



Programmeren Gevorderd

Les 4 // Drie lagen architectuur

3 Tier Applications



Probleem met werkwijze tot nu

Code door elkaar

Geen duidelijke aflijning binnen project

Delen vervangen = moeilijk project zit verstrengeld = Spaghetti code



Oplossing?

Design pattern gebruiken

Duidelijke lagen en verantwoordelijkheden

Gescheiden code = makkelijker aanpassingen aan één laag

Gekende design patterns: MVC, MVVM, PAC, etc.

3 Tier Model

3 Lagen Model

Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.



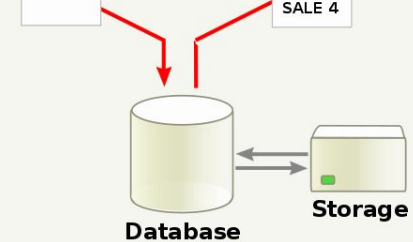
Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.



Data tier

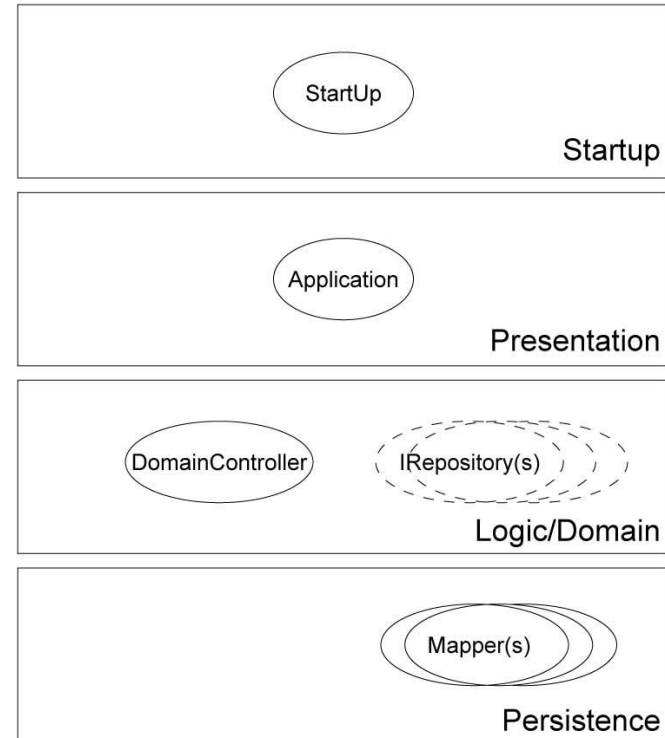
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.



3 lagen

1. Presentation
 - input & output voor gebruiker afhandelen
 - bevat zo weinig mogelijk logica, moet zo dom mogelijk blijven
2. Logic/Domain
 - kloppend hart van applicatie
3. Persistence
 - externe data inlezen/opslaan
4. Startup
 - geen volwaardige laag, staat apart om proper te werken

Elke laag kan nog extra klassen bevatten. Dit is de kapstok waar je ze aan ophangt





