Rozwiązania szkieletowe w tworzeniu aplikacji www

Tomasz Łukaszuk

Katedra Oprogramowania
Wydział Informatyki
Politechnika Białostocka

Temat wykładu:

Wprowadzenie do frameworków webowych

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
NCBR Narodowe Centrum Badań i Rozwoju

Materiały przygotowane w ramach projektu "PB 5.0 – dostosowanie oferty dydaktycznej Politechniki Białostockiej do potrzeb nowoczesnej gospodarki oraz zielonej i cyfrowej transformacji" (nr umowy FERS.01.05-IP.08-0327/23-00) w ramach programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027 współfinansowanego ze środków Europejskiego Funduszu Społecznego Plus.

Wprowadzenie do frameworków webowych: Agenda

1. Podstawowe pojęcia
2. Wzorzec projektowy MVC
3. Django i Ruby on Rails

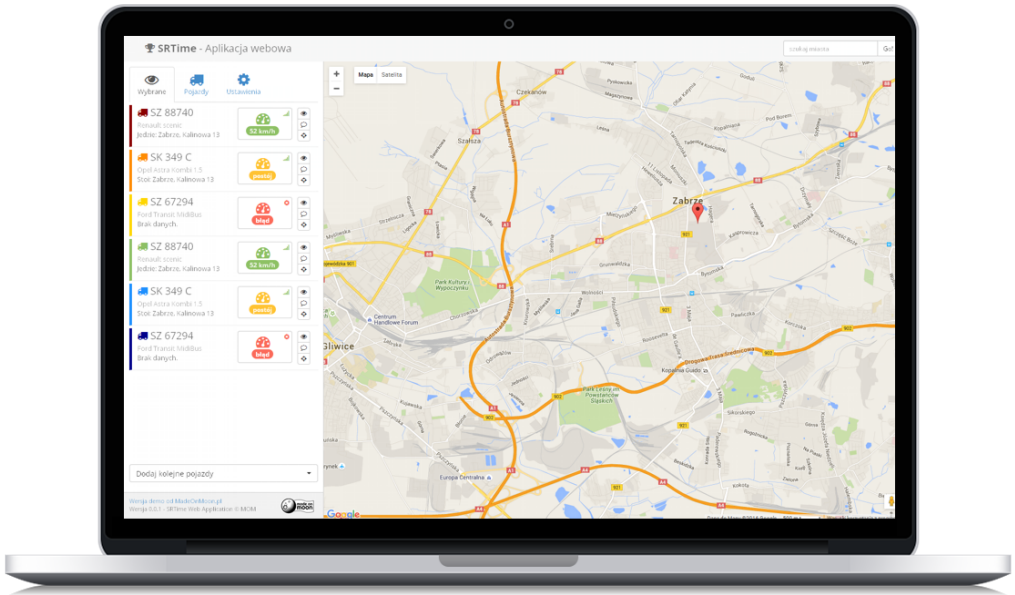
Podstawowe pojęcia

Podstawowe pojęcia

- aplikacja webowa
- framework webowy
- protokół HTTP, żądanie (request), odpowiedź (response)
- agent, serwer
- MVC

Aplikacja webowa, aplikacja internetowa

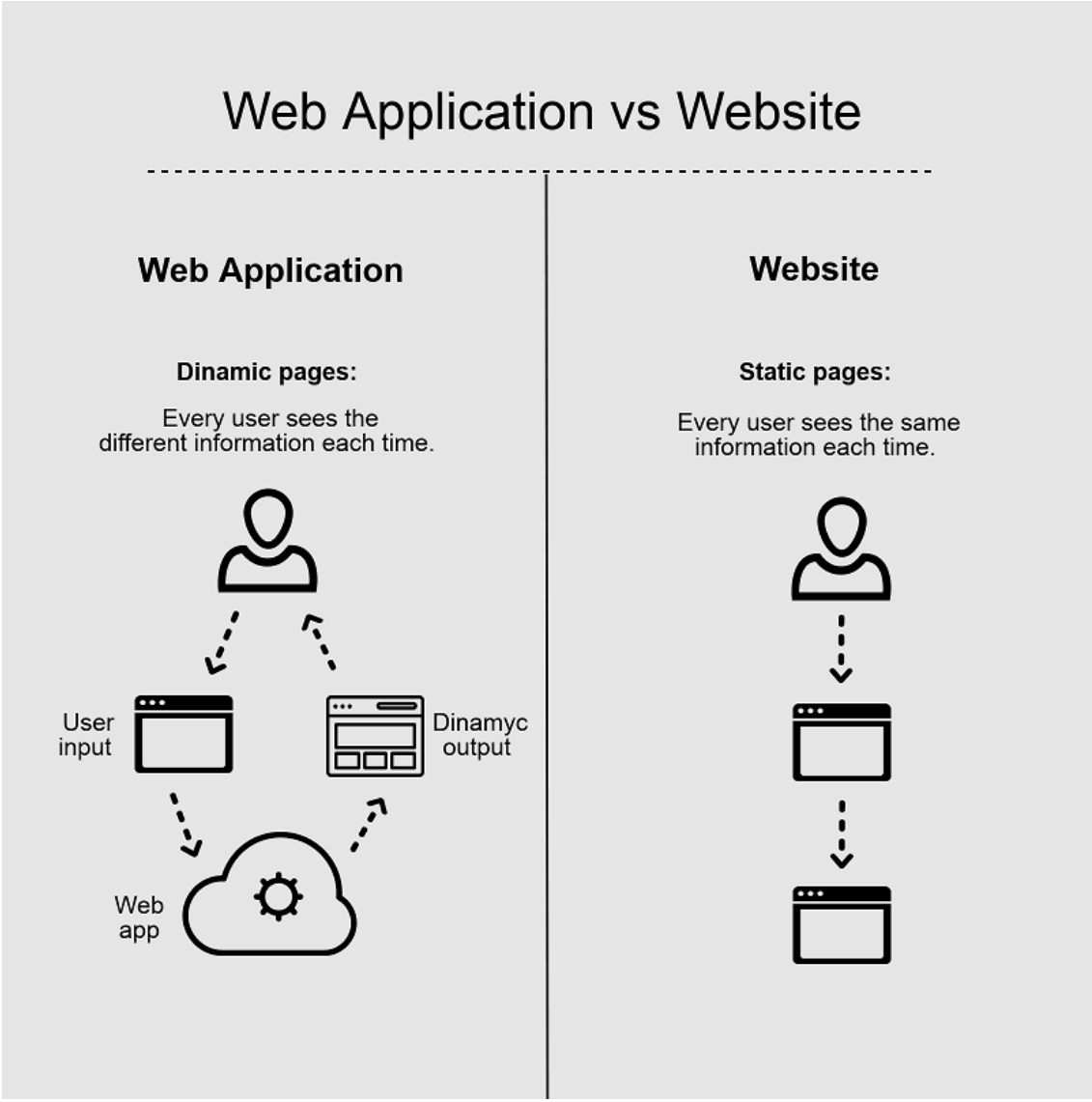
Program komputerowy, który pracuje na serwerze i komunikuje się poprzez sieć komputerową z hostem użytkownika komputera. W tym celu wykorzystuje się przeglądarkę internetową użytkownika, będącego interaktywnym klientem aplikacji internetowej, pełniąc funkcję interfejsu użytkownika aplikacji internetowej.



