7 Kompozycja aplikacji

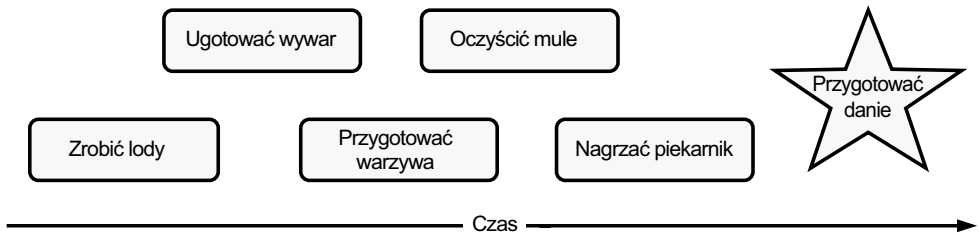
W rozdziale:

- Komponowanie aplikacji konsolowych
- Komponowanie aplikacji Universal Windows Programming (UWP)
- Komponowanie aplikacji ASP.NET Core MVC

Gotowanie pysznego posiłku składającego się z kilku dań jest wymagającym przedsięwzięciem, w szczególności jeśli gotujący chce również brać udział w samym jedzeniu. Nie można jeść i gotować w tym samym czasie, a wiele dań wymaga przygotowania tuż przed podaniem. Zawodowi kucharze wiedzą, jak poradzić sobie z wieloma z tych wyzwań. Używają wielu trików, a także stosują ogólną zasadę *mise en place*, co można luźno przetłumaczyć z francuskiego na *wszystko na miejscu*[1]. Wszystko, co może być przygotowane zawczasu jest, no cóż, przygotowane zawczasu. Umyte i pokrojone warzywa, pokrojone mięso, ugotowany wywar, nagrzane piekarniki, ułożone potrzebne narzędzia itd.

[1] Francuska wymowa: miz en plas.

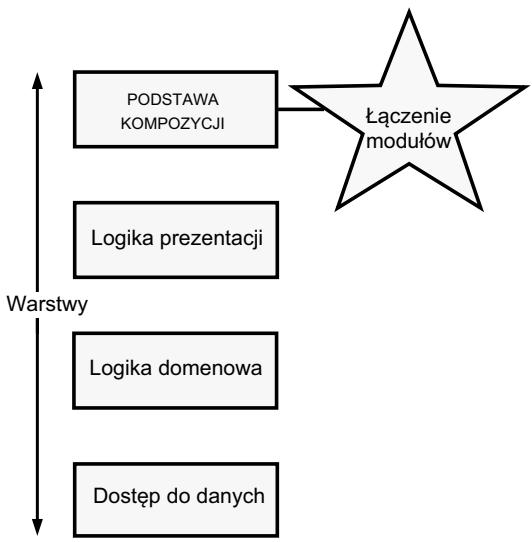
Jeśli lody są częścią deseru, można je zrobić dzień wcześniej. Jeśli pierwsze danie składa się z muli, można je oczyścić zawczasu. Nawet tak delikatny składnik jak sos berneński może zostać przygotowany do godziny wcześniej. Kiedy goście są gotowi na jedzenie, tylko ostatnie czynności wymagają wykonania: odgrzanie sosu, wysmażenie mięsa itd. W wielu przypadkach końcowa kompozycja dania nie zajmuje więcej niż od 5 do 10 minut. Rysunek 7.1 ilustruje ten proces.



Rysunek 7.1. *Mise en place* oznacza przygotowanie wszystkich komponentów posiłku wcześniej, tak by końcowa kompozycja posiłku mogła być wykonana tak szybko i bez wysiłku, jak jest to możliwe

Zasada *mise en place* jest podobna do tworzenia luźno powiązanej aplikacji przy wykorzystaniu DI. Wszystkie wymagane komponenty można napisać wcześniej i połączyć je wtedy, gdy jest ku temu potrzeba.

Tak jak w przypadku wszystkich analogii, tej możemy użyć tylko w takim stopniu, a potem są już tylko same różnice. W gotowaniu przygotowanie i kompozycja są oddzielone czasem, a w tworzeniu aplikacji separacja pojawia się na poziomie modułów i warstw. Rysunek 7.2 pokazuje, jak połączyć elementy W PODSTAWIE KOMPOZYCJI.



