Ze świata USOS. Część 7 – USOS API: komputerowy interfejs użytkownika

Wojciech RYGIELSKI[*]

To, co widzisz, przeglądając strony takie jak Facebook lub Gmail, nazywamy interfejsem użytkownika. Jeden serwis często ma bardzo dużo odmian interfejsów użytkownika, np. Facebook oglądany przez przeglądarkę internetową w telefonie wygląda zupełnie inaczej niż na komputerze. Pewnie o tym wiesz. **Ale czy wiesz, że większość tych serwisów ma jeszcze jeden interfejs użytkownika, zupełnie nieznany, a przeznaczony tylko i wyłącznie dla programistów?**

Problem

Zastanawiałeś się kiedyś, skąd Twój telefon „wie", że dostałeś nową wiadomość e-mail, albo że Twój znajomy właśnie oznaczył Cię na zdjęciu na Facebooku? **Czy gdybyś sam miał napisać podobną aplikację, np. powiadamiającą użytkownika o tym, że wykładowca właśnie wstawił jego ocenę z kolokwium do systemu USOS, to czy wiedziałbyś, jak się do tego zabrać?**

Web scraping (źle!)

Niegdyś, gdy programista spotykał się z podobnym problemem, to często jedynym dostępnym rozwiązaniem było użycie tzw. *web scrapingu* (dosł. „zeskrobywanie z sieci"). Technika ta polega na tym, że Twoja aplikacja co jakiś czas otwiera „po cichu" odpowiednią stronę w Internecie i próbuje „wyłuskać" z niej odpowiednie dane. W tym przypadku byłaby to strona z ocenami w USOSwebie. Wtedy, jeśli aplikacja zauważy, że pojawiła się nowa ocena, to może wyświetlić użytkownikowi odpowiednie powiadomienie.

Web scraping ma jedną (i tylko jedną) zaletę: można go zastosować praktycznie zawsze, czyli również wtedy, gdy system, z którego chcemy dane „wyciągnąć" nie udostępnia żadnego publicznego API (o tym, czym jest API, dowiesz się za chwilę). **Niestety, web scraping ma też ogromnie dużo wad i powinien być używany tylko w ostateczności.**

Zacznijmy od tego, że jest to technika trudna. Znalezienie odpowiedniej informacji na stronie WWW nie wydaje się trudne dla użytkownika oglądającego tę stronę, lecz powiedzenie programowi komputerowemu, żeby „spojrzał w róg tej tabelki", wcale już takie łatwe nie jest. Często programista musi mozolnie pisać setki wierszy kodu, aby dobrze komputerowi wytłumaczyć, gdzie trzeba spojrzeć.

Pojawiają się też inne problemy, np. potrzeba wcześniejszego zalogowania się do systemu. A nawet problemy związane z prawem, gdyż wiele serwisów uważa użycie web scrapingu za naruszenie zasad korzystania z ich usług.

A najgorsze jest to, że nawet jeśli już biedny programista przebrnie przez wszystkie te problemy i opublikuje swoją aplikację, to i tak **taka aplikacja może niespodziewanie przestać działać z całkiem banalnego powodu**. Wystarczy, że projektanci USOSweba postanowią, że ocena będzie lepiej wyglądała w trochę innym miejscu strony internetowej niż dotychczas.

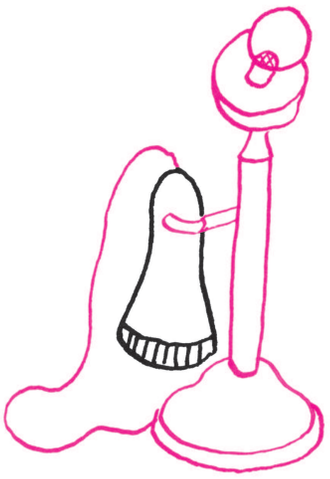
Nagle okazuje się, że nasz program wymaga pilnej poprawki, szef jest wściekły, a my musimy dwa tygodnie wcześniej wrócić z wakacji, aby program naprawić. Czy mogliśmy jakoś tej katastrofy uniknąć?

Kompatybilność wstecz

Powyższy problem jest związany z pojęciem „kompatybilności wstecz". Przeczytaj poniższą definicję dwa razy, bo od tej chwili pojęcie to będzie się przewijać w tym artykule niczym mantra!

