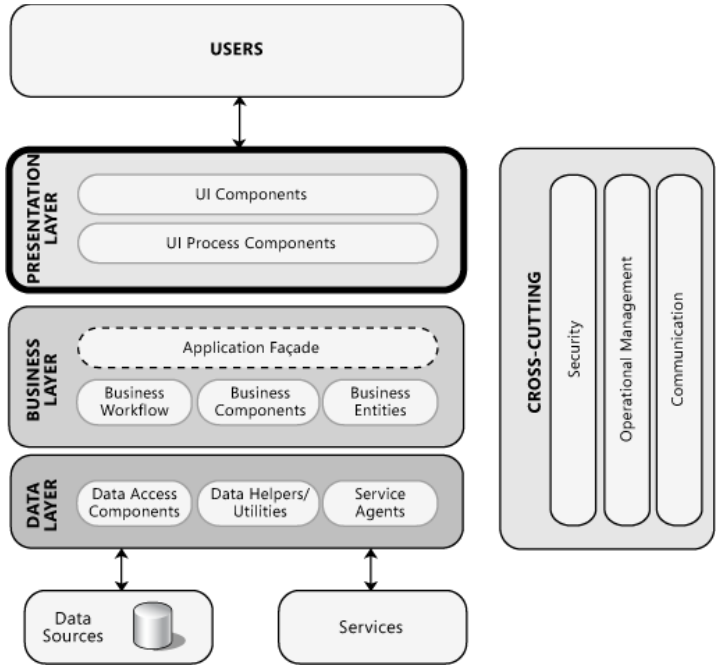
Prezentační vrstva

Štefan Németh, Jakub Macák

Prezentační vrstva

- prezentační vrstva zahrnuje komponenty, jež zajišťují zobrazení uživatelského rozhraní a interakci s uživatelem
- zabezpečenie vykonania aplikačnej logiky, resp. posun v zobrazení, na základe používateľských interakcií



Při návrhu prezentační vrstvy dbáme na:

- správná identifikace uživatele a jeho požadavků
- správné zobrazení dat a jejich validace
- korektní oddělení business logiky a prezentační vrstvy
- komunikace s ostatními vrstvami

Caching

- zlepšení výkonu aplikace a responzibility uživatelského rozhraní
- uchování výsledků náročných a opakujících se procesů
- bez duplicitních výpočtů
- použití: dynamicky generované webové stránky, ke kterým je přistupováno velmi často

Na co musíme dbát:

- kde budeme data uchovávat
- jaká data budeme uchovávat
- nastavení expirace a úklid nepotřebných dat
- načítání “cachovaných” dat
- pozor na caching duplicitních informací

Uchování dat

- na prezentační vrstvě
  - data jsou přímo zobrazována uživateli
- na business vrstvě
  - data, která nemohou být efektivně získána z databáze (výpočty), neuchováváme často se měnící informace
- v databázi
  - uchování velkého množství dat a přístup po částech

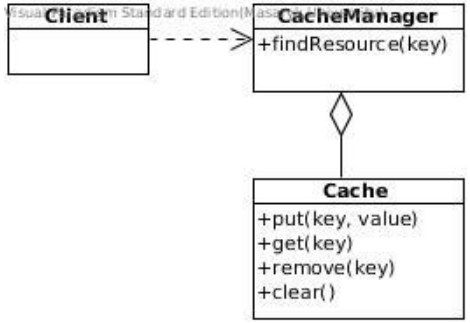
Guidliness (1)

- vyhýbáme se cachování dat pro jednotlivé uživatele (ztráta efektivity)
- vyhýbáme se uchování často se měnících dat
- chceme-li použít transformovaná data, měníme je před cachováním
- loose coupling, high cohesion

Guidliness (2)

- používáme asynchronní komunikaci
- vyhýbáme se dlouhotrvajícím atomickým transakcím
- uvolňujeme zdroje co nejdříve
- minimalizace objemu dat při přenosu
- důležitá je volba optimální datové struktury
- citlivá data musíme šifrovat

Příklady použití



- RSS feed
  - není nutné vytvářet feed z dat při každém požadavku, vytvoří se jednou (můžeme jej uložit do databáze)
- přehled poplatků za studium, přehled stipendií, …
  - mohou být počítány z externích zdrojů, složitější výpočty uchováváme, nemění se tak často

Composition

- znižovanie zložitosti pri návrhu, testovaní a údržbe
- menšie tesné väzby komponent so svojim okolím
- výhoda opätovnej použiteľnosti
- zjednodušenie modifikácie prezentačnej logiky

Čo musíme zvážiť:

Lahšia udržiavatelnosť ak prezentačná vrstva používa moduly a pohľady?

- minimalizácia kódu a závislosti knihovní
- minimalizácia závislosti medzi komponentami
- používať Template View pre dynamické stránky
- rozdelenie aplikácie na samostatné moduly, ktoré je možné meniť

Hodnotenie komponent

- hodnotenie na základe poskytovaných funkcií a z hľadiska behových parametrov

1. pamäťová náročnosť
2. vyťaženie procesoru
3. úroveň zabezpečenia prenosu

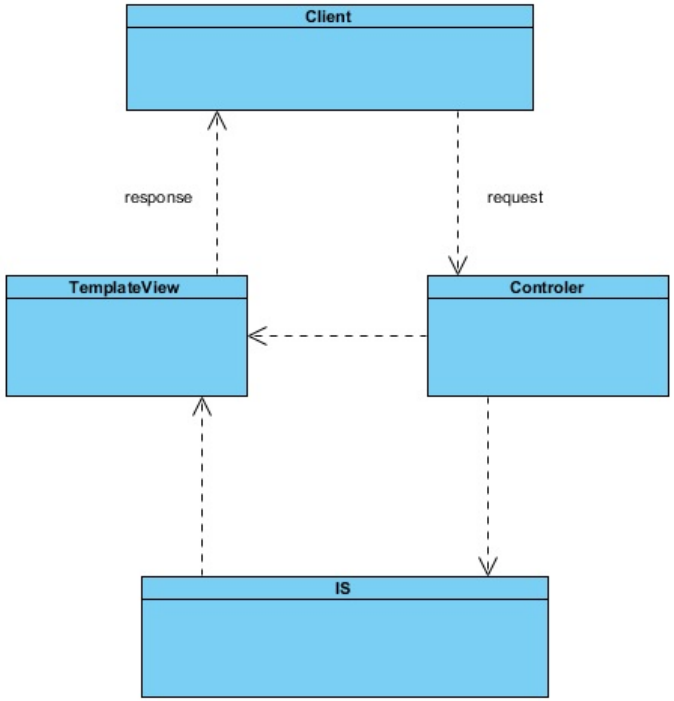
- mimofunkčné charakteristiky

Model View Controller

Model – poskytuje údaje pre zobrazenie, stav aplikácie
(Domain model, Tansaction script)

View (UI) – interpretácia dát z modelu a ich prezentácia užívateľovi

Controller (UI) – spracovanie udalostí, akcie na úpravu modelu



Zdroje

https://msdn.microsoft.com/en-us/library/ff647801.aspx#scalenetchapt03_topic12

https://msdn.microsoft.com/en-us/library/ee658081.aspx#Caching

https://msdn.microsoft.com/en-us/library/ee658081.aspx#Composition

http://www.guidanceshare.com/wiki/Application_Architecture_Guide_-_Chapter_10_-_Presentation_Layer_Guidelines#Presentation_Layer_Frame

http://www.slideshare.net/olandri/l17-presentation-layer-design