StartUp

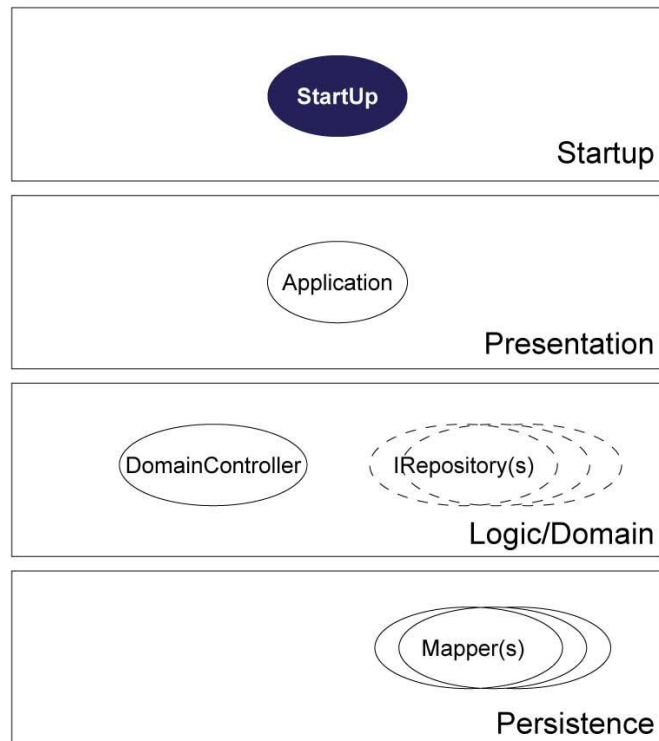
Startpunt van applicatie

Enige plaats die Main() methode mag bevatten

Staat in een aparte namespace & laag omdat dit nergens
thuishoort

StartUp gebeurt in 3 stappen

1. Repositories/mappers instantiëren
2. Domain controller aanmaken + repositories injecteren
3. Application aanmaken + domain controller injecteren





