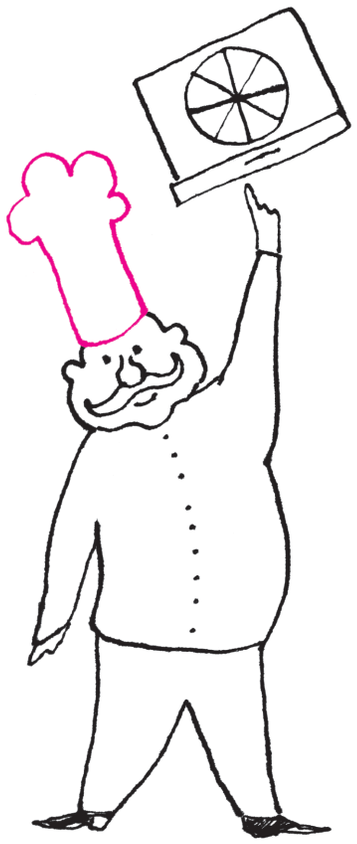
Ze świata USOS. Część 6 – Jak szybko przyrządzić smaczną aplikację

Łukasz KARNIEWSKI[*]

Wyobraźmy sobie, że wchodzimy do lokalu znanej sieci pizzerii i zamawiamy swoją ulubioną pizzę. Kelner czym prędzej biegnie z naszym zamówieniem do kuchni i już po chwili rozpoczyna się proces wytwórczy. Z czego składa się taka pizza? Zasadniczo jest to płaskie ciasto w kształcie koła, posmarowane sosem pomidorowym, na którym układa się różne dodatki, a całość posypuje serem i wstawia na jakiś czas do pieca. Trudno chyba o prostszy przepis. Kucharz przystępuje zatem do dzieła, jednak na potrzeby naszej historyjki wyobraźmy sobie, że przygotowuje on naszą pizzę od podstaw. I to samiuteńkich podstaw: mieli trochę ziarna na mąkę do ciasta, przeciera kilka pomidorów na sos, idzie do ogródka po parę warzyw, a do mleczarni po mleko na ser, po drodze zbierając drewno do rozpalenia w piecu... Wiem, wiem, brzmi to absurdalnie i na całe szczęście żaden pozostający przy zdrowych zmysłach kucharz nie zgodziłby się pracować w taki sposób: po realizacji kilku zamówień padłby z przepracowania, a oczekujący przy stolikach klienci z głodu.



Tymczasem w prawdziwym świecie zwykle już po 20 (no, może bliżej 30) minutach pojawia się kelner z naszym wymarzonym daniem. W jaki sposób udaje mu się to tak szybko? Ano dlatego, że zmyślny kucharz spodziewa się naszego zamówienia na pizzę i wszystko to, o czym przed chwilą pisałem, ma już od dawna przygotowane: ciasto wyrobione i podzielone na odmierzone porcje (na pizzę małą, dużą lub średnią), sos gotowy, wszystkie możliwe dodatki pokrojone, ser utarty, piec rozgrzany – jednym słowem ma wszystkie *komponenty* (tzn. składniki) pod ręką i to w ilości wystarczającej na wiele zamówień. Teraz wystarczy tylko wziąć te składające się na wybraną przez klienta pizzę i zbudować z nich gotowy produkt. My się cieszymy z szybkiej obsługi, kucharz się cieszy z szybkiego i łatwego zarobku, wszyscy są zadowoleni.

Uff, to już koniec analogii. Zapytacie teraz pewnie, czy to wszystko ma w ogóle cokolwiek wspólnego z wytwarzaniem oprogramowania? Otóż okazuje się, że w pewnym stopniu ma. Programista, który zabiera się do napisania nowej aplikacji, jest trochę jak nasz kucharz zabierający się do przyrządzenia nowej pizzy. Może postąpić na dwa sposoby: podobnie jak w pierwszym przypadku własnoręcznie zbudować wszystko od zera (i powtarzać tę żmudną pracę dla każdego zamówienia) albo pójść po rozum do głowy i skorzystać z „wyposażonej i zaopatrzonej kuchni" – czyli z *frameworku*.