Usługa jest kompatybilna wstecz, jeśli już *na zawsze* zachowuje się zgodnie z wcześniej ustalonymi i udokumentowanymi zasadami.

[*]Laboratorium Komputerowe, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

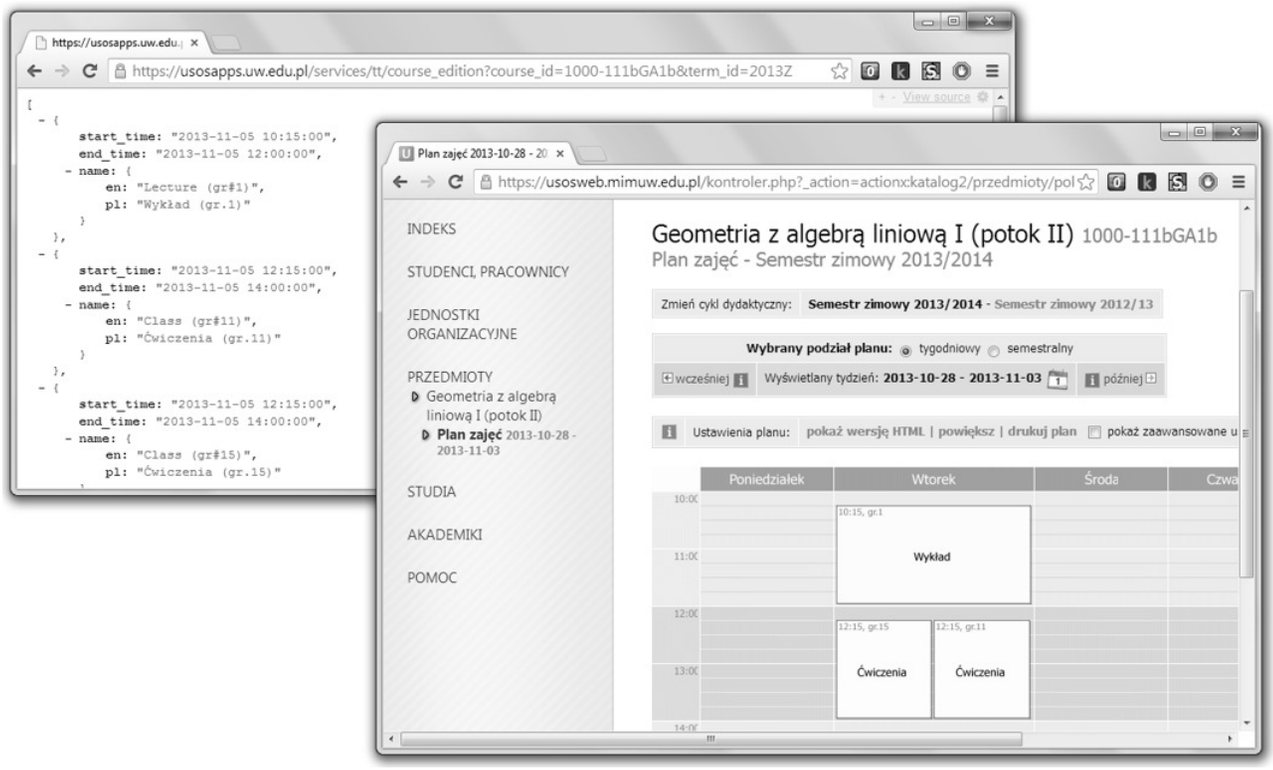


Na przykład, gdybyś przed opublikowaniem swojej aplikacji dogadał się z projektantami USOSweba, że ocena zawsze będzie wyświetlana w tym samym miejscu na stronie, to wtedy można byłoby powiedzieć, że „struktura strony z oceną jest kompatybilna wstecz". Wtedy Twój program nigdy nie wymagałby poprawek. Świetnie!

Niestety, ani projektanci USOSweba, ani żadnej innej strony internetowej nigdy się nie zgodzą na to, aby ich strona miała już „na zawsze" wyglądać tak samo. To dość zrozumiałe, w końcu na ich miejscu również chcielibyśmy, aby wygląd naszej strony podążał za trendami.

