Theory: Data and object mapping

12 minutes 0 / 4 problems solved

Programming languages may include primitive types, classes and data structures to store information. If you want to modify an object or interact with it somehow, you would prefer to do it in your favorite programming language, wouldn't you? As a developer, you use tools you're familiar with. In the local environment, you can complete most of your tasks this way. However, to communicate with other systems and to store data persistently, you need to convert your local types to commonly used data representation.

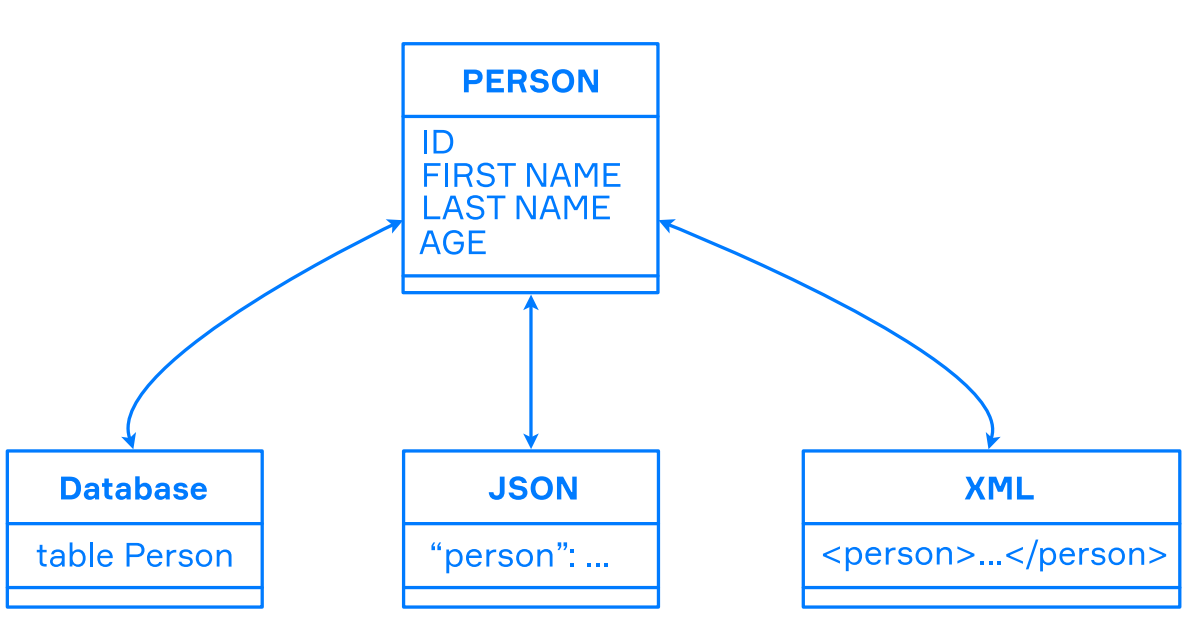
§1. Data mapping

Assume that you are developing a social network service. Every moment, some users search for a friend and the service should reply with some information about other users in the network.

We represent each person as an instance of the `Person` class in the code. Let's stick to a simple diagram to avoid using any specific programming language:

The problem arises when you want to retrieve the information about a person from the storage you use. Relational databases often play the role of the storage but in the form of tables and relations between them. Luckily you're not the first person that faces the task to convert one common data type to another. Just find a library for **data mapping** and it will help you a lot.

Data mapping is matching fields of **source** data representation to the fields of its **destination**. In our case, the source is a database, and the destination is the `Person` class, but the operation is applicable vice versa. In other tasks, we could use other mappings for the `Person` class and convert it to/from JSON, XML.



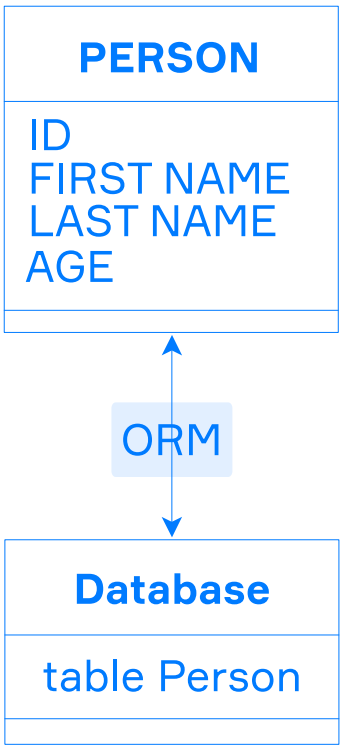
We use data mapping to convert the information from one type to another, but we don't synchronize the data changes over time by default, we only want a different view at the particular moment.

