Model View Controller (MVC) - Part I (Theorie)

== Was ist MVC überhaupt? ==
MVC ist ein Design Pattern zur Softwareentwicklung. Es besteht aus 3 großen Teilen, die (wie der Name schon erahnen lässt) Models, Views und Controller sind. Ziel ist es Eingabe, Ausgabe und Verarbeitung einer Anwendung zu trennen. Im zweiten Part dieses Wikieintrags werden wir das ganze anhand einer kleinen Community-Anwendung in PHP durchgehen. Die Übertragung in andere Sprachen sollte kein Problem sein.

[Blockierte Grafik: http://img819.imageshack.us/img819/6996/cou4.png]
Quelle: http://stackoverflow.com/questions/19023309/whats-the-correct-way-of-implementing-mvc
== Model ==
Ein **Model** beinhaltet die Daten deiner Anwendung. Es stellt die Daten aus der Datenbank oder dem Dateisystem zu Verfügung, verändert diese jedoch auch. In deiner Community-Anwendung, gibt es zum Beispiel das Model „user_model“, welches Funktionen zum Auslesen der Userdaten (Passwortprüfung) oder auch verändern der Userdaten (Passwortänderung, .) bereitstellt.

== View ==
Ein **View** ist für die Darstellung von Daten (sprich die HTML-Ausgabe an den Browsern) zuständig. In deiner Community-Anwendung wäre zum Beispiel „user_profile_view“ ein View, der die Benutzerdaten in einem Profil darstellt. Eigentlich kann man den View auch als Template bezeichnen, da es meistens nur HTML-Code mit Template-Variablen sind.

== Controller ==
Ein **Controller** ist für die Interaktion zwischen User und Anwendung zuständig und reagiert mit entsprechenden Aktionen (Model oder View) darauf. Das mag erst mal verwirrend klingen, ist aber im Prinzip ganz simpel an deiner Community-Anwendung zu erklären.

== Theoretisches Beispiel ==
Wir haben ein Formular zum Ändern des Passworts deiner User. Klickt er auf Senden werden die POST-Daten (Altes Passwort, Neues Passwort) an Beispielsweiße „index.php?controller=change_password&function=process“ gesendet. Unsere Anwendung erkennt durch einen sogenannten Router, dass der Controller „change_password“ aufgerufen werden soll und dessen Funktion „process“ ausgeführt wird. Die Funktion holt sich zuerst einmal über das „user_model“ das Passwort deines Users, um zu überprüfen, ob er denn das richtige Passwort eingegeben hat. Sollte er ein falsches Passwort eingegeben haben, lädt dein „change_password_controller“ den View „wrong_password_view“ und zeigt ihn im Browser an. Der View enthält im Prinzip nur HTML-Code, du kannst jedoch auch Daten hinzufügen, dazu allerdings erst im Code-Beispiel mehr.
Gehen wir als nächstes vom besten Fall aus, dass dein User sein richtiges Passwort angegeben hat. Nun ruft die Funktion „process“ unseres Controllers die Funktion „update_password“ des „user_model“ auf und ändert in der Datenbank das Passwort. Nach dem Ändern wird über den „change_password_controller“ der View „changed_password_view“ im Browser ausgegeben, der anzeigt, dass das Passwort erfolgreich geändert wurde.