Co to takiego jest framework? Mówiąc najprościej, jest to uniwersalna platforma programistyczna wielokrotnego użytku, której zadaniem jest przyśpieszenie i ułatwienie procesu tworzenia oprogramowania. Jest to swego rodzaju szkielet, na bazie którego programista jest w stanie w krótkim czasie i przy możliwie niewielkim nakładzie pracy stworzyć zupełnie nową aplikację. W jaki sposób szkielety pomagają osiągnąć ten cel? Przede wszystkim dostarczając gotowe rozwiązania pewnych standardowych zadań, dzięki czemu programista może skupić się na kwestiach niestandardowych, czyli funkcjonalności specyficznej dla danego projektu. Szkielety definiują strukturę aplikacji oraz ogólny mechanizm jej działania, zwykle są też wyposażone w przydatne narzędzia i zaopatrzone w komponenty, z których łatwo buduje się docelowe elementy systemu.

No tak, dużo mądrych zdań, z których trudno cokolwiek zapamiętać, pora więc na przykład.

Załóżmy, że nasz zespół programistyczny ma za zadanie napisać sklep internetowy. Jak w przypadku każdej aplikacji, pracę należy zacząć od ustalenia funkcjonalności. Pierwsze wymagania są oczywiste: ma to być serwis webowy, zatem musi odbierać żądania od przeglądarki i odpowiadać na nie, wyświetlając strony HTML, w międzyczasie zapewne pobierając i zapisując jakieś informacje w bazie danych. Musi umieć rozróżniać użytkowników przeglądających serwis, umożliwić im rejestrację i logowanie, kontrolować ich uprawnienia, pozwolić na zmianę języka tekstów, wysyłać wiadomości e-mail, oferować interfejs administracyjny, uff! Nagle zrobiło się tego bardzo dużo, a przecież nie doszliśmy nawet do funkcjonalności faktycznie związanych ze sklepem, takich jak prezentacja

[*]Instytut Informatyki, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

towaru czy składanie i realizacja zamówień! Jednym słowem wygląda na to, że mamy przed sobą mnóstwo pracy. I to w dużej części tej samej pracy, którą setki innych programistów wykonało już kiedyś przed nami, bo przecież prawie każdy serwis internetowy sięga do bazy, generuje strony HTML, pozwala się zalogować...

I kiedy już jesteśmy pewni, że nie unikniemy nudnego odtwarzania tego, co już dawno wymyślono i zrobiono, ktoś wpada na pomysł wykorzystania frameworku webowego. Eureka! W jednej krótkiej chwili i całkowicie za darmo dostajemy wszystkie elementy typowe dla serwisów internetowych, wśród nich między innymi:

- mechanizm obsługi żądań HTTP,
- mechanizm generowania stron HTML,
- mechanizm zarządzania modelem danych i dostępem do bazy,
- mechanizm obsługi sesji użytkowników,
- obsługę wielojęzyczności i lokalizacji,
- moduł uwierzytelniania i autoryzacji

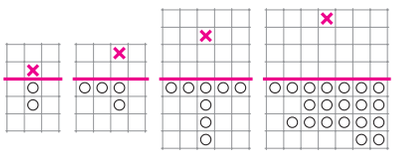
oraz dużo, dużo więcej. Hurra! Już nie musimy ponownie wynajdywać koła! Mając taki zestaw startowy, jesteśmy w stanie w ciągu kilku chwil uruchomić testowo nasz (zupełnie jeszcze „surowy" i „nicpókiconierobiący") serwis. I skupić się na kwestiach naprawdę dla nas istotnych, czyli nad nowatorską funkcjonalnością naszego sklepu internetowego.

