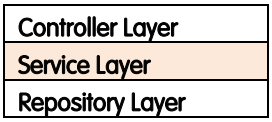
Serverapplikationen

Serverapplikationen werden ihrerseits wieder in Schichten aufgebaut. Einen typischen Aufbau zeigt die folgende Abbildung:



Controller Layer

Im Controller Layer werden Anfragen entgegengenommen, die Befehle analysiert und entsprechende Aktionen im Service Layer ausgelöst. Anschliessend werden ggf. Resultate an den Client weitergeleitet.

Damit sich der Client und der Controller verstehen, müssen beide dasselbe Protokoll benutzen. Weiter müssen sie sich einig sein, welche Art von Daten sie miteinander austauschen wollen (HTML, JSON etc.).

Service Layer

Das Service Layer stellt die eigentliche Applikation dar. Es enthält die Business-Logic, also alle Funktionalität welche unser Programm anbietet.

Damit das Service Layer seinen Job erledigen kann, braucht es Daten, sogenannte Business Objects. Business Objects kapseln die Daten der DB und bieten Funktionalität zum Ändern ihres Zustands. Kurz, es sind ganz gewöhnliche OO-Objekte.

Persistence Layer (Repository Layer)

Das Repository Layer hat die Aufgabe, Daten aus einem Datastore zu besorgen und allfällige Zustandsänderungen der Business Objects im Datastore persistent zu machen (dauerhaft abzuspeichern). Es dient als Abstraktionsschicht für den Datenzugriff.

Der Begriff Datastore bezeichnet in diesem Zusammenhang alle möglichen Arten von Datenquellen. Das können SQL-Datenbanken sein, das Filesystem, eine Objektdatenbank oder ggf. auch ein weiterer Server welcher uns Daten liefert.

Business Objects

Businessobjekte sind einerseits Abstraktionen (Was ist typisch für einen Film? Er hat einen Titel, eine durchschnittliche Bewertung und eine Liste von Kommentaren) und andererseits implementieren sie Funktionalität (Eine neue Bewertung oder einen neuen Kommentar hinzufügen).

Diese Unterscheidung ist wichtig. Im Service Layer interessiert uns nur die Abstraktion (Gib mir die durchschnittliche Bewertung dieses Filmes). Alle modifizierenden Funktionalitäten dürfen nur im Persistence Layer aufgerufen werden damit sichergestellt ist, dass eine Änderung des Objektzustandes auf den entsprechenden Datastore geschrieben wird. Die Datenkonsistenz zwischen Objekten im Speicher und dem Datastore muss immer sichergestellt sein.

Für die Implementierung trennen wir deshalb die Abstraktion von der Implementierung. Das Service Layer kennt nur die Abstraktion (pubic interface Movie { int getRating(); …}) während das Repository Layer die konkrete Implementierung benutzt (public class MovieImpl implements Movie { int getRating() {…}; void addNewRating(int rating) {…}; …), und seinerseits passende Modifikationsfunktionen anbietet (public interface MovieRepository { void addRating(Movie movie, int newRating); }).

Damit stellen wir sicher, dass nicht versehentlich ein Businessobjekt geändert wird ohne dass die Änderung im Datastore landet.