ASP.NET MVC

Model-View-Controller

ASP.NET MVC

ASP.NET MVC – platforma aplikacyjna do budowy aplikacji internetowych opartych na wzorcu Model-View-Controller (MVC) oparta na technologii ASP.NET. Wzorzec architektoniczny MVC oraz jego modyfikacje są powszechnie wykorzystywane w wielu innych popularnych platformach aplikacyjnych.

Główną zaletą zastosowania tego wzorca projektowego jest przede wszystkim lepsza separacja warstw aplikacji (np. prezentacji i logiki biznesowej) kosztem wprowadzenia dodatkowej złożoności w architekturze aplikacji. Aplikacje zbudowane przy użyciu tego wzorca są również zazwyczaj łatwiejsze do rozbudowy i utrzymywania oraz bardziej przyjazne w testowaniu przy użyciu testów jednostkowych i podejścia Test-driven Development (TDD).

Model reprezentuje stan aplikacji będący zazwyczaj odzwierciadleniem wybranej części problematyki tworzonego rozwiązania.

Kontroler obsługuje interakcje oraz zmiany stanu modelu i przekazuje te informacje do warstwy prezentacji aplikacji, czyli widoku. Widok natomiast pobiera informacje przekazywane przez kontroler i uaktualnia odpowiednio interfejs użytkownika.

Widok

Widok jest odpowiedzialny za prezentację danych w obrębie graficznego interfejsu użytkownika. Może składać się z podwidoków zarządzających mniejszymi elementami składowymi. Widoki posiadają bezpośrednie referencje do modeli, z których pobierają dane, gdy otrzymują od kontrolera żądanie odświeżenia. Widoki mogą także modyfikować stan modelu, jeśli dana modyfikacja dotyczy sposobu prezentacji danych[5].

