SOEN 387 Web-based Enterprise Application Design

Stuart Thiel

Concordia University
Department of Computer & Software Engineering

Fall, 2015

Outline

New Patterns, Old Responsabilities

Outline I

Domain Problem

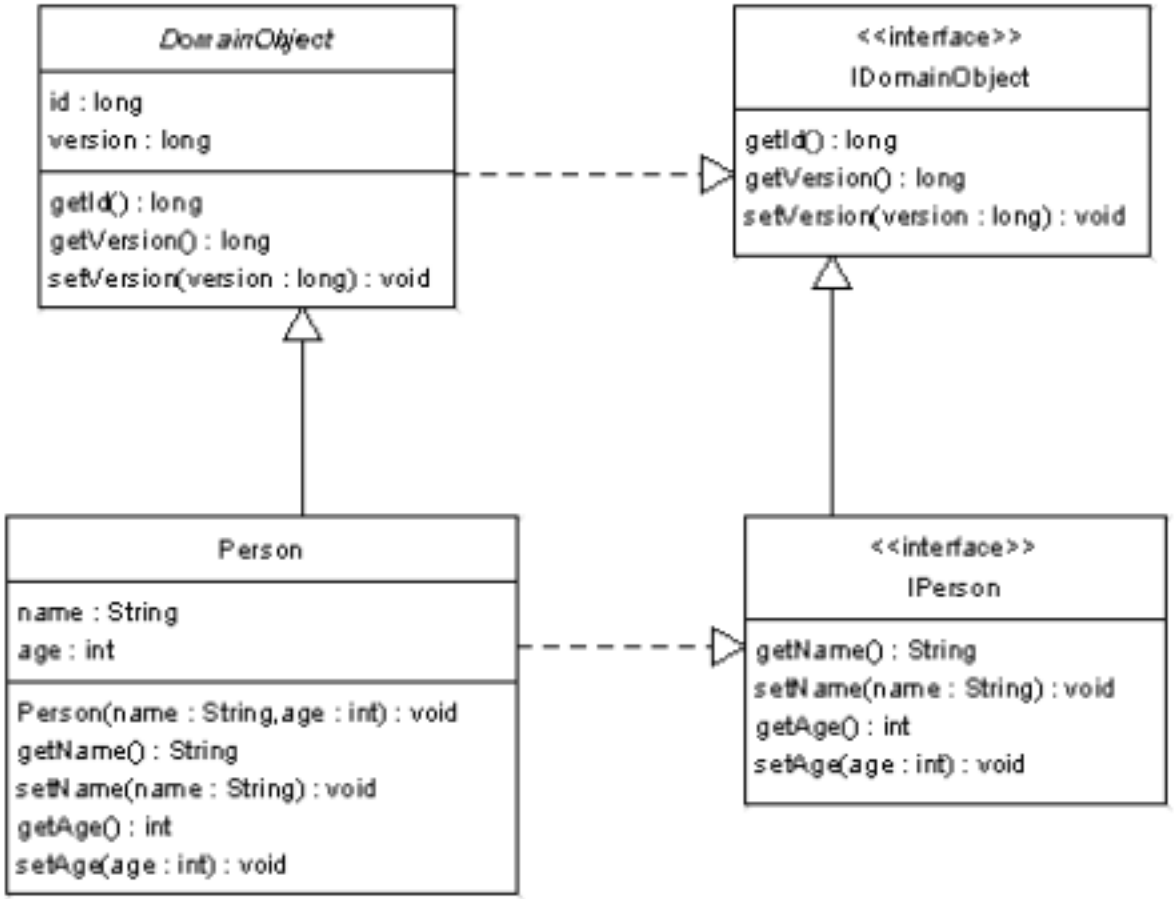
- We need to reduce representational gap
- We need to represent identity
- We need to represent a temporal identity
- We need to do this simply in a way that can reflect persisted data
- It may help to think of these as responsibilities that we have always had, we are just now specifically addressing them.

Domain Object

- We have already used Domain Object as a term, let's formalize it
- It represents necessary identity
- It isolates our data along conceptual boundaries
- It is easy to map to persisted storage using our existing patterns

Domain Object UML

- Why not treat it as Layer Supertype?



Domain Object Alternatives

- In SOENEA we support different id types besides long
- . . . this includes compound ids. Is that a good idea?
- Does every Domain Object need a version?
- Could we move some of the responsibility of interacting with UoW into Domain Objects?