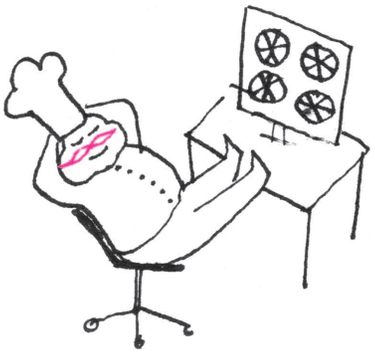
Myślę, że w tym momencie zalety płynące z używania szkieletów do budowy aplikacji powoli stają się jasne, a jeśli nie, to oto one. Po pierwsze, zwiększa się efektywność procesu tworzenia oprogramowania, a zmniejsza znacznie ilość czasu potrzebnego do napisania kodu. Programiści nie tracą czasu na powtórzenie wielu standardowych czynności (nie muszą się nawet tak dobrze znać na siedzących pod maską technicznych aspektach) i mogą poświęcić więcej uwagi biznesowej stronie projektu. Po drugie zwiększa się jakość samego kodu. Szkielety narzucają dobrze przemyślaną wewnętrzną logikę i organizację kodu, dzięki czemu zmniejsza się ryzyko błędów. I wreszcie, szkielety są zwykle dobrze zaprojektowane i przetestowane, dzięki czemu można być pewnym ich niezawodności. Są to więc bardzo solidne fundamenty, na których można bezpiecznie budować złożone aplikacje.

Oczywiście, jak każdy kij, także i ten ma swój drugi koniec. Uniwersalna i elastyczna natura frameworków powoduje, że często ceną za ułatwienie pracy jest obniżenie wydajności aplikacji. Duża złożoność szkieletów może także utrudnić, zwłaszcza początkującym, korzystanie ze wszystkich oferowanych przez nie dobrodziejstw – zwykle trzeba poświęcić sporo czasu oraz przebrnąć przez wiele stron dokumentacji, aby poznać wszystkie tajniki danej platformy.

Korzyści płynące ze stosowania szkieletów aplikacyjnych są jednak nieporównanie większe od potencjalnych wad i dlatego wykorzystanie ich do budowania oprogramowania stało się szeroko stosowaną praktyką. Na chwilę obecną każda popularna technologia samych tylko szkieletów webowych posiada co najmniej kilka. Są to np.: ASP.NET MVC (język ASP.NET), Google Web Toolkit, Spring (Java), Ruby on Rails (Ruby), Django, Flask (Python) czy Zend (PHP). A przecież serwisy internetowe to nie jedyny rodzaj oprogramowania: istnieją też oczywiście szkielety do budowy aplikacji desktopowych (np. Java Foundation Classes, Qt), czy dla urządzeń mobilnych (np. Android Framework).

No, a gdzie w tym wszystkim USOS? Choć niewielu użytkowników zdaje sobie z tego sprawę, na system USOS składa się wiele aplikacji. Najważniejszą z nich jest desktopowa aplikacja, której używają pracownicy administracyjni uczelni, dająca dostęp do centralnej bazy systemu. Ale USOS to także szereg serwisów webowych przeznaczonych dla kandydatów, studentów i nauczycieli – dzięki nim mogą oni sami załatwiać wiele związanych ze studiami spraw i nie zawracać tak często głowy paniom w dziekanacie. Te aplikacje to m.in.: USOSweb, UL (system rejestracji żetonowej), SRS, APD, Ankieter, IRK, USOS API... jak widać, jest ich całkiem sporo (pełną listę można znaleźć na stronie `www.usos.edu.pl/serwisy`). Gdyby każdy z tych serwisów miał powstawać od zera, wiele z nich pewnie nie powstałoby do dziś – to właśnie zastosowanie frameworków pozwoliło stosunkowo niewielkiemu zespołowi programistów na rozwinięcie tylu aplikacji.