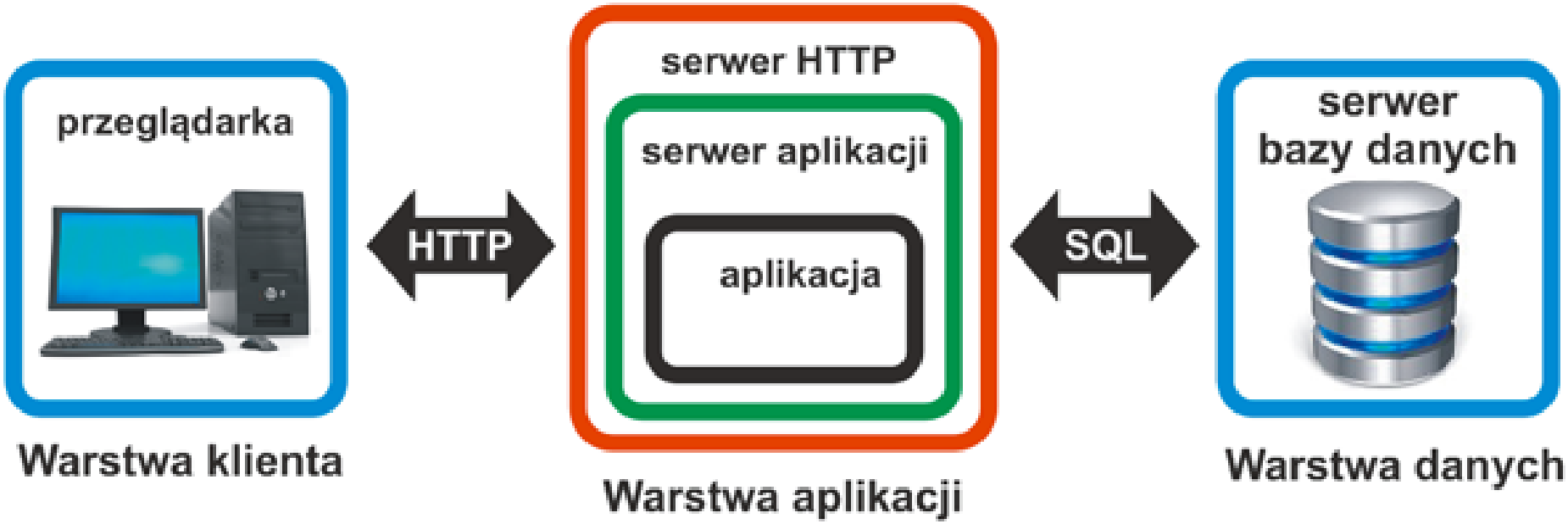
Aplikacja webowa a strona www

Aplikacja webowa ma charakter interaktywny, w przeciwieństwie do strony www, która z założenia ma charakter informacyjny. Na stronie nie wykonasz żadnych działań, poza przeklikiwaniem się na kolejne strony. Aplikacje webowe dostarczają różnych funkcjonalności, a przy tym mają bardziej rozbudowany interfejs.

