Frameworky architektury MVC

MVC, MVP, MVVM, Mode, View, Controller, Presenter, Page, PageModel, request, response, http

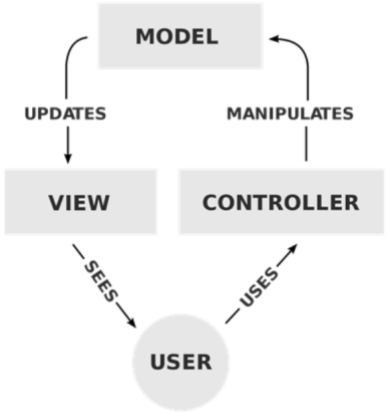
Joseph Soukup – P4

Architekturní návrhové vzory

- Návrhový vzor = zobecněné řešení problému používané pro řešení nějakého problému při návrhu softwaru
- Návrhový vzor je členění projektu pro lepší a efektivnější práci na něm:
    - Mezi více lidí
    - Na více na sobě nezávislých samostatných částí
    - Mezi frameworky
- Příklady:
    - C# - ASP.NET MVC (core)
    - Javascript – express, angular
    - Java – spring
    - Php – Lavarel, PHPCake
    - Python – Django, Flask

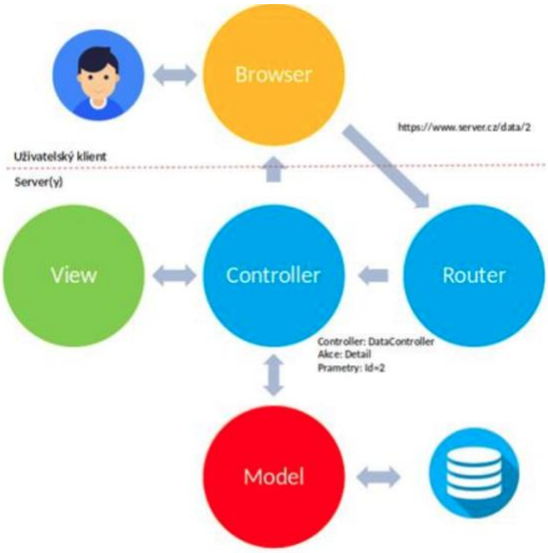
MVC

- **Model-View-Controller**
- **Model** = data
- **View** = výstup (data) pro uživatele
- **Controller** = příkazy od uživatele vyhodnotí a na základě toho požádá model o data
- Výhody:
    - View a Controller se nemusí starat o to, kde Model data vezme
    - Model se nemusí starat o to, jak budou data interpretována



Moderní MVC

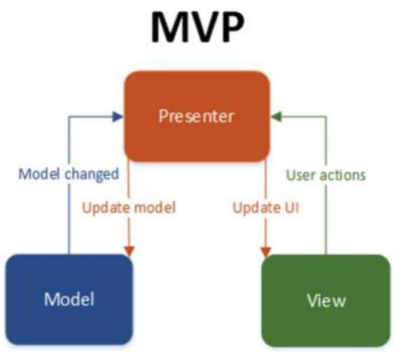
- **Controller** = Controller + Router
- **Model =** data
- **View =** sestavuje webovou stránku na základě dat
- **Controller =** řídí komunikaci, má navázáno několik akcí a View
- Prohlížeč požádá server o akce => Router vyhodnotí URL a předá Controlleru => Controller žádá Model o data z databáze => Controller žádá View o sestavení webové stránky s daty -> Controller pošle vyhotovenou stránku do prohlížeče



-

MVP

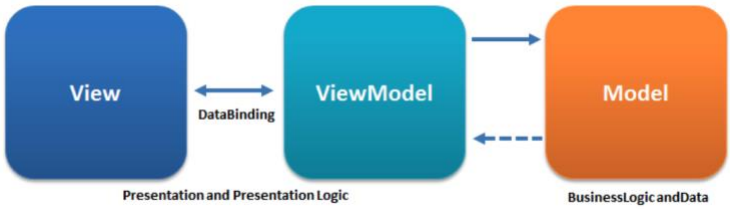
- **Model-View-Presenter**
- Stejný pattern jako MVC
- Controller je prezentován jako Presenter
- **Model =** data, upozorní Presenter přes událost v případě změny
- **View =** komunikace s uživatelem, upozorní Presenter přes událost v případě změny
- **Presenter =** řídí komunikaci a reaguje na změny dat v Modelu a ve View