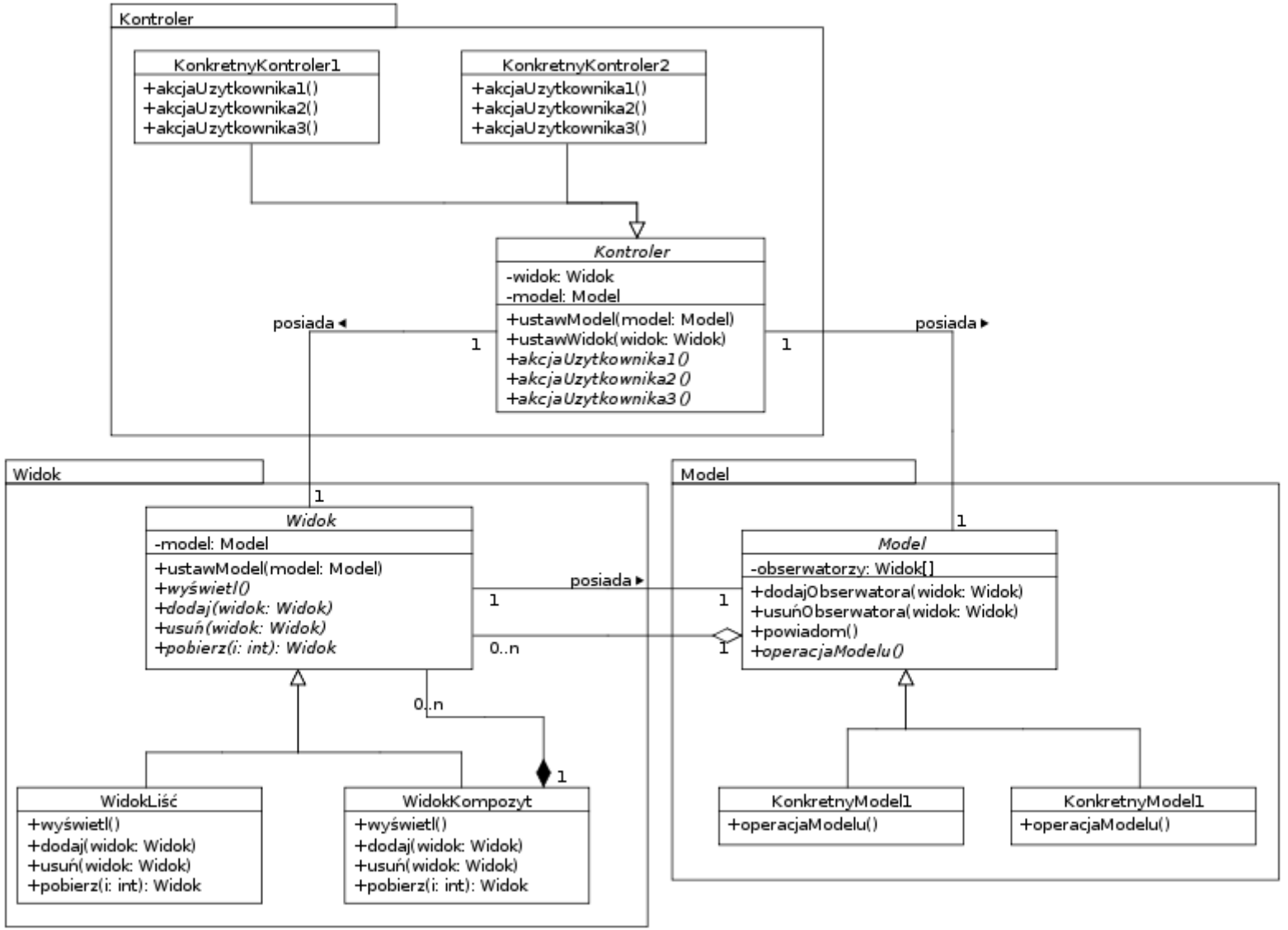
Kontroler

Zadaniem kontrolera jest odbiór, przetworzenie oraz analiza danych wejściowych od użytkownika. W typowej aplikacji źródłami danych wejściowych będą klawiatura i mysz. Po przetworzeniu odebranych danych kontroler może wykonać następujące czynności:

- zmienić stan modelu,
- odświeżyć widok,
- przełączyć sterowanie na inny kontroler.

Każdy kontroler posiada bezpośrednie wskazania na określone modele i widoki, z którymi współpracuje, a jednocześnie w aplikacji może istnieć wiele kontrolerów. W danym momencie tylko jeden z nich steruje aplikacją.

MVC


Kontroler
KonkretnyKontroler1
+akcjaUzytkownika1()
+akcjaUzytkownika2()
+akcjaUzytkownika3()
KonkretnyKontroler2
+akcjaUzytkownika1()
+akcjaUzytkownika2()
+akcjaUzytkownika3()
Kontroler
-widok: Widok
-model: Model
+ustawModel(model: Model)
+ustawWidok(widok: Widok)
+akcjaUzytkownika1()
+akcjaUzytkownika2()
+akcjaUzytkownika3()
posiada
1
1
posiada
Widok
Model
1
Widok
-model: Model
+ustawModel(model: Model)
+wyświetl()
+dodaj(widok: Widok)
+usuń(widok: Widok)
+pobierz(i: int): Widok
posiada
1
0..n
1
Model
-obserwatorzy: Widok[]
+dodajObserwatora(widok: Widok)
+usuńObserwatora(widok: Widok)
+powiadom()
+operacjaModelu()
1
1
0..n
1
WidokLiść
+wyświetl()
+dodaj(widok: Widok)
+usuń(widok: Widok)
+pobierz(i: int): Widok
WidokKompozyt
+wyświetl()
+dodaj(widok: Widok)
+usuń(widok: Widok)
+pobierz(i: int): Widok
KonkretnyModel1
+operacjaModelu()
KonkretnyModel1
+operacjaModelu()

Wady i zalety

Zalety:

- Brak zależności modelu od widoków — model jest niezależny od widoków, dlatego w aplikacji może współistnieć wiele widoków prezentujących te same dane na różne sposoby
- Łatwiejsza rozbudowa widoków — interfejs użytkownika oraz warstwa prezentacji zmieniają się o wiele częściej niż logika biznesowa aplikacji. Ponieważ obie te części są oddzielone, można łatwo dodawać oraz modyfikować istniejące widoki bez wpływu na kluczową część systemu

Wady:

- Złożoność — implementacje MVC wprowadzają dodatkową warstwę abstrakcji oraz dodatkowe sposoby interakcji, czyniąc w ten sposób aplikację potencjalnie trudniejszą do debugowania
- Kosztowne zmiany modelu — ponieważ model nie jest zależny od widoku, programiści rozwijający tę część nie muszą przejmować się zależnościami w przeciwnym kierunku. Jeżeli interfejs modelu ulega częstym zmianom, oznacza to konieczność poprawiania wszystkich korzystających z niego widoków
- Trudne testowanie widoków — widoki są zależne od modeli, a ponadto zawierają własną, dodatkową logikę. Testowanie złożonych interfejsów użytkownika uważane jest za zadanie trudne