Aplikacja webowa a strona www



Struktura aplikacji webowej

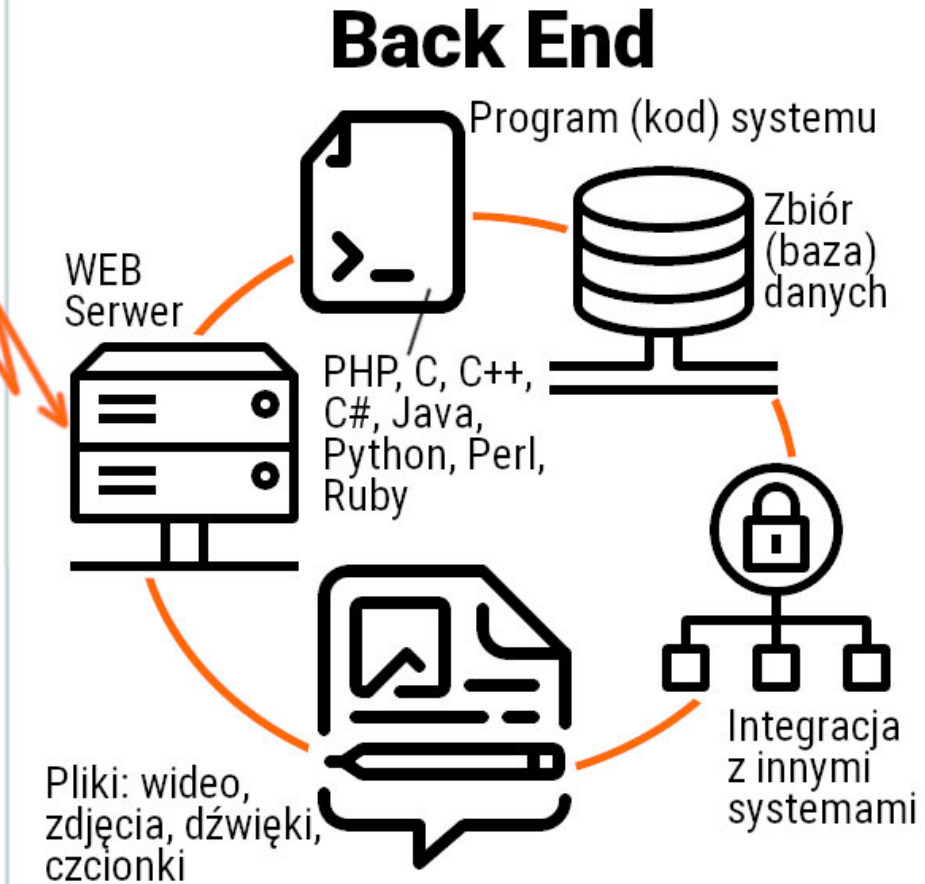


Struktura aplikacji webowej

Front End

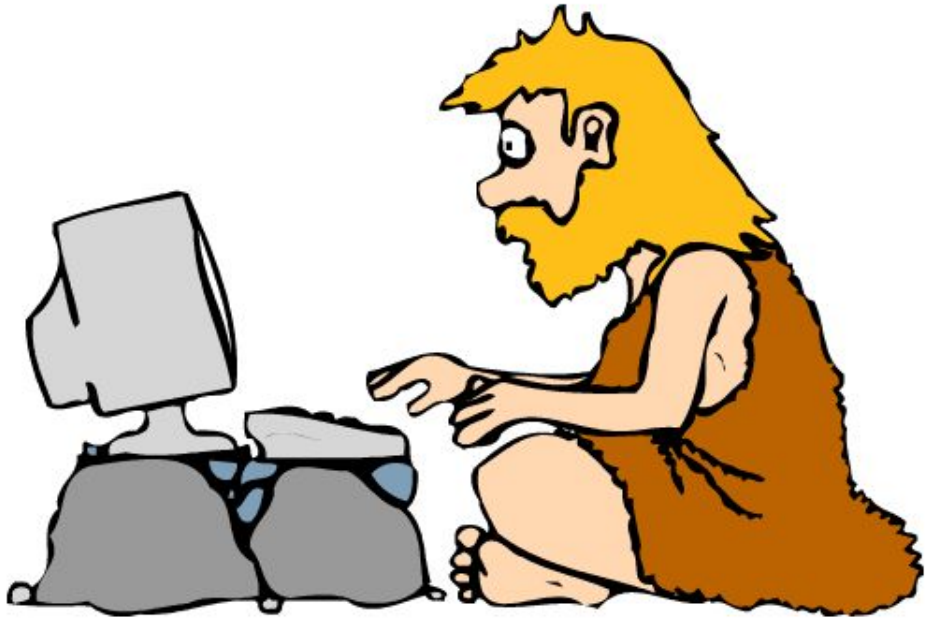
Widok strony, formularze, nawigacja

HTML, CSS, Java Script



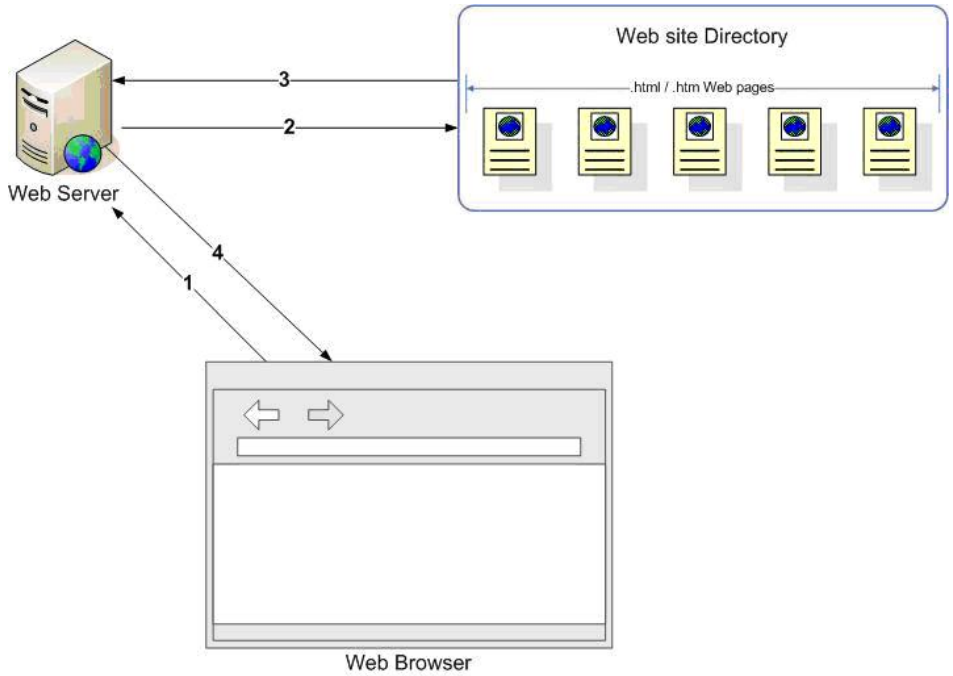
Historia aplikacji webowych

- statyczne strony internetowe
- SSI (Server Side Includes)
- CGI (Common Gateway Interface)
- moduły mod_* - interpretery języków skryptowych, np. mod_perl, mod_php czy mod_python
- PHP
- Java Server Pages (JSP)
- ASP, ASP.NET
- frameworki webowe



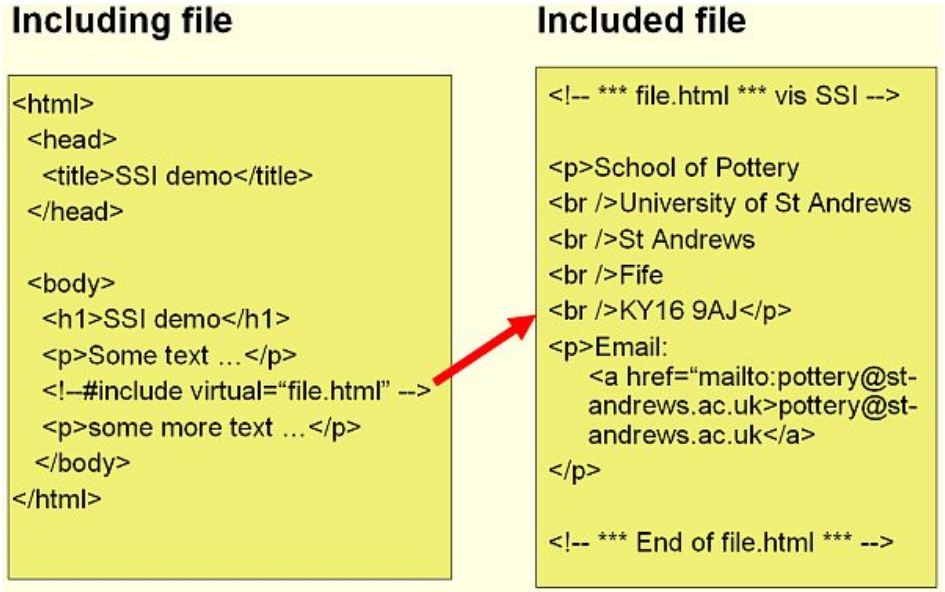
Statyczne strony

- Tworzone przez ludzi
- Zapisywane jako pliki na serwerze
- Agent wysyła żądanie strony
- Serwer ładuje plik z dysku i wysyła go
- Bardzo szybki mechanizm



- Możliwość łatwego cachowania na serwerze albo na serwerach proxy