Outline I

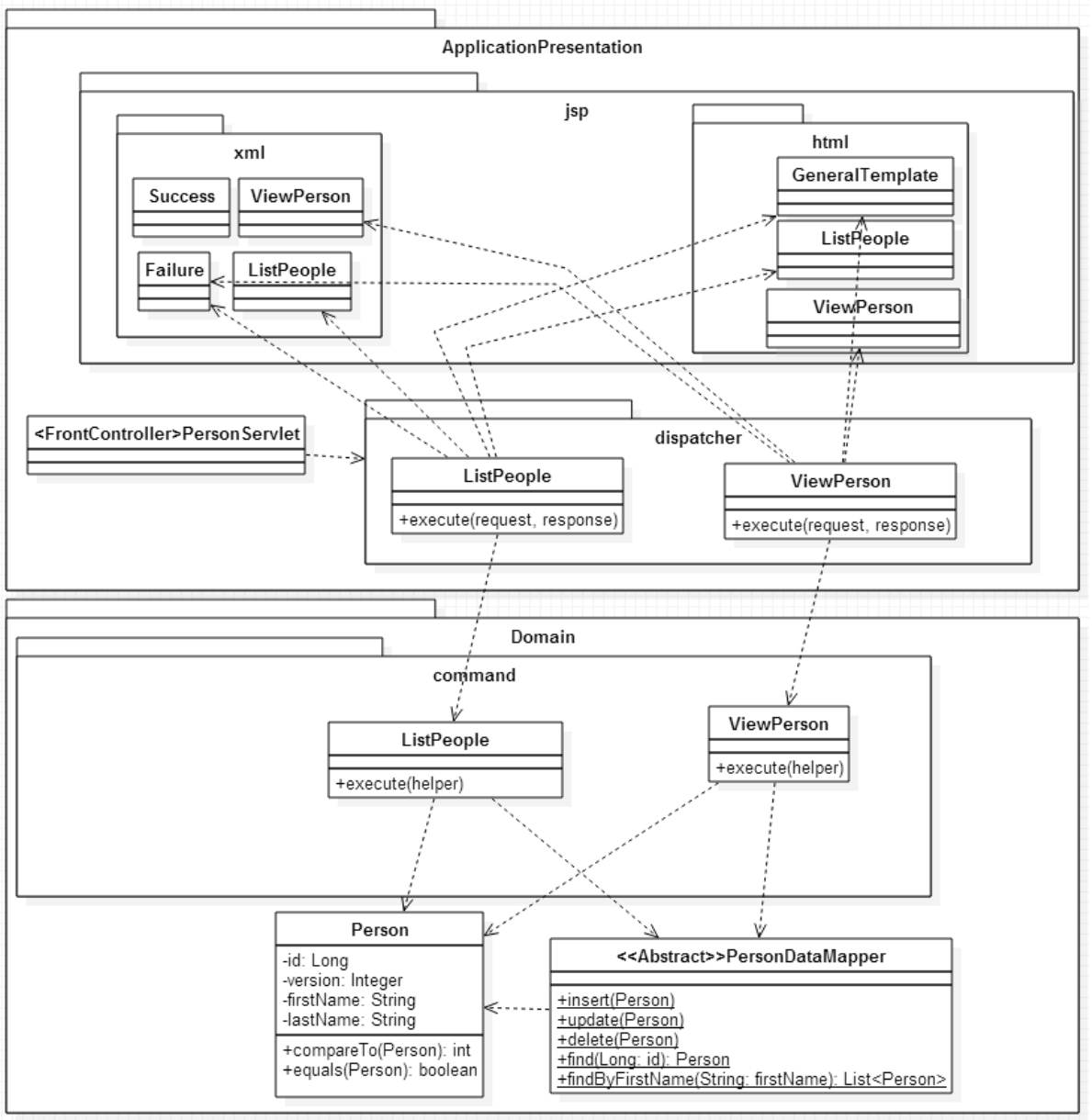
The Problem With Front Commands

- What Layer are Front Commands in?
- What do they do?

Dispatchers

- Commands can be in the domain layer if they are strictly domain logic
- Split out interaction with views
- Dispatchers then Dispatch to Commands and to Views
- Dispatchers read like high-level use cases
- Commands read like low-level use cases

Dispatcher UML



Outline I

Lots of Lists

- Warning: Over the years, I've become less fond of this approach. But...
- Sometimes your classes have fields that are lists of other things
- Really, this probably applies to any collection (Lists, Sets, Maps, Heaps)
- We have seen proxies as a means to load less data
- Can we use proxies for the same thing here?

Lists of Proxies

- What if instead of loading up each Domain Object we loaded up a list of their ids
- We could then populate a list with proxies
- Proxies only loaded as needed
- Table with this list of ids is often a separate table anyway
- Is this a good enough reason?

Proxy Lists

- What if the whole list was a proxy?
- What if it knew the parent id, and type of thing wanted
- Effectively, it knows which find method to call and has a Domain Object to pass in
- Database call gets all rows at once, only when needed
- Does this sound like it works?

Hint about a solution we'll see later

- These solutions still try to hold these lists as structurally relevant
- Maybe they are!
- But maybe they're just needed in certain cases. . . behaviourally relevant
- Then we can skip proxies like this altogether