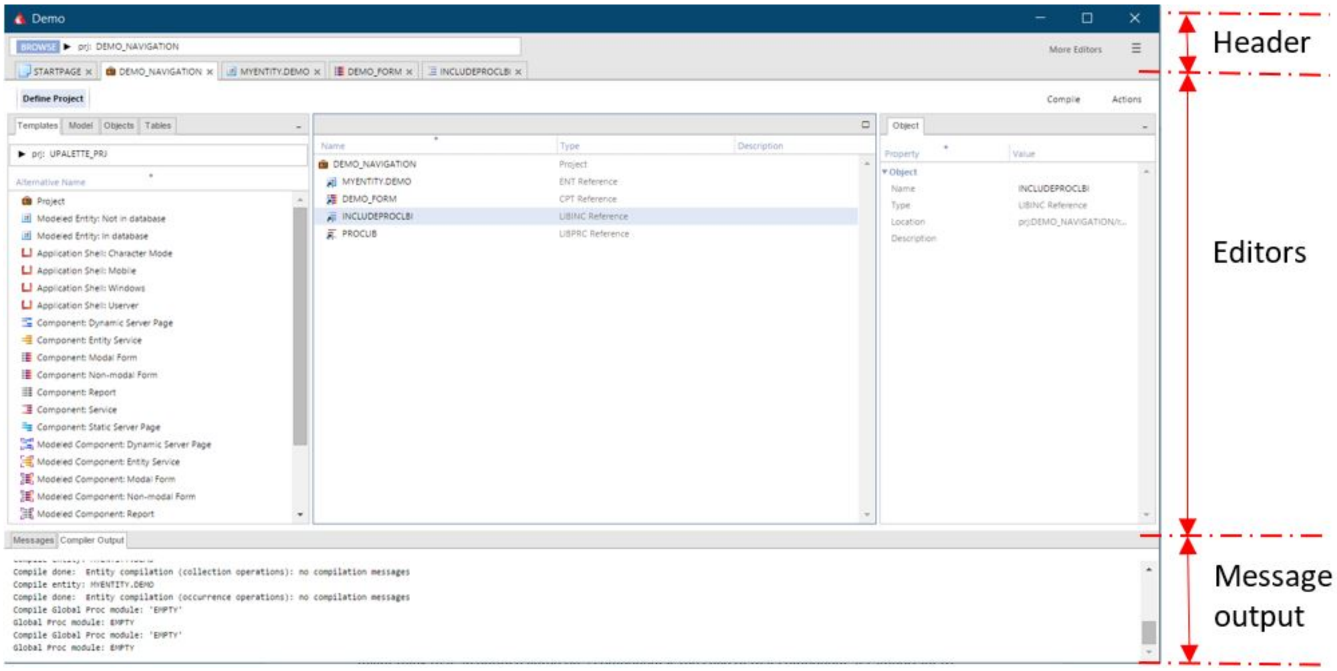
Blog by Peter Lammersma

What I like and expect in every well-designed system is predictability. Can I find what I am searching for in the expected location? In other words, is the interface intuitive and consistent?

That's one of the best things about Uniface 10. Having worked with several earlier versions of Uniface, I can confidently say that this latest version's user interface gives the developer unprecedented levels of consistency, predictability and effectiveness.

Overview of the Uniface 10 IDE

At a high level the IDE can be split into three main sections.


Demo
BROWSE
prj: DEMO_NAVIGATION
More Editors
STARTPAGE
DEMO_NAVIGATION
MYENTITY.DEMO
DEMO_FORM
INCLUDEPROCLBI
Define Project
Compile
Actions
Templates
Model
Objects
Tables
prj: UPALETTE_PRJ
Alternative Name
Project
Modeled Entity: Not in database
Modeled Entity: In database
Application Shell: Character Mode
Application Shell: Mobile
Application Shell: Windows
Application Shell: Userver
Component: Dynamic Server Page
Component: Entity Service
Component: Modal Form
Component: Non-modal Form
Component: Report
Component: Service
Component: Static Server Page
Modeled Component: Dynamic Server Page
Modeled Component: Entity Service
Modeled Component: Modal Form
Modeled Component: Non-modal Form
Modeled Component: Report
Object
Messages
Compiler Output
Header
Editors
Message output

Name	Type	Description
DEMO_NAVIGATION	Project	
MYENTITY.DEMO	ENT Reference	
DEMO_FORM	CPT Reference	
INCLUDEPROCLBI	LIBINC Reference	
PROCLIB	LIBPRC Reference	

Property	Value
Object	
Name	INCLUDEPROCLBI
Type	LIBINC Reference
Location	prj:DEMO_NAVIGATION/...
Description	

```
Compile done:  Entity compilation (collection operations): no compilation messages
Compile entity: MYENTITY.DEMO
Compile done:  Entity compilation (occurrence operations): no compilation messages
Compile Global Proc module: 'EMPTY'
Global Proc module: EMPTY
Compile Global Proc module: 'EMPTY'
Global Proc module: EMPTY
```

Figure 1 IDE main sections

As shown in Figure 1, these main sections are header, editors and message output. The last section, message output, contains the output of actions performed in the IDE. Pretty clear isn't it? In this article I want to focus on the other two sections, because these are where the magic happens.