- Każda zmiana musi być dokonana przez człowieka przez modyfikację pliku HTML

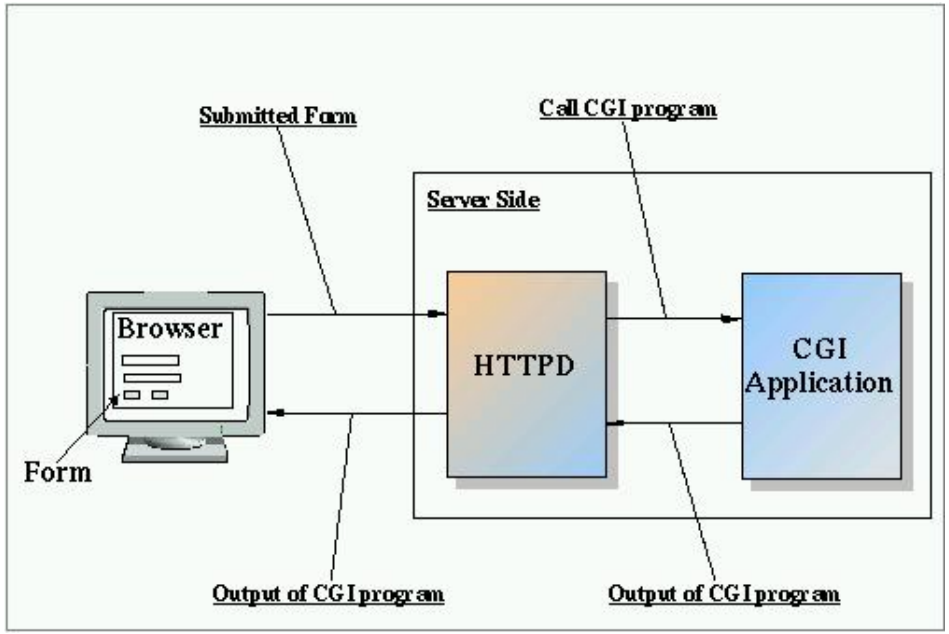
SSI



- Server Side Includes
- Małe fragmenty kodu wbudowane w HTML
- Plik HTML musi mieć prawa wykonywania
- Możliwość wywołania zawnętrzengo programu
- Ograniczony zestaw funkcjonalności
- Ryzyko związane z bezpieczeństwem

CGI

- Common Gateway Interface
- Protokół komunikacji pomiędzy serwerem HTTP a zewnętrznym programem
- Serwer przekazuje żądanie agenta do zewnętrznego programu



- Zewnętrzny program odsyła odpowiedź, która jest zwracana do agenta
- Jedno żądania — jeden program jest uruchamiany
- Problem z serwowanie wielu stron przez jeden program
- Ograniczona rola serwera — tylko przekazuje odpowiedź od programu CGI

Inne technologie

- mod_* in Apache
- PHP
- Java Server Pages (JSP)
- ASP and ASP.NET
- ...

