Wprowadzenie do technologii Flex – wyszukiwarka serwisu twitter.com

Flex jest jedną z najbardziej zaawansowanych technologii do budowania aplikacji typu RIA w bezpośrednim tłumaczeniu bogatych aplikacji internetowych. Silnikiem wyświetlającym aplikacje Flex'owe jest technologia Adobe Flash, która pozwala na osiągnięcie jednolitego wyglądu uruchamianej aplikacji, niezależnie od wykorzystywanej przeglądarki czy systemu operacyjnego.

Kluczowymi cechami samego Flash'a jest wysoka dynamiczność i interaktywność, co przenosi się również bezpośrednio do Flex'a. Dodatkowo zestaw Flex SDK udostępniany jest na licencji *Open Source*.

Narzędzia dla programistów

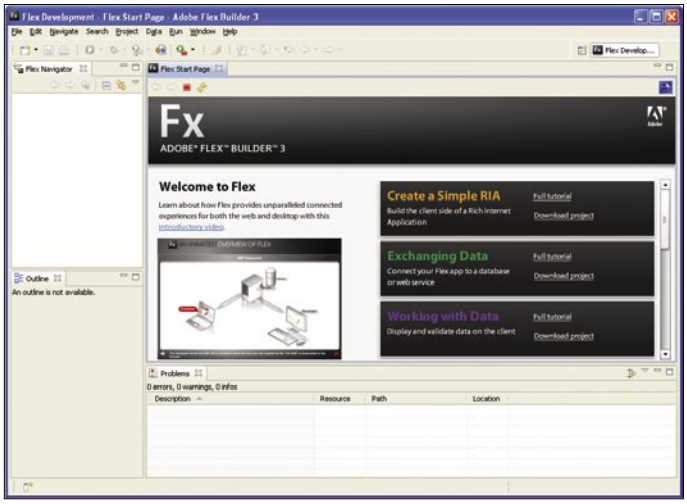
Aby rozpocząć tworzenie naszej pierwszej aplikacji typu RIA, można skorzystać z narzędzia Flex Builder. Jest to komercyjne narzędzie, ale na stronach firmy Adobe pod adresem *http://www.adobe.com/go/flex_trial* dostępna jest 60-dniowa wersja testowa. Dodatkowo studenci oraz osoby akademickie mogą się zarejestrować na stronie *https://freeriatools.adobe.com/* i otrzymać pełną wersję pakietu Flex Builder Professional zupełnie za darmo! Oczywiście na rynku jest szereg innych rozwiązań, takich jak: FlashDevelop (darmowy otwarty edytor), FDT (komercyjne narzędzie wspierające programistów języka ActionScript), Ensemble Tofino (darmowa wtyczka do środowiska Visual Studio), SapphireSteel Amethyst (również wtyczka do środowiska Visual Studio) oraz parę innych, dopiero od niedawna na rynku, wtyczek do środowisk NetBeans oraz IntelliJ IDEA.

Instalacja wymaganego oprogramowania

W naszym przypadku skorzystamy z testowej wersji oprogramowania Flex Builder. Po wcześniejszym ściągnięciu i zapisaniu na dysku komputera należy uruchomić instalator, który po wybraniu języka, akceptacji licencji oraz wyborze dodatkowych komponentów (dla potrzeb tego artykułu można pozostawić wartości domyślne) nastąpi faktyczny proces instalacji. W skład instalowanych pakietów wchodzi: środowisko Eclipse (na bazie, którego stworzone jest nasze narzędzie), oczywiście wtyczki samego Flex Builder'a dla Eclipse, SDK zawierające kompilator, zestaw bibliotek, dokumentację API oraz szereg przykładów stworzonych w technologii Flex. Dla niektórych programistów ważna może być również informacja, że środowisko Flex Builder można ściągnąć w postaci samych wtyczek do Eclipse'a. Taka konfiguracja może w dużym stopniu ułatwić przebieg pracy nad aplikacjami wykorzystującymi takie technologie, jak: Java lub PHP (wtyczki Eclipse/JDT, Eclipse/PDT oraz całe Zend Studio również bazują na platformie Eclipse) po stronie serwera i Flex'a jako warstwa widoku. Po uruchomieniu zainstalowanej aplikacji powinniśmy zobaczyć okno zgodne z Rysunkiem 1.

Pierwsza aplikacja typu RIA

Jako pierwsze ćwiczenie stworzymy aplikację pozwalającą na przeszukiwanie wpisów użytkowników portalu *twitter.com* (polskim odpowiednikiem tego portalu jest *blip.pl*). Pomysł na ten projekt zaczerpnąłem z video tutorialu mojego kolegi Lee Brimelow, również ewangelisty technicznego w firmie Adobe. Aby rozpocząć proces tworzenia nowego projektu z menu górnego wybieramy odpowiednią opcję *File>New>Flex Project*. Następnie otworzy się kreator ułatwiający tworzenie nowego projektu, tak jak to jest przedstawione na Rysunku 2. Wprowadzamy nazwę naszego projektu, np. *WyszukiwarkaTwittera* (inne opcje dostępne w kreatorze postaram się wyjaśnić w kolejnych artykułach w ramach Klubu Technicznego SDJ) i następnie możemy nacisnąć przycisk *Finish*. Po zakończeniu tego procesu dostaniemy nowo wygenerowany projekt, który automatycznie otworzy się w trybie edycji na pliku *WyszukiwarkaTwittera.mxml* (Rysunek 3). Plik ten jest domyślnie uruchamianym widokiem naszej aplikacji. Jest to plik w formacie XML, zgodny z dialektem MXML, który służy do opisu interfejsów użytkownika w technologii Flex. W tym miejscu dodam jeszcze, że językiem programowania służącym do implementacji logiki naszej aplikacji jest ActionScript 3. Jest to język obiektowy, bazujący na specyfikacji ECMA-262 i języku ECMAScript, na którym również bazuje JavaScript,



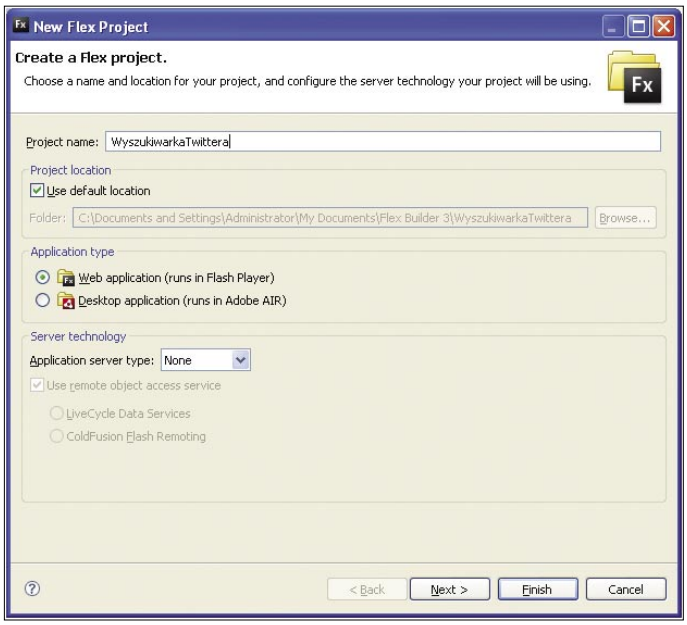
Rysunek 1. *Środowisko Flex Builder po pierwszym uruchomieniu*

co czyni go łatwym do zrozumienia dla programistów takich technologii, jak HTML/JavaScript czy AJAX.

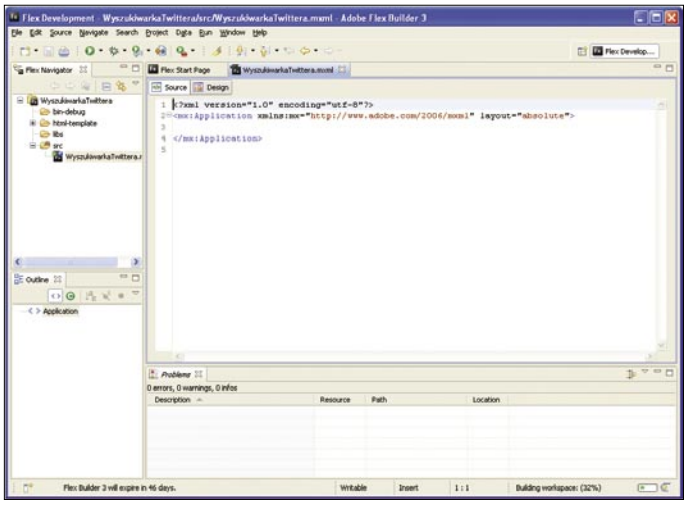
Teraz jesteśmy gotowi do przejścia w tryb wizualnej edycji naszej aplikacji, w tym celu naciskamy przycisk *Design* powyżej edytora kodu źródłowego. Po przejściu widzimy prostokątną powierzchnię roboczą (w odcieniu szaro-zielonym), na której możemy w łatwy sposób, za pomocą metody przeciągnij-upuść, układać komponenty aplikacji. Komponenty te dostępne są w lewym dolnym rogu Flex Builder'a, w okienku *Components* zawierającym listę, tak jak jest to przedstawione na Rysunku 4. Zacznijmy od przycisku, czyli elementu listy komponentów o nazwie *Button*. Przycisk poprzez przeciągnięcie możemy umieścić w prawym górnym rogu naszego obszaru roboczego. Zauważmy, że dojeżdżając we wskazane miejsce, edytor graficzny wyświetli jaskrawe linie sygnalizujące, że układany komponent znajduje się w odległości 10 pikseli od prawej i górnej części naszej aplikacji. Po upuszczeniu przycisk zostanie dodany, teraz możemy ustalić tzw. ograniczenia, które pozwolą zakotwiczyć komponent w wybranym miejscu. Jest to bardzo istotny krok szczególnie, gdy chcemy stworzyć aplikację, która automatycznie skaluje swoje elementy składowe w zależności od rozmiaru okna oraz rozdzielczości ekranu użytkownika. Do tego wykorzystamy edytor właściwości komponentów graficznych dostępny w prawym dolnym rogu Flex Builder'a. Edytor ten należy przewinąć paskiem przewijania na sam dół, by zobaczyć funkcje zgodne z tymi przedstawionymi na Rysunku 5. Zaznaczając odpowiednio dwa tzw. *checkbox'y* w prawym górnym rogu w pionie i poziomo, zakotwiczymy nasz przycisk tak, że niezależnie od zmian wielkości okna będzie on w odstępie 10 pikseli od obramowania naszej aplikacji. Jeżeli chcemy zmienić wyświetlany opis oraz identyfikator komponentu, należy przewinąć edytor właściwości graficznych do samej góry i wprowadzić odpowiednie wartości w polach *ID* oraz *Label*. W moim przypadku są to odpowiednio wartości *btnSearch* oraz *Szukaj*. Kolejne komponenty, jakie będziemy chcieli dodać, to pole tekstowe (komponent o nazwie *TextInput*), do którego użytkownik będzie mógł wpisać szukaną frazę oraz lista (komponent o nazwie *List*), w której zostaną wyświetlone wyniki wyszukiwania. W przypadku tych komponentów postępujemy analogicznie, jak w przypadku przycisku wyszukiwania, tak żeby uzyskać efekt zgodny z Rysunkiem 6. i Listingiem 1. Do edytora kodu źródłowego przechodzimy naciskając przycisk *Source* powyżej edytora wizualnego. Jeżeli chcemy sprawdzić, jak będzie wyglądał wynik końcowy, w oknie przeglądarki możemy uruchomić naszą aplikację, najłatwiej jest to zrobić za pomocą skrótowych przycisków w górnej belce menu. Mamy do dyspozycji odpowiednio dwa przyciski , pierwszy pozwalający uruchomić naszą aplikację w trybie normalnym, drugi w trybie debugowania. Po wybraniu jednej z tych opcji, aplikacja zostanie uruchomiona w domyślnej przeglądarce na naszym komputerze.

Implementacja logiki

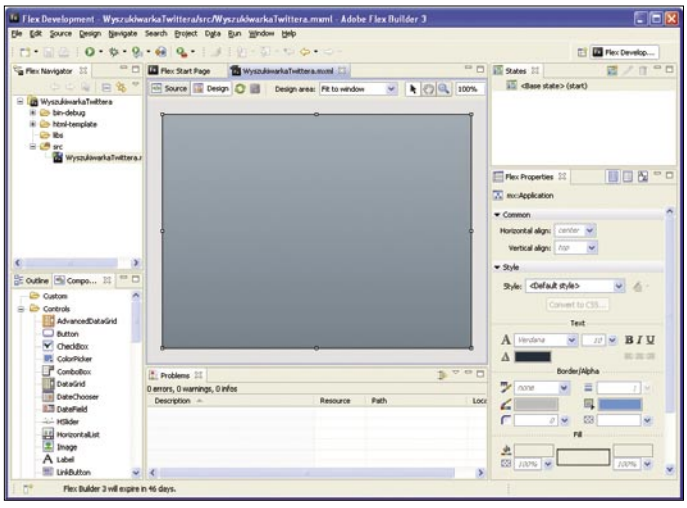
Serwis *twitter.com* w swoim API pozwala na wyszukiwanie wpisów użytkowników za pomocą prostych wywołań HTTP, które zwracają dane w formacie ATOM lub JSON. Więcej informacji na ten temat można przeczytać pod adresem *http://apiwiki.twitter.com/Search+API+Documentation*. W naszym przypadku skorzystamy z formatu ATOM oraz po stronie Flex'a z klasy HTTPService. Klasa ta pozwala na wykonanie zdalnych wywołań HTTP oraz przekazanie parametrów metodą GET, czyli parametrów zawartych w adresie URL. Klasę HTTPService możemy dodać jako komponent MXML, tutaj przychodzi nam z pomocą Flex Builder i jego mechanizm podpowiadania. Wystarczy będąc w edytorze kodu źródłowego wpisać część nazwy klasy, oczywiście wcześniej poprzedzając to nawiasem ostrokątnym otwierającym. Edytor wyświetli nam obok kursora listę dostępnych klas, z której możemy wybrać szukaną klasę HTTPService. Mechanizm ten (tzw. *Code Completion*) znany jest z praktycznie każdego narzędzia typu IDE dla współczesnych języków programowania. Po dodaniu klasy, możemy ustawić wartość atrybutu id np. na twitterService, pozwoli to odwoływać się do tego obiektu z innych elementów naszej aplikacji. Dodatkowo możemy ustawić wartość atrybutu url na *http://search.twitter.com/search.atom?q={txtSearch.text}*. Tutaj



Rysunek 2. *Kreator nowego projektu*

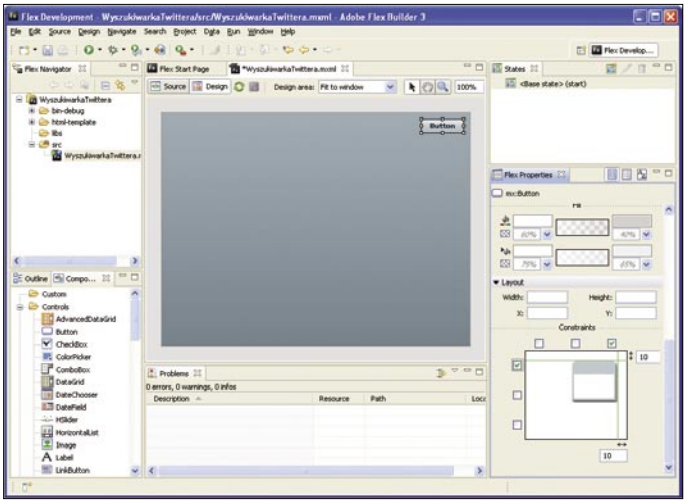


Rysunek 3. *Nowo utworzony projekt w trybie edycji pliku WyszukiwarkaTwitter a.mxml*