A Uniface developer works in the editors' section. The IDE contains several editors. Let's start by understanding these.

Main development objects and their editors

Uniface has introduced the term *object* in version 10. It's a general term for the 'things' developers create and use, such as projects, components and fields. An object is a way for developers to communicate about their work without the need to bother with implementation details. In the article 'About templates, models and objects' I have described this concept in more detail.

Objects can be nested as children and parents. Some objects, such as fields, can only exist as children of other objects. Other objects, called *main development objects*, can exist without a parent and can have children of their own: examples include modeled entities, start-up shells, and projects

All main development objects have a dedicated editor in Uniface 10. When an object can only exist as a child of another object, that child must be maintained in an editor created for the parent.". For instance, an entity painted on a component is the child of that component and must be maintained in the parent component's editor.

It's important to understand the difference between an entity created in the entity editor, which is called a modeled entity, and an entity that is painted on a component in the component editor, which is called a derived entity.

Uniface has a specific editor for every main development object. In the non-modal development environment, a developer can work on several objects in the same environment. Every open editor is identified by a tab.

Scope of your action

I started this article by talking about the predictability of the IDE. In the previous paragraphs I explained that Uniface has an editor for every main development object. The daily activities of the Uniface developer are actions such as creating new components, modifying properties of objects, writing script code, and compiling components. Some *actions* will be started from the IDE header, others in the editor section. To find the location of the action you want to do in the IDE, it’s good to consider the scope of your action.

- Does it do something to exactly one opened object? It will be in the editor of that object.
- Does your action involve more than one object, or is it not particularly related to an object in In short: The header contains global actions, while the editor contains actions relating to the content of the editor.

A few examples:

Compile all components in the repository. This is an action that is not bound to exactly one opened component. This compile action can be found on the main menu in the header.

Compiling a component. This is an action for an open object in the component editor, so it’s to be found somewhere in the editor. Of course, it is also possible to compile the same component from the compile option in the header. The result is the same.

Export all child objects from a project. The boundary of this action is the project’s content. Therefore, it is to be found in the project’s editor.

You can draw an imaginary line between the header and the editor section. Everything that relates to one *particular* object is somewhere below that line.

Worksheets

Let’s focus now on this editor section. "Although most editors have been rebuilt in Uniface 10, some of the less commonly used objects – for instance the maintenance of glyphs and keyboard translation tables – still use the U9 editors. As part of Uniface’s continuous delivery, these editors will be migrated to the new structure in the future. Every experienced Uniface developer will recognize the U9 versions immediately though.

The new editors are composed of one or more worksheets tailored to the object they serve. An entity editor consists of different worksheets from those in a component editor. The strength of these new editors is best demonstrated with the component editor. A form component is in other ways similar to a dynamic server page (DSP) – they are both components – but at the same time it is different. The form has a client-server interface, while the DSP has a HTML GUI that can contain JavaScript. Some worksheets are the same, and others differ.