Framework

framework = szkielet do budowy aplikacji

Cechy frameworku:

- odwrócone sterowanie (przepływ sterowania narzucany przez framework, a nie przez użytkownika, reguła Hollywood: nie dzwoń do nas, my zadzwonimy do Ciebie)
- domyślne zachowanie (domyślna konfiguracja, która musi być użyteczna i dawać sensowny wynik)
- rozszerzalność (komponenty frameworka rozszerzalne przez programistę)
- zamknięta struktura wewnętrzna (programista może rozbudowywać framework, ale nie poprzez modyfikację domyślnego kodu)

Framework webowy

- framework, który wspiera tworzenie aplikacji webowych i/lub usług sieciowych (web services)
- ma na celu zmniejszenie narzutu związanego z typowmi mechanizmami występującymi w aplikacjach sieciowych, np. dostęp do bazy danych, interfejs użytkownika, zarządzanie sesją, itp.
- achitektura większości frameworków webowych oparta jest na wzorcu projektowym Model-Widok-Kontroler

Typowe cechy frameworków webowych

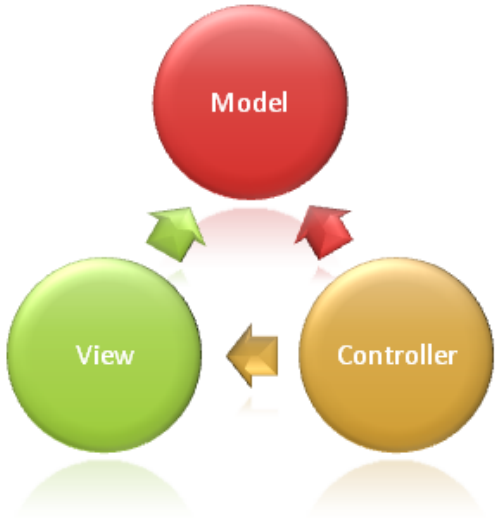
- system szablonów www (web template)
- caching
- wsparcie mechanizmów bezpieczeństwa
- wsparcie dostępu do bazy danych i odwzorowanie relacyjno-obiektowe (ORM)
- mapowanie adresów URL
- wsparcie AJAX
- automatyczna konfiguracja
- wsparcie web service'ów

Wzorzec projektowy MVC

Wprowadzenie

MVC = Model-View-Controller = Model-Widok-Kontroler

- architektoniczny wzorzec projektowy
- organizowanie struktury aplikacji posiadających graficzne interfejsy użytkownika
- może być także traktowany jako złożony wzorzec wykorzystujący idee wzorców prostych takich, jak Obserwator, Strategia, Kompozyt



Rys historyczny

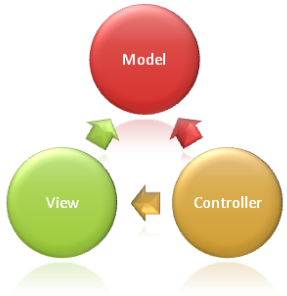
- zaprojektowany w 1979 roku przez norweskiego programistę Trygve Reenskaug pracującego w laboratoriach PARC (Palo Alto Research Centre) firmy Xerox
- pierwotnie MVC został wykorzystany do zaprojektowania interfejsu użytkownika w języku Smalltalk
- zastosowanie MVC w aplikacjach internetowych dokonał Sun na potrzeby Javy
- najbardziej znana implementacja MVC powstała dla Javy, projekt Struts stworzony przez Apache Jakarta Project

Główne założenia

- trzy główne rodzaje aktywności typowej aplikacji: struktury danych, algorytmy przetwarzania, kanały komunikacyjne
- W MVC dla każdego rodzaju aktywności jest jawnie wydzielana osobna część
- model służy do reprezentowania danych przetwarzanych w programie
- widok służy do generowania wyjścia i prezentowania wyników
- kontroler służy do obsługi wejścia
- każda z części MVC ma wyłączność na zarządzanie swoją częścią procesu

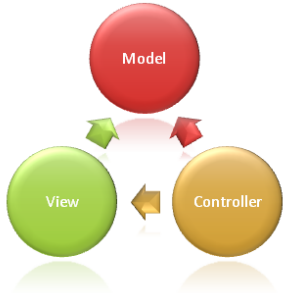
Model

- odpowiada za tzw. logikę biznesową (funkcjonalność związana ze sposobem, w jaki aplikacja przechowuje dane)
- jedyna część aplikacji, która przechowuje dane w sposób trwały
- różne sposoby przechowywania danych (baza danych, pliki tekstowe, Web Services, itd.)
- szczegóły implementacyjne związane ze sposobem przechowywania danych ukryte przed resztą aplikacji
- model dostarcza innym komponentom spójnego interfejsu do przetwarzania danych
- model aktywny, model pasywny



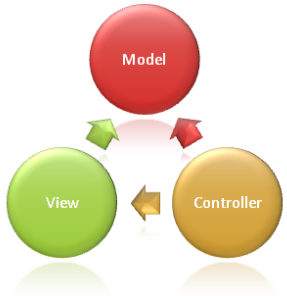
Model (cd)