Jakiś czas temu programiści webowych serwisów USOS postanowili jeszcze bardziej ułatwić sobie życie. W pewnym momencie okazało się, że kilka właśnie powstających lub mających wkrótce powstać aplikacji używa tej samej technologii (Python + Django) i ma bardzo podobną architekturę, a także współdzieli pewne wymagania dotyczące funkcjonalności. Naturalnie pojawił się pomysł, aby przygotować dla tych aplikacji jakąś wspólną bazę, od której każda z nich mogłaby wyjść i podążyć w swoim kierunku – innymi słowy, napisać dla nich framework. Tak też zrobiono: powstała swego rodzaju nakładka na Django zawierająca szereg modyfikacji, komponentów i modułów gotowych do wykorzystania w nowych serwisach USOS-owych. Znalazły się w niej takie elementy, jak między innymi:

- szablon układu stron i definicja ogólnej, znanej z innych serwisów szaty graficznej,
- mechanizm logowania za pomocą Centralnego Systemu Uwierzytelniania (CAS),
- moduł administracyjny z podstawową funkcjonalnością,
- moduł wyszukiwania osób i wysyłania e-maili.

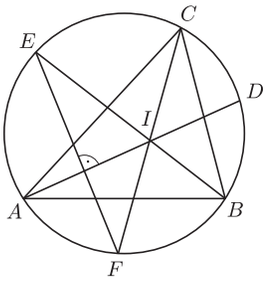
Nowy szkielet zdał egzamin: z jego pomocą jeden programista był w stanie samodzielnie w przeciągu miesiąca napisać całkiem nową aplikację, która posłużyła do przeprowadzenia okresowej ewaluacji pracowników dydaktycznych na Wydziale Matematyki, Informatyki i Mechaniki UW.

Jak wszystkim na pewno świetnie wiadomo, programowanie samo w sobie jest już bardzo ciekawą, choć czasem trudną czynnością. A frameworki sprawiają, że tworzenie złożonych systemów staje się nie tylko sporo łatwiejszym, ale także dużo przyjemniejszym zajęciem.

Zadania

Redaguje Tomasz TKOCZ

M 1414. Niech I będzie środkiem okręgu wpisanego w trójkąt ABC. Półproste AI, BI, CI przecinają okrąg opisany na nim odpowiednio w punktach D, E, F. Udowodnić, że proste AD i EF są prostopadłe.
Rozwiązanie na str. 7



M 1415. Znaleźć wszystkie funkcje f odwzorowujące zbiór liczb rzeczywistych w siebie i spełniające dla każdych liczb rzeczywistych x, y równanie

$$f(xf(y)) = f(xy) + x.$$

Rozwiązanie na str. 4

M 1416. W pewnej szkole jest $2n$ uczniów, $n \geqslant 2$. Każdego tygodnia n uczniów dostaje bilety i jedzie na wycieczkę. Po k tygodniach okazało się, że każdych dwóch uczniów było razem na przynajmniej jednej wycieczce. Udowodnić, że $k \geqslant 6$.
Rozwiązanie na str. 4

Przygotowali Andrzej MAJHOFER i Michał NAWROCKI

F 851. Na transmisyjną siatkę dyfrakcyjną pada prostopadle do jej powierzchni równoległa wiązka światła o długości fali $\lambda = 0{,}633\,\mu\text{m}$ z lasera He-Ne. Za siatką obserwuje się $k = 7$ maksimów dyfrakcyjnych. Ile wynosi stała użytej siatki dyfrakcyjnej?
Rozwiązanie na str. 11

F 852. W pomieszczeniach wypełnionych pyłem węglowym zdarzają się eksplozje spowodowane samozapłonem pyłu. Znaleźć temperaturę końcową po wybuchowym spaleniu węgla w pomieszczeniu o sztywnych ścianach, jeśli wybuch wyczerpał cały tlen zawarty początkowo w powietrzu wypełniającym pomieszczenie. Temperatura początkowa wynosiła $T_0 = 300\,\text{K}$, zaś 21% objętości pomieszczenia wypełniał tlen. Spalaniu węgla do dwutlenku węgla towarzyszy wydzielenie ciepła $Q = 406\,\text{kJ/mol}$. Jaka była wysokość warstwy pyłu węglowego leżącego początkowo na podłodze, jeśli pomieszczenie miało wysokość $H = 2\,\text{m}$? Można przyjąć, że 1 kg pyłu czystego węgla wypełnia objętość 1/2 l.
Rozwiązanie na str. 4