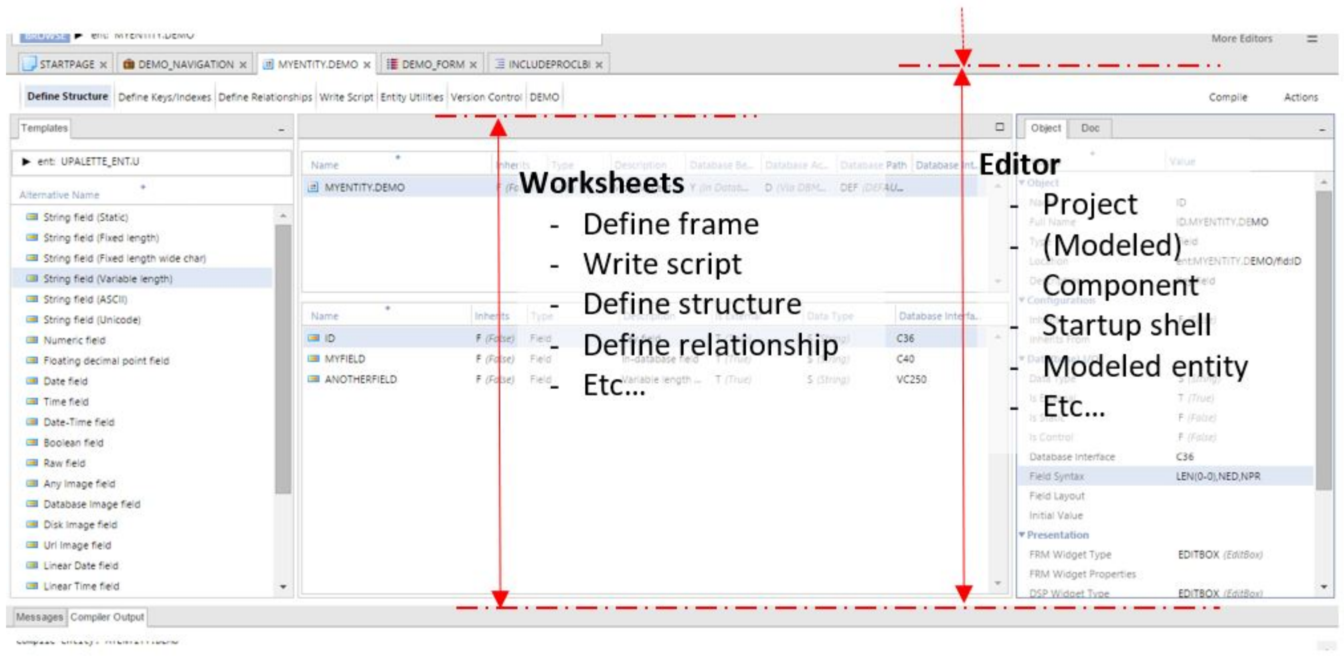


Figure 2 The editor is a collection of worksheets

Every worksheet contains one or more tools. These tools depend on the editor and the worksheet. A typical worksheet composition has a resource browser on the left, the properties inspector on the right, and a large tool pane in the center. It's in this tool pane the actual editing work is done. In this container the developer can, for instance, edit the script, define the key fields of an entity, or compose the structure of a service component. The general structure in the IDE is: editor à worksheet à tool.

Let's take the editor shown in Figure 2 as an example. This is the modeled entity editor. The active worksheet is the Define Structure. This worksheet contains three tools. the left pane contains the 'Resource Browser', in the middle is the 'Fields Composer', and on the right is the 'Properties Inspector.'

User-defined worksheet

Zooming in on the worksheet tabs, I want to show you something interesting.

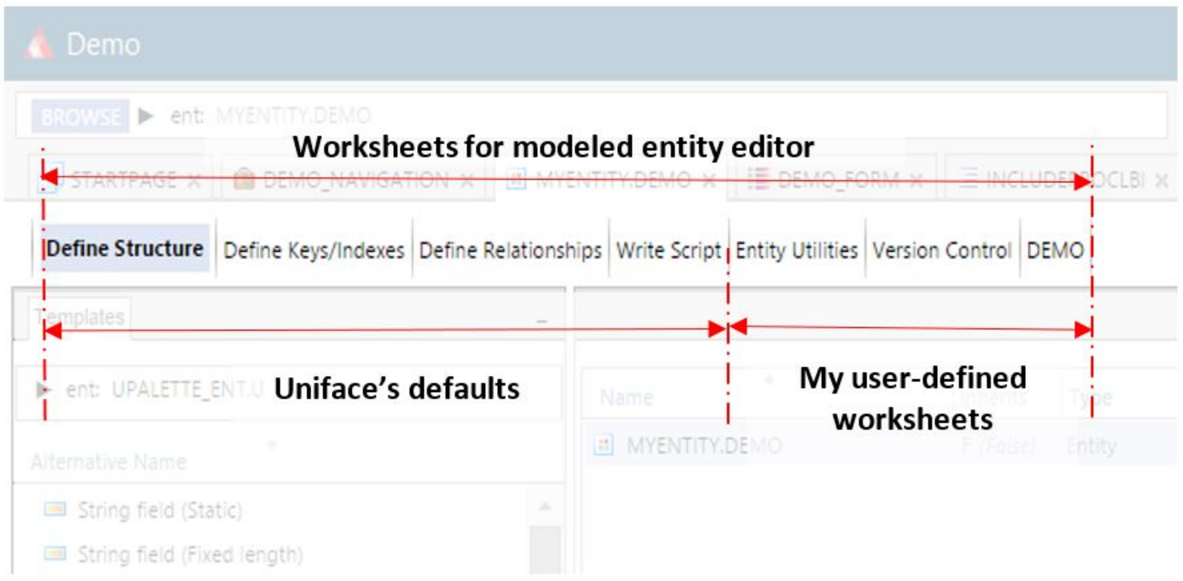


Figure 3 The worksheet tabs: my worksheets next to the Uniface defaults

In Figure 3, the Uniface default worksheets are shown and next to them are my own worksheets. In another article on 'Enhancing the provided toolset', I described how these worksheets can be added to your preferred Uniface editor(s), to make your work as a Uniface developer even more efficient. The worksheets offer great flexibility and power. In your own worksheet(s), you are not bound to the tool panes Uniface is using, and your actions have unlimited scope. You, the developer, are completely free to add a worksheet that performs actions outside the open object, though it is advisable to take the context of the object into consideration.

Summary

The IDE of Uniface 10 has the predictability I expect because it is a consistent implementation of the Uniface development paradigm. Uniface 10 is built by developers, for developers. It makes sense – common sense.

I can talk and write about the IDE for hours, but the best advice I can give is: just use it. I am convinced that you will, just like I do.