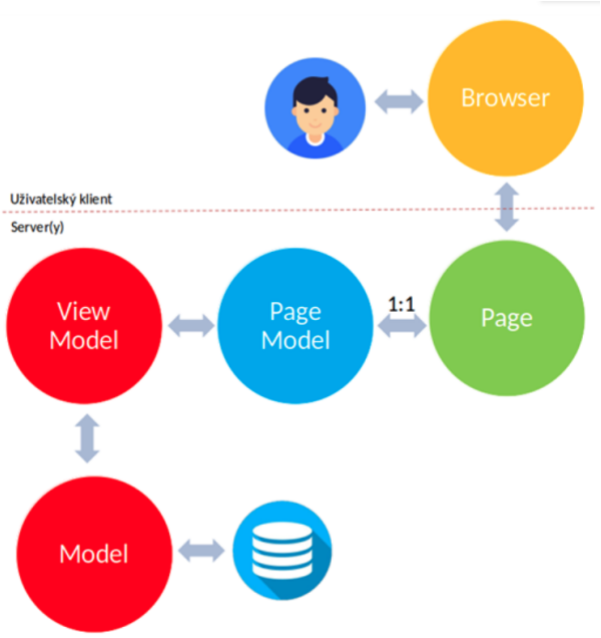
-

MVVM

- **Model-View-ViewModel**
- **Model =** data (komunikace s databází)
- **View =** komunikace s uživatelem (kam se mají data vložit)
- **ViewModel =** definuje způsob svázání dat pomocí databindingu, vnitřní logika aplikace

- 

PageModel

- Modifikace MVC vzoru
- Používá spojení Controller a View do jedné entity PageMode-Page
- Projekt se člení do jednotlivých souborů dle částí
- Role:
    - Model – data a stav aplikace
    - ViewModel – data připravená pro danou stránku a její úlohu (ViewModel, InputModel)
    - Page-Model – datová a aplikační logika pro stránku
    - Page – šablona stránky
- 

Specifika client-server webových aplikací

- Webové aplikace běží na serveru a zobrazuje je klient
- Pro komunikaci se používá protokol http
- Každá komunikace je kompletně samostatná
- Server si vůbec nic nepamatuje
- „Paměť“ serveru (např. přihlášení uživatele) je dosahována pomocí specifických mechanismů (cookies, sessions)
- Komunikace je založená na schématu Request -> Response
- Request: klient zažádá o nějaká data
- Response: server je poskytne, nebo vrací chybovou zprávu (404, 200, 500)

Styly komunikace

- Remote Procedure Calls (RPC) - Aplikace poskytuje libovolně pojmenované funkce, které klient volá a předává jim parametry
- Representional State Transfer (REST) - url reprezentuje přímý přístup k datům
- REST metody: GET, POST, DELETE, PUT, PATCH
- http://www.server.cz/knihy => vrací knihy
- http://www.server.cz/knihy/4 => vrací konkrétní knihu s identifikátorem 4

Multi Page Aplikace

- Tradiční
- Aplikace se skládá z více stránek
- Při změně je pokaždé vyrenderována nová stránka
- Simplicistní, však pracnější kódování
- Aplikace běží z větší části na serveru

Single Page Aplikace

- Aplikace je tvořena jednou stránkou
- Aplikace převážně běží na straně klienta, prostřednictvím skriptů, pomocí kterých jsou jednotlivé části postupně nahrazovány
- Úspornější na přenos dat
- Nízká bezpečnost kvůli nutnosti použít skripty pro samotnou funkci stránky