Rysunek 7.2. PODSTAWA KOMPOZYCJI składa się ze wszystkich niezależnych modułów aplikacji

W czasie uruchomienia najpierw pojawia się KOMPOZYCJA OBIEKTOWA. Kiedy graf obiektów zostaje zbudowany, KOMPOZYCJA OBIEKTOWA kończy się, a elementy składowe przejmują kontrolę. W tym rozdziale skupimy się na PODSTAWIE KOMPOZYCJI kilku frameworków aplikacji. W porównaniu do *mise en place* KOMPOZYCJA OBIEKTOWA nie dzieje się tak późno, jak to możliwe, ale dzieje się w miejscu, gdzie integracja różnych modułów jest wymagana.

DEFINICJA KOMPOZYCJA OBIEKTOWA jest działaniem budowania hierarchii pokrewnych komponentów. Ta kompozycja zachodzi wewnątrz PODSTAWY KOMPOZYCJI.

KOMPOZYCJA OBIEKTOWA jest podstawą DI i do tego jedną z najłatwiejszych części do zrozumienia. Czytelnicy na pewno wiedzą, jak ją przeprowadzić, ponieważ kompozycja następuje przy każdorazowym stworzeniu instancji obiektów zawierających inne obiekty.

W sekcji 4.1 omawialiśmy podstawy tego, jak i kiedy budować aplikację. Ten rozdział nie powiela tych informacji. Zamiast tego chcemy pomóc w adresowaniu niektórych z wyzwań, które mogą pojawić się w trakcie tworzenia obiektów. Wyzwania te nie biorą się z samej KOMPOZYCJI OBIEKTOWEJ, ale z frameworków aplikacji, w których się pracuje. Kwestie mają tendencję do bycia konkretnymi dla każdego frameworka i tak samo jest z ich rozwiązaniami. Z naszego doświadczenia wiem, że wyzwania te stanowią jedne z największych przeszkód przed udanym zastosowaniem DI, a więc skupimy się właśnie na nich. Przez to rozdział ten stanie się mniej teoretyczny, a bardziej praktyczny w porównaniu do wcześniejszych.

WAŻNE Jeśli ktoś chce przeczytać jedynie o zastosowaniu DI do swojego wybranego frameworka, może pominąć tę sekcję w tym rozdziale. Każda sekcja jest stworzona tak, by można je było czytać niezależnie.

Łatwo jest stworzyć całą hierarchię ZALEŻNOŚCI aplikacji, kiedy ma się całkowitą kontrolę nad cyklem życia aplikacji (tak jak w przypadku aplikacji konsolowych). Ale niektóre z frameworków w .NET (na przykład ASP.NET Core) angażują ODWRÓCENIE STEROWANIA, co może czasami utrudnić zastosowanie DI. Zrozumienie SZWÓW każdego z frameworków jest kluczowe w zastosowaniu DI dla tego konkretnego frameworka. W tym rozdziale zbadamy, jak implementować PODSTAWĘ KOMPOZYCJI w najczęściej używanych frameworkach .NET Core.

Każdą sekcję rozpoczniemy od ogólnego wprowadzenia do zastosowania DI w konkretnym frameworku, a następnie pokażemy rozbudowany przykład oparty na przykładzie e-commerce, który pojawia się w większości tej książki. Zaczniemy od najprostszego frameworka odnośnie do zastosowania DI, a potem stopniowo przejdziemy do tych bardziej skomplikowanych. Jak do tej pory najprostszym w aplikowaniu DI jest typ aplikacji konsolowej, a więc omówimy go teraz.

WAŻNE Z jednej strony, niektóre ze starych frameworków .NET (na przykład PowerShell lub starsze wersje ASP.NET Web Forms) są zdecydowanie wrogimi środowiskami do zastosowania DI. Z drugiej strony, nowsze frameworki .NET Core są bardziej przyjazne dla DI. W tej książce głównie skupiamy się na tych nowszych. Jeśli ktoś chce się dowiedzieć, jak aplikować DI do ASP.NET MVC, Web Forms, WCF, WPF lub PowerShell, może znaleźć to w cyfrowej wersji pierwszej edycji tej książki; jest ona dostępna wraz z zakupem tej edycji. Rozdział 7 omawia każdy Framework bardzo szczegółowo.