Summary of project - Boyum

This is a short description of the solution given to the problem regarding reading a XML file and displaying its content in a WPF app.

Approach

By looking at the requirement specifications the first approach for me is to design the central models of the "business". This regarding the Order and Item models and the including logic. When these where in place the next step was to ensure the data foundation of the system by defining a path from data source to datamodel. From here on it was a focus on how to display it in a viewmode in the WPF project. Actually before all this, I had to take descissions about the architecture, reasons for using Dependency Injection, logging frameworks, and solution structure.

Solution setup

The solution have three projects, where it is only the MPE.Boyum that is interesting. The last two are only for test and display reason to ensure the backen business logic worked.

MPE.Boyum

This project contains all the backend logic. Most of the projects classes are internal to limit clutter in using projects. It is a project using Dependency Injection (SimpleInjector) as a basis, so I'll be able to switch different implementations at any given time, and to make it easier to UnitTest in a real-world scenario.

The flow in the backend code is centered around the implementation of a IFileObjectReader. Its responsibility is to read the data from a given file, where the format required is determined by the implementation - In this case XmlFileObjectReader. If the file is parsing with no problems, it will call a IConverter that maps the XmlWebOrder model to be a business-model of the type WebOrder. This is to remove the Xml specific properties (XmlDate) used to handle the funky date format, but also to make a clear cut between the storage and the business layer.

The Converter implementation should, in a real world scenario, also contain a validation step to ensure that the data we are letting into our system is valid - perhaps a two step model, one focused on the XML (XSD schema) and another focused on the object state.

If the XML are not in a parsable format it will through an exception of the type ParseException that can be acted opon in the frontend.

It is also this project that contains the calculations to be displayed in the frontend. These are implemented in a seperate service, these could also be located inside the WebOrder and WebOrderItem models themselfs - it depends on coding styles, I tend to seperate them if I have to choose.

Mads Pedersen