Jeśli więc strona WWW dostępna dla użytkowników nie może być kompatybilna wstecz, to może chociaż namówimy programistów USOS, aby stworzyli inną wersję tej samej strony, niekoniecznie ładną, ale kompatybilną wstecz, stworzoną nie z myślą o użytkownikach, lecz o programach takich jak nasz?

Takie strony istnieją i mają własną nazwę. Nazywamy je usługami internetowymi (ang. *web services*).



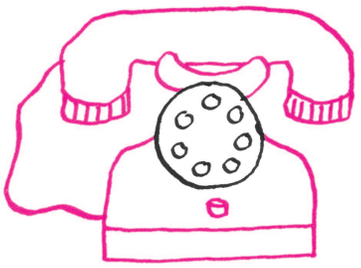
Jedna strona w dwóch wersjach. Po prawej wersja dla użytkownków (USOSweb), a po lewej wersja dla komputerów (USOS API). Obie strony przedstawiają plan zajęć przedmiotu w jednym tygodniu, a poziom szczegółowości wyświetlanych informacji można dowolnie konfigurować.

USOS API

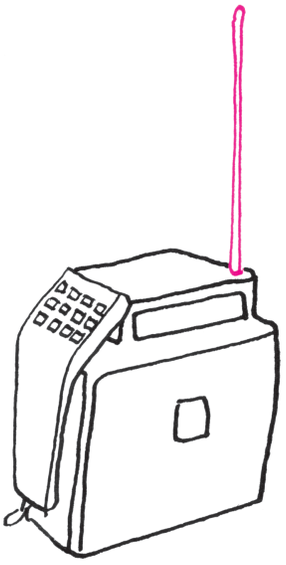
Aby zanadto nie komplikować, w tym artykule piszemy tylko o tym najprostszym sposobie komunikacji między programami, który faktycznie przypomina „zwykłe" surfowanie po Internecie (architektura klient-serwer). Nie jest to jednak jedyny sposób komunikacji. API może dokumentować wiele takich sposobów.

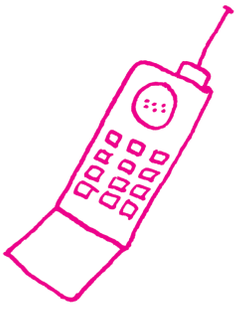
API, czyli interfejs programowania aplikacji (ang. *Application Programming Interface*), jest to ściśle określony zestaw reguł, zgodnie z którymi programy komunikują się ze sobą.

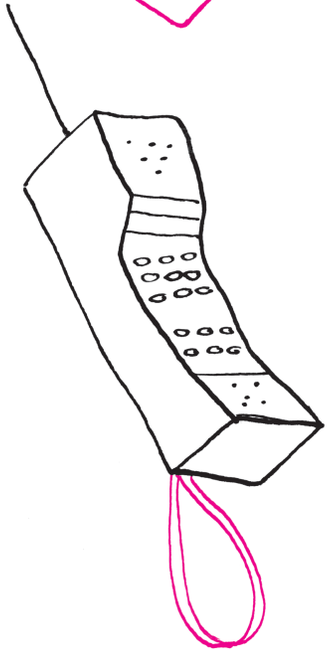
USOS udostępnia własne, publiczne API, którego główną składową są usługi internetowe, zwane również metodami. **Usługę internetową prezentującą listę ocen możesz wyświetlić w swojej przeglądarce, podobnie jak zwykłą stronę z ocenami w serwisie USOSweba**. Zwykle jednak nie będziesz tego robił, bo strona ta jest zaprojektowana w taki sposób, aby była czytelna dla programów komputerowych, a nie dla ludzi.



W przypadku stron internetowych priorytetem jest ich wygląd oraz prostota korzystania. W przypadku usług internetowych priorytetem jest ścisła dokumentacja oraz kompatybilność wstecz.







Większość z metod USOS API jest kompatybilna wstecz, a to oznacza, że obiecujemy, iż format przekazywanych przez nie danych nigdy się nie zmieni.

Inaczej mówiąc, dołożymy wszelkich starań, aby Twoja aplikacja powiadamiająca o nowych ocenach zawsze mogła na naszym API polegać, bez potrzeby żadnych zmian ani aktualizacji po Twojej stronie.

Refaktoryzacja w API

Refaktoryzację (ang. *refactoring*) możemy porównać do programistycznego odpowiednika „robienia porządków". Przykładem takich porządków jest zmiana nazwy metody lub jej parametrów tak, aby metoda była bardziej czytelna.