- jest samodzielny — do poprawnej pracy nie wymaga obecności dwóch pozostałych części
- ma postać klas udostępniających interfejs programistyczny do manipulacji strukturami danych oraz wykonywania pewnych akcji
- dobrym sposobem implementacji modelu jest utworzenie oddzielnej klasy dla każdego logicznego obiektu (np. użytkownicy, artykuły, koszyk)
- najczęściej model przechowuje informacje w bazie danych
- dobrze zaprojektowany model powinien być przenośny



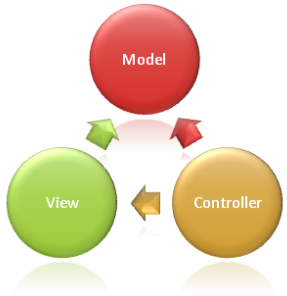
Widok

- odpowiada za prezentację, wyświetlanie danych użytkownikom
- może prezentować wyniki w postaci HTML, XML, WML, obrazków, plików PDF, itd.
- wykorzystuje model do pobrania danych, które będą wyświetlone (tworzy instancje klas modeli i wywołuje z nich metody)
- nie powinien zmieniać stanu aplikacji
- może być implementowany w postaci wielu klas, po jednej na każdy zwracany element
- w dobrze zaprojektowanej aplikacji widok powinien być "wymienny"

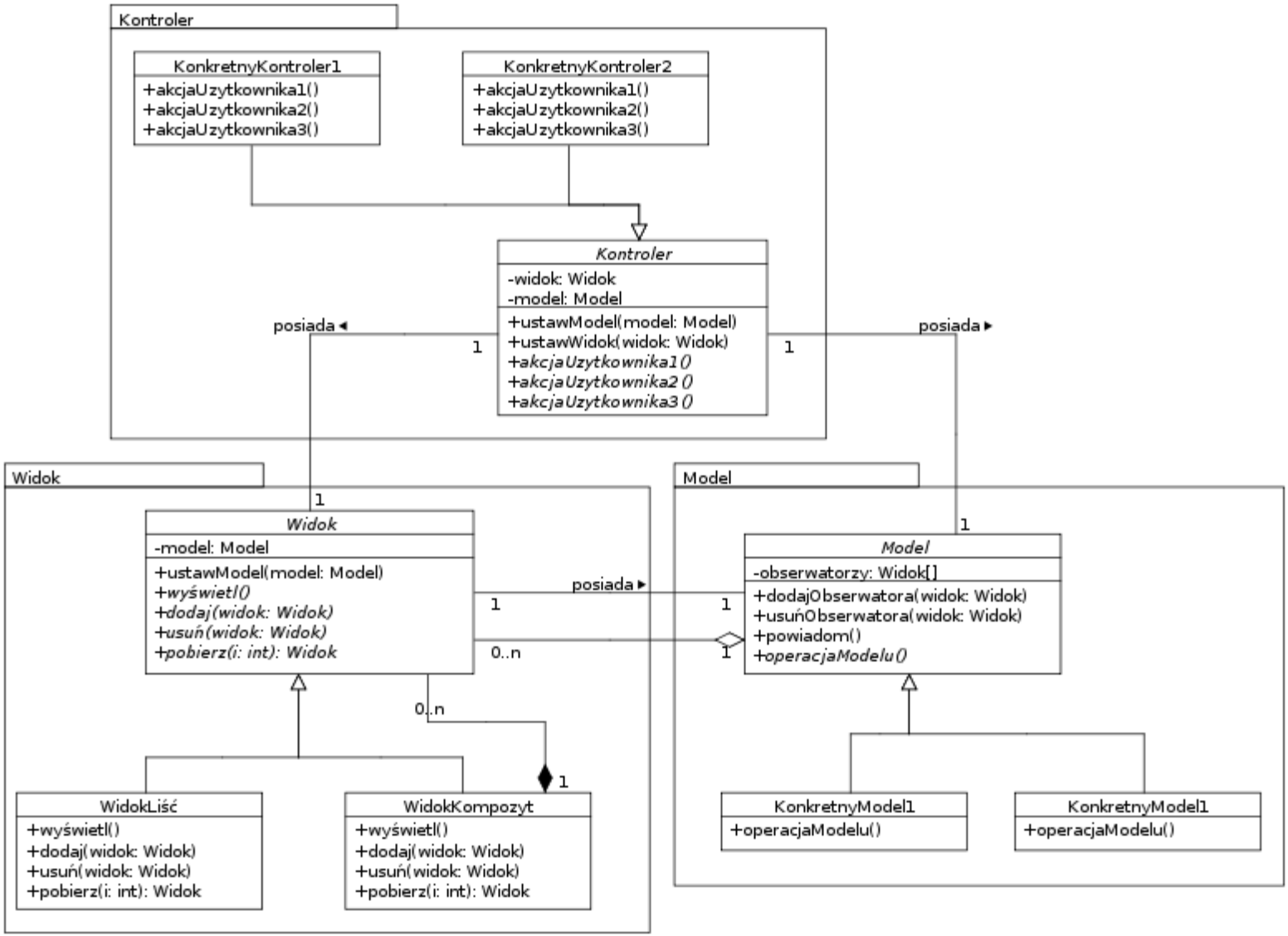


Kontroler

- zadaniem kontrolera jest odbiór, przetworzenie oraz analiza danych wejściowych
- może zmienić stan modelu, odświeżyć widok, przełączyć sterowanie na inny kontroler
- może istnieć wiele kontrolerów, w danym momencie tylko jeden z nich steruje aplikacją
- kontroler posiada bezpośrednie wskazania na modele i widoki, z którymi współpracuje
- jest zasadniczą częścią każdej biblioteki implementującej MVC
- posiada akcje, tj. czynności wykonywane przez aplikację i określające jaki widok należy wyświetlić