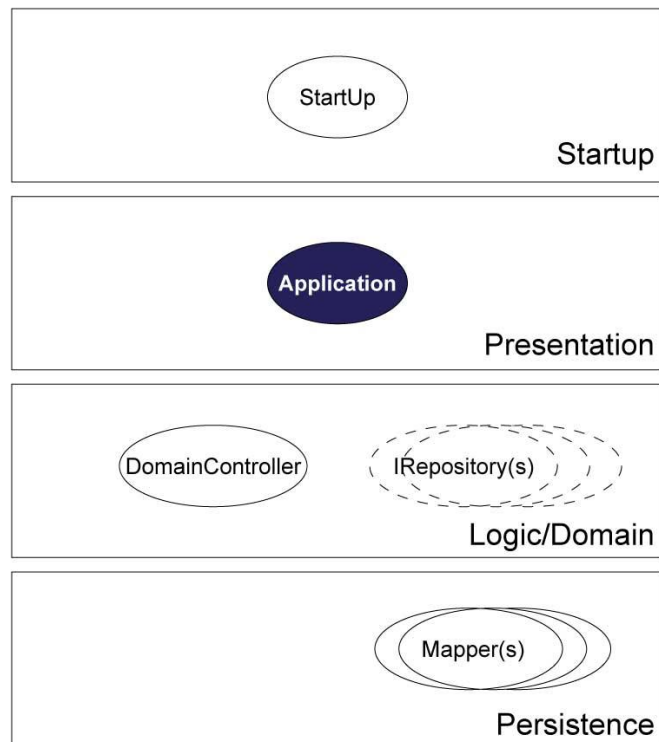
Application

Handelt gebruiker acties af

Toont output uit domain laag

Wordt aangemaakt door startup

Bevat referentie naar DomainController

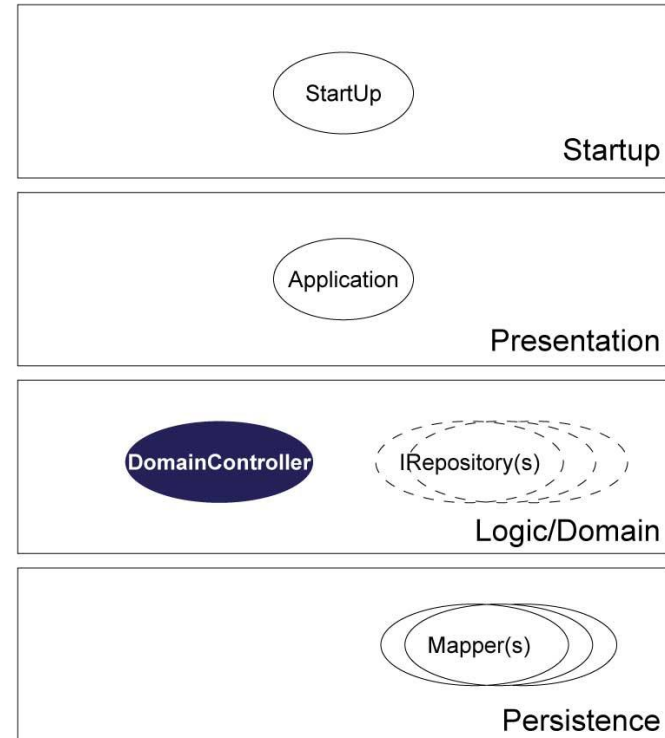




DomainController

Centraal punt waar alles op samen komt

Delegeert tussen verschillende lagen



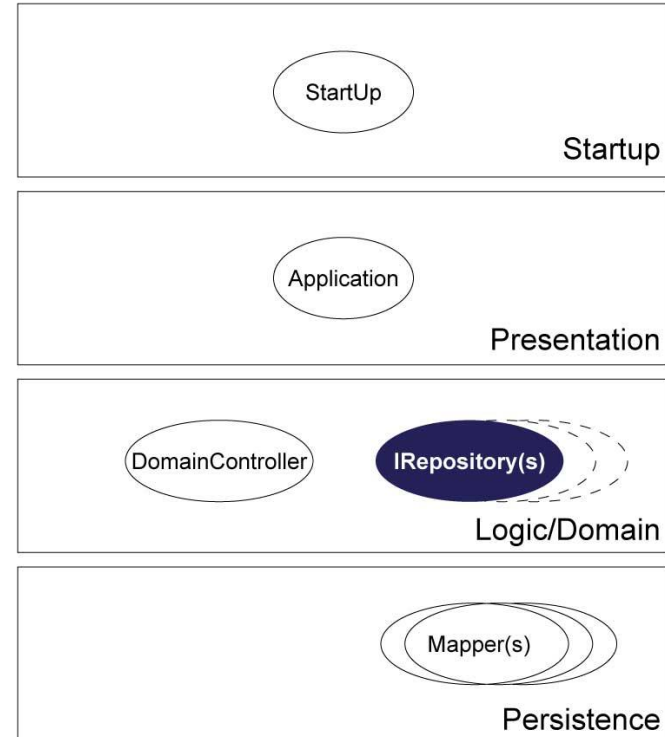


IRepositories

Interface

Abstractie om vervangen van persistence laag te
vergemakkelijken

Elke mapper implementeert eigen interface





Mappers

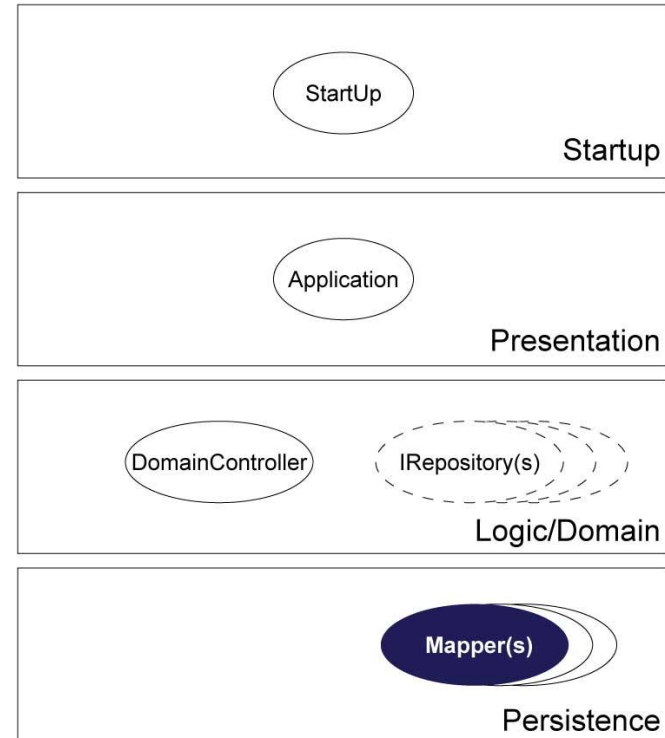
Een of meerdere mappers

Mappen applicatie data naar persistentie data

Bijvoorbeeld tekstfile naar objecten

Meestal één per klasse

Gebruik methodes om toegang tot data te beperken (bv GetAll(), GetById())





Wat moet in welke laag?

Presentation & persistence laag kunnen vervangen worden

Als een van de lagen vervangen wordt door een andere, mag noodzakelijke domain code niet verloren gaan

Domain layer is technologie agnostisch en wordt nooit vervangen

Voorbeeld: SetupGuide