W każdym większym projekcie programistycznym co jakiś czas takie porządki się robi. **Jednak w przypadku API potrzeba utrzymania kompatybilności wstecz uniemożliwia programistom typową formę refaktoryzacji**. Jak więc możemy zachować porządek w dokumentacji USOS API, jeśli musimy coś zmienić, a jednocześnie musimy zachować kompatybilność wstecz?

Najczęściej tworzymy wtedy nową wersję metody, którą chcemy poprawić. Nie zastępujemy starej metody, lecz tworzymy drugą, działającą inaczej. Starą metodę oznaczamy zaś jako „przestarzałą" (ang. *deprecated*), lecz nadal w pełni funkcjonalną.

Inaczej mówiąc – nawet jeśli stwierdzimy w przyszłości, że moglibyśmy przekazywać te same dane „ładniej" bądź efektywniej, to i tak pozostawimy w naszym API poprzednią wersję metody, która nadal dostarczy dane „jak dotąd" – w sposób, być może, „brzydszy" i wolniejszy, lecz niezmiennie ten sam.

Z czasem może okazać się, że jedna metoda będzie mieć 3, 5, a nawet więcej różnych wersji, z czego tylko tę ostatnią zalecamy używać. Czasem tworzymy też nowe wersje całych modułów, a nie tylko pojedynczych metod. Utrzymanie tych wszystkich wersji nie zawsze jest łatwe i, być może, wydaje się mało porządne, ale kompatybilność wstecz jest warta wielu wyrzeczeń. My również z tego korzystamy.

Jak wykorzystujemy USOS API wewnątrz USOS

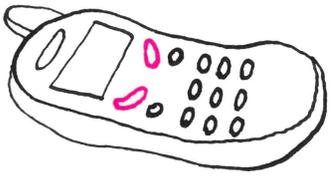
Wielu programistów uważa, że API są przeznaczone wyłącznie dla programistów „z zewnątrz". Na przykład dla studentów, którzy piszą aplikacje komórkowe dla własnej praktyki lub zabawy. Nic bardziej mylnego! **Programiści wszystkich aplikacji związanych z USOS również chętnie korzystają z USOS API**. Powodów jest parę, lecz główny z nich pewnie już znacie – my również widzimy zalety korzystania z rzeczy kompatybilnych wstecz. Jeśli raz napiszemy aplikację opartą na USOS API, to już nigdy nie będziemy musieli jej zmieniać.

USOS API również pomaga nam w rozwiązywaniu problemów związanych z synchronizacją danych między różnymi aplikacjami. Na przykład, zanim powstało USOS API, każda nasza aplikacja musiała utrzymywać duży zbiór zdjęć legitymacyjnych pracowników i studentów. Teraz większość aplikacji, gdy ma wyświetlić czyjeś zdjęcie, to „ściąga" je szybko z USOS API, w wersji odpowiednio wykadrowanej i pomniejszonej. Tym samym oszczędzamy nasze zasoby i unikamy powtarzania kodu. Podobnych przykładów jest wiele.

A czy mnie USOS API się do czegoś też przyda?

Jeśli planujesz studiować informatykę na uczelni korzystającej z systemu USOS, to nic nie stoi na przeszkodzie, żebyś Ty również skorzystał z USOS API. Dostęp do API jest otwarty, a wszystkie metody dobrze udokumentowane.

Głównym celem tego artykułu było jednak pokazanie Czytelnikom, że usługi internetowe są popularne i bardzo przydatne. Nie tylko w USOS, ale też w wielu innych systemach. Jeśli planujesz zostać programistą, to na pewno z wieloma takimi usługami się zetkniesz. I mamy nadzieję, że docenisz w nich to, za co tak je lubimy w USOS.



Oczywiście, czasem będziemy chcieli np. poprawić wygląd tej aplikacji itp. Niemniej jednak architektura oparta na kompatybilnym wstecz API pozwala takie zmiany robić wtedy, kiedy chcemy, tzn. **nigdy nie musimy jej robić**. Jest to szczególnie istotne w przypadku aplikacji komórkowych, których nie da się jednocześnie zaktualizować na urządzeniach wszystkich użytkowników.

Dokumentacja API (w języku angielskim) jest na `https://usosapps.uw.edu.pl/developers/api/`