MVC jako wzorzec złożony



Kompozyt (widok)

Obserwator (model i widok)

Strategia (widok i kontroler)

Metoda wytwórcza (kontroler)

Dekorator (widok)

Zalety używania MVC

modyfikowalność

- brak zależności modelu od widoków
- łatwiejsza rozbudowa widoków
- dzięki oddzieleniu kodu od wyglądu aplikacji pomagamy w pracy grafikowi

bezpieczeństwo

- podejście obiektowe i hermetyzacja
- oddzielenie klas łączących się z bazą od reszty aplikacji

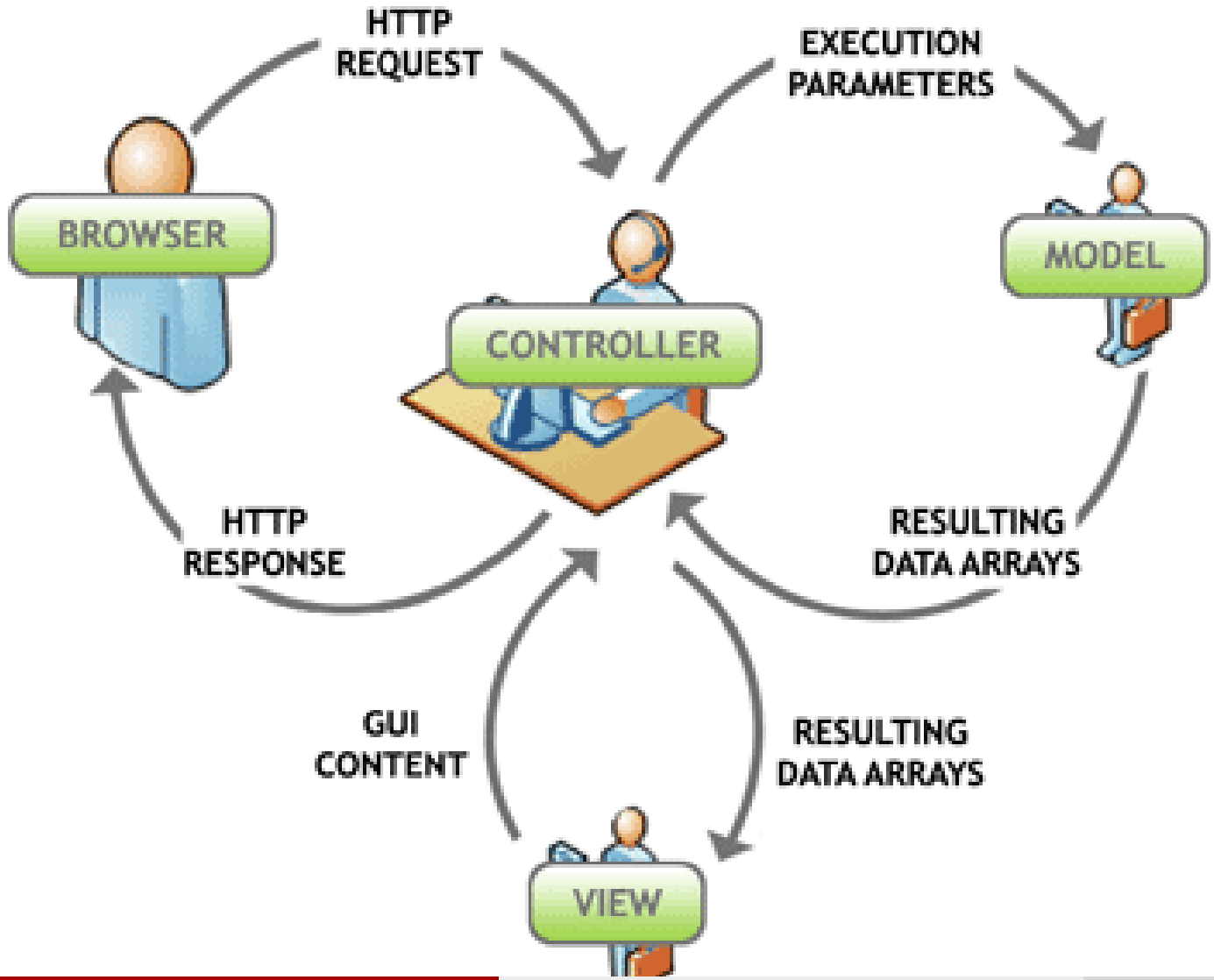
Wady używania MVC

- złożoność - dodatkowa warstwa abstrakcji oraz dodatkowe sposoby interakcji, trudniejsze debugowanie
- kosztowne zmiany modelu - jeżeli interfejs modelu ulega częstym zmianom, oznacza to konieczność poprawiania wszystkich korzystających z niego widoków
- trudne testowanie widoków - testowanie złożonych interfejsów użytkownika uważane jest za zadanie trudne

Framework webowy MVC

- MVC idealnie wpasowuje się w specyfikę zastosowań w aplikacjach sieciowych
- każde żądanie nowej strony jest interakcją użytkownika z systemem i musi zostać jakoś przetworzone (kontroler), większość aplikacji musi przechowywać dane w sposób trwały - najczęściej w bazie danych (model), dane te muszą zostać wyświetlone w jakiś sposób (widok)
- brak aktywnych modeli wynikający z zasady działania protokołu HTTP

Obsługa żądania



Obsługa żądania (cd)

