ASP.NET MVC Design – Übungen

Du kannst auf dem Stand der Basic-Übungen aufbauen oder das Source-Paket von <https://github.com/garaio/aspnetmvc-intro/zipball/master> herunterladen.

# Step 1: Dependency Injection Framework einbauen

* Du kannst statt StructureMap auch ein anderes Framework Deiner Wahl verwenden. Passe die Anweisungen gegebenenfalls sinngemäss an.
* Füge Deinem Projekt StructureMap hinzu (<https://github.com/structuremap/structuremap/downloads> )
* Ersetze die Standard-Controller-Factory durch eine Version, welche StructureMap verwendet:

***In global.asax / Application\_Start:***

ControllerBuilder.Current.SetControllerFactory(**new** StructureMapControllerFactory());

***Anderswo:***

**public** **class** StructureMapControllerFactory : DefaultControllerFactory {

**protected** **override** IController GetControllerInstance(RequestContext requestContext, Type controllerType) {

**if** (controllerType == **null**) {

**return** **base**.GetControllerInstance(requestContext, controllerType);

    }

**return** (IController)ObjectFactory.GetInstance(controllerType);

  }

}

* Starte die Applikation und stelle sicher, dass sie immer noch funktioniert.

# Step 2: Abhängigkeiten des Controllers sichtbar machen

* Führe ein Refactoring so aus, dass alles, was der KontakteController benötigt, offensichtlich ist.
* Insbesondere soll er keine statischen Elemente mehr aufrufen.
* Der KontakteStore (oder wie auch immer Du ihn genannt hast) soll als Konstruktor-Parameter übergeben werden.
* Nimm die notwendigen Anpassungen vor, damit die Applikation wieder funktioniert.

# Step 3: Tests für den Controller

* Stelle Dir Deine Erwartungen an den Controller zusammen: Welche Operationen unterstützt er? Wie reagiert er auf welche Input-Parameter?
* Füge Deiner Solution ein Test-Projekt hinzu (z.B. mit NUnit oder MSTest).
* Füge Deinem Test-Projekt ein Mocking-Framework hinzu (z.B. Moq)
* Schreibe einen ersten Test, der sicherstellt, dass die Kontaktliste durch die Index-Operation vom Store geladen und dann an die View übergeben wird.
* Stelle sicher, dass Dein Controller sich verhält wie gewünscht.

# Step 4: Test-Driven Development: PLZ-Lookup

* Lade die Postleitzahlendatei von <https://match.post.ch/downloadCenter?product=2> (PLZ Light) herunter.
* Erstelle Tests, welche den Lookup in einer Reihe von Fällen testen (wie z.B. 1027 🡪 Lonay; 2610 🡪 St-Imier, Mont-Soleil, Mont-Crosin). Im Verlauf dieser Tests „entsteht“ das Interface der PLZ-Lookup-Klasse durch die Verwendung.
* Implementiere die PLZ-Lookup-Klasse so, dass alle Deine Tests grün werden.
* Nimm die erforderlichen Refactorings vor, damit die Prinzipien SRP, DRY und KISS eingehalten werden.

# Step 5: PLZ-Lookup in die MVC-Applikation integrieren

* Erstelle Tests, welche einen Controller beüben, der anhand eines PLZ-Strings ein JSON-Array mit den gefundenen Orten zurückgibt. Er soll dazu irgendeine PLZ-Lookup-Klasse verwenden können.
* Implementiere diesen neuen Controller, um die Tests zu erfüllen.
* Integriere diesen neu erstellten PLZ-Lookup mit jQuery/Javascript in das Erfassungsformular für neue Kontakte.
* Du stellst fest, dass weder die MVC-Applikation noch das PLZ-Lookup starr vom anderen abhängig ist.

# Step 6: Mehrsprachigkeit

* Extrahiere alle (oder mindestens einige der) zu übersetzenden Texte in Ressourcen-Dateien. Füge (einzelne) Übersetzungen in Französisch hinzu.
* Passe Deine Routen so an, dass an erster Stelle neu eine Sprache angegeben werden kann:  
  /de/kontakte/index statt /kontakte/index
* Erstelle eine Basisklasse für Deine Controller, welche die Initialize-Methode überschreibt und darin die CurrentThread.CurrentCulture und CurrentUICulture anhand der RouteData setzt.

# Step 7: Routing Fun

* Füge eine Route hinzu, die statt /fr/kontakte einfach /contacts heisst, aber das gleiche bewirkt.
* Prüfe, welche weiteren Anpassungen Du vornehmen musst, damit die Applikation funktioniert.

# Step 8: PLZ-Prüfung beim Speichern

* Integriere die PLZ zu Ort Prüfung beim Speichern der Kontakte. Stelle sicher, dass der eingegebene Ort zur eingegebenen PLZ passt und weise das Speichern sonst zurück.
* Führe diese Erweiterung testgetrieben aus und beachte die identifizierten Regeln und Grundsätze. Achte besonders auf DRY und SRP.
* Prüfe, ob hier ein Command Model hilfreich sein könnte.

# Step 9: Kontakte editieren

* Erweitere die Applikation um die Möglichkeit, Kontakte zu editieren. Achte auf DRY.

# Step 10: Autorisierung

* Passe die Applikation so an, dass nur eingeloggte User schreibende Aktionen ausführen können. Versuche, dies so einfach wie möglich zu machen.

# Step 11: Terminverwaltung

* Erstelle eine Kopie der Kontaktverwaltung mit den gleichen Funktionen, aber für Termine.
* Ein Termin hat einen Betreff, einen Anfang und ein Ende.
* Extrahiere die Gemeinsamkeiten im Sinne von DRY.