Erledigt

Model-View-Controller (MVC) Explained Through Ordering Drinks At The Bar

Beitrag von „d-vise“ vom 12. April 2018, 12:54

Eine tolle Darstellung!

Inhaltlich bin ich mir nicht sicher, ob ich wirklich komplett einverstanden sein kann, aber fast. Wie gesagt, die Darstellung ist sehr gut.

Ich füge hier zum Vergleich zwei Darstellungen aus den Handbüchern von Smalltalk 80/ObjectWorks als Attachments an. Smalltalk 72 und später Smalltalk 80 waren die ersten Systeme, die dieses Pattern genutzt und quasi eingeführt haben. Die Darstellung in den Handbüchern ist ebenfalls hervorragend. Damals waren Bücher noch mehr oder weniger die einzige Möglichkeit Wissen zu übermitteln, aber das wisst Ihr ja..

Es geht bei dem Muster im wesentlichen darum, Komponenten stecken zu können. Das war die ursprüngliche Idee. Daher ist der Controller fast immer auf den View abgestimmt, aber das Modell ist im Grunde ganz unwissend, wer es modifiziert und wie und warum. Es reflektiert die Änderungen einfach – übrigens fast immer synchron -, so dass beliebig viele Views (aka Observer) darauf reagieren können.

View und Controller treten immer paarweise auf, weil der Controller die Nutzerinteraktion mit dem View ermöglicht (oder teilweise unterbindet, Stichwort "No Controller" oder "Read Only Controller").

Die Controller selbst formen eine eigene Hiearchie, die in Smalltalk mit dem "Chain of Responsibility Pattern" umgesetzt worden ist. Diese Hierarchie erlaubt es beispielsweise, sämtliche Menü-Optionen (in Smalltalk auf der rechten, "blauen" Taste) aus den Views und ihren Super-Views zusammenzuziehen. Die Views sind ebenfalls in einer Hierarchie organisiert (Composite Pattern), die unabhängig von der Controller-Hierarchie ist.

All das ist etwas aufgeweicht worden, als in den frühen Jahren Web-Frameworks ihre grundlegenden Muster mit MVC beschrieben. Mir fällt hierzu als Beispiel für ein frühes

Framework Apache Struts ein.