Użytkownik żąda wyświetlenia określonego widoku

1. Żądanie użytkownika (żądanie GET od przeglądarki) trafia do kontrolera
2. Kontroler określa, który widok jest odpowiedni dla tego żądania, tworzy obiekt widoku i przekazuje mu sterowanie
3. Widok tworzy potrzebne mu klasy modelu i prosi model o podanie niezbędnych danych
4. Model wysyła zapytanie do bazy danych
5. Baza danych zwraca modelowi odpowiednie dane
6. Model zwraca dane widokowi
7. Widok formatuje dane i przekazuje odpowiedź kontrolerowi
8. Kontroler wysyła odpowiedź użytkownikowi

Obsługa żądania (cd)

Użytkownik wysyła dane i oczekuje zmiany stanu aplikacji

1. Żądanie użytkownika (na ogół żądanie POST) trafia do kontrolera
2. Kontroler określa, że obsłużenie żądania wymaga wykonania akcji, uruchamia akcję i przekazuje sterowanie
3. Akcja sprawdza dane zawarte w POST, po czym tworzy potrzebne jej klasy modelu i prosi o zmianę danych
4. Model zmienia zawartość bazy danych
5. Akcja informuje kontroler o zakończeniu przetwarzania i podaje, jaki widok należy wyświetlić
6. Kontroler tworzy odpowiedni obiekt widoku i przekazuje sterowanie
7. Widok tworzy potrzebne mu klasy modelu i prosi go o podanie danych
8. Model wysyła zapytanie do bazy danych
9. Baza danych zwraca modelowi odpowiednie dane
10. Model zwraca dane widokowi
11. Widok formatuje dane i przekazuje odpowiedź kontrolerowi
12. Kontroler wysyła odpowiedź użytkownikowi

Django i Ruby on Rails

Inspiracja

- Duże aplikacje webowe mogą być skomplikowane
  - Zarządzanie użytkownikami
  - Wiele stron
  - Szablony
  - Podobne fragmenty na wielu stronach
  - Wspólna funkcjonalność
  - Wdrażanie
  - Zarządzanie konfiguracją
- Prowadzi to do konieczności radzenia sobie ze złożonością aplikacji i zapanowania nad tym
- Wiele mechanizmów występuje w większości aplikacji, np. zarządzanie sesjami, użytkownikami, cacheowanie, itd.

Cechy frameworków Django i RoR

- Cross-platform
- Bazują na językach skryptowych wysokiego poziomu
  - możliwość szybkiego przetestowania nowych idei
- Wykorzystanie istniejących standardów i bibliotek
- Convention over configuration
- Ładnie wyglądające adresy URL
- Łatwość rozpoczęcia pracy z frameworkiem
- “Proste zadania powinny być proste, trudne powinny być możliwe do wykonania” — motto z języka Perl

Cechy frameworków Django i RoR, cd

- Open source
- Język skryptowy pozwala zobaczyć wnętrze frameworka
- Ale nie jest to potrzebne w typowych zadaniach
- Możliwość rozszerzenia i udoskonalenia aplikacji, gdy staje się popularna
- Plug-ins, add-ons
- Możliwość rozszerzenia frameworka

Języki wysokiego poziomu

- Python i Ruby są językami skryptowymi
- Posiadają wiele wbudowanych typów danych: integer, float, boolean, string
- Wbudowane bardziej złożone typy danych: sets, dictionaries, tuples
- Wiele bibliotek dostarczających niezbędnych funkcjonalności
- Zaawansowane techniki programowania:
  - Iteratory
  - Anonimowe metody
  - Introspekcja

Ukryta równoległość

- Serwer może obsługiwać wiele żądań w tym samym czasie
- Programista nie musi się martwić o równoległość
- Threads/processes są zarządzane przez serwer
- Programista nie powinien korzystać z zasobów udostępnionych przez różne wątki

Duck-typing

"Jeśli chodzi jak kaczka, pływa jak kaczka i kwacze jak kaczka, to musi być kaczka"

- Rozpoznawanie typu obiektu nie na podstawie deklaracji, ale przez badanie metod udostępnionych przez obiekt
- Umożliwia łatwe wstrzykiwanie obiektów z dodatkowymi funkcjonalnościami

Konwencja a konfiguracja

- Frameworki są raczej złożone
  - Konfiguracja bazy dnaych
  - Katalogi
  - Zarządzanie zawartością
- To zwiększa komplikację plików konfiguracyjnych i kod, który zarządza wszystkimi opcjami konfiguracji
- Django i Ruby on Rails stosują podejście “convention over configuration”
- Większość opcji konfiguracyjnych ma ustawione domyślne rozsądne wartości
- Jeśli chce się je zmienić, że jest to możliwe, ale nie zalecane
- Konwencja oznacza, że większość aplikacji ma podobne opcje i różnią się tym, co jest ważne

Główne zasady projektowania

- Jakość i czytelność kodu
- Loose coupling
- Don’t Repeat Yourself (DRY)
- Failing on errors — nigdy nie ukrywaj błędów

Ruby on Rails

- początek: 2003, David Heinemeier Hansson, wydany w 2004
- język programowania: Ruby
- ostatnie wydanie: 8.0.1 (13 grudnia 2024)
- http://rubyonrails.org/
- przykłady aplikacji: Basecamp, GitHub, Shopify, Airbnb, Twitch, SoundCloud, Hulu, Zendesk, Square, Cookpad.

Django

- początek: 2003, Lawrence Journal-World, wydany w 2005
- język programowania: Python
- wzorzec projektowy MVT (Model-View-Template), podobny do MVC
- automatycznie generowany i kompletny panel administracyjny
- ostatnie wydanie: 5.1.6 (5 lutego 2025)
- http://www.djangoproject.com/
- przykłady aplikacji: Public Broadcasting Service, Pinterest, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, Nextdoor