§2. Object Mapping

The structure of data in the source system may be different from the structure of data in the destination system. We could store the information about the name in one place and the information about the age in another because retrieving data to `Person` class is not the only purpose of the storage. This time data mapping will only match the part of the data we needed. For complex operations like this one, we need a more appropriate instrument such as **object mapping**.

Object mapping is matching data from a source system to a complex object in the destination system.

The most commonly used example is **ORM** (Object-Relational Mapping) when the data from relational databases is matched with the classes in object-oriented languages.



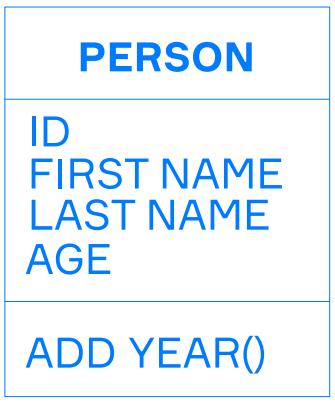
The main distinction between data and object mapping is that object mapping not only stores data but also emulates the behavior of an object and reflects changes in its source system. As a developer, you can call methods of the class, and data mappings will change at the same time. In short, object mapping is like data mapping, but with high-level control over changes.

§3. The life cycle of mapping

To see what is data and object mapping in action, let's continue our work with the social network. We receive a query with some name and look for it in our data storage. As soon as we find an appropriate result, we retrieve the information about the person from the database. We fill the class instance with the result of the search.

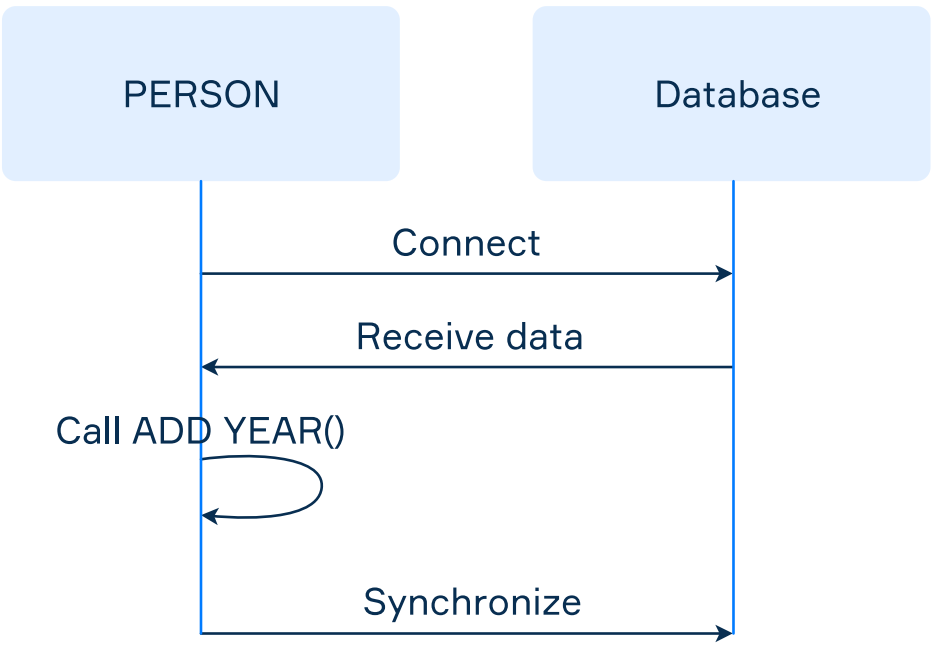
Now our class is only a data container and nothing more, and this action is an example of data mapping. If we store the object in the database, it's a reversed example.

The age of a person is changing over time, but the storage is not aware of such changes, so we implement a method that adds a year to the age attribute:



After the age is changed in the instance of the class, the value in the storage becomes outdated. If we want to mirror it, we should put some effort into working with the storage, then we need to synchronize the information. This

can take place in the same method or we can define another one for it.



This operation refers to object mapping; we are not only changing the object but also synchronizing data for it. Now our object has actual information in the storage, too. Synchronization is one of the important features of mapping; we want to have the definite representation of an object in our language and in the storage it uses. However, it's not obligatory to synchronize objects every time we change them. Sometimes it's more efficient to accumulate all the changes before the synchronization.

§4. Conclusion

We have learned two new concepts: data mapping and object mapping. We use data mapping to match data in two different systems, and we use object mapping to represent more complex objects and to add control from one system to data in another.

Report a typo

283 users liked this theory. **41** didn't like it. **What about you?**

Start practicing

Comments (10) Hints (0) Useful links (0) Show discussion