Programming Applications with Databases

Exercise Set 9

There are given the following high-level main requirements of the e-Shop application:

- customer is able to browse the product catalog,
- customer is able to put products into the cart,
- manager is able to create and delete products.

Please keep in mind that the scope of the application should be very limited and basic.

1. Prepare prototypes of all different types of views, i.e. prepare a visual representation of how the application is going to look like. It can be created in any application including famous MS Paint or even MS Power Point, but usage of a dedicated *mockup* tool is highly recommended.
   [2p]
2. Following the attached example *DDDSample* create *application*, *domain* and *infrastructure* layers expressed by projects (in .NET) or in another way aligned with the selected technology. On top of that create *cross-cutting component* (e.g. *CommonComponent*, *GenericComponent*, or *Utility-Component*) and empty placeholder for the user interface.
   [1p]
3. Referring to the e-Shop requirements and prototypes created in the Exercise 1, propose a draft domain model, i.e. define
   - define objects and mark them as either entities or value objects,
   - for entities define their lifecycle, i.e. enumerate stages of "life" (e.g. order created, order completed, order payed, etc.),
   - propose aggregates keeping in mind transactions that are required to perform all operations in context of all views.

   As a result prepare appropriate class structure in the *domain* layer. At this stage each object needs to include all properties, however, no business methods are required for now.
   [3p]
4. For each aggregate create a respective repository interface in the *domain* layer and then the respective *in-memory* implementation in the *infrastructure* layer. Consider creating a generic base class implementation.
   [2p]
5. Create appropriate application service(s), i.e. for each domain model aggregate:
   - create an application service class with methods (empty for now) to support all methods required by UI views,
   - methods which supposed to return data (ensure the implementation that returns certain fake data for now).

   [2p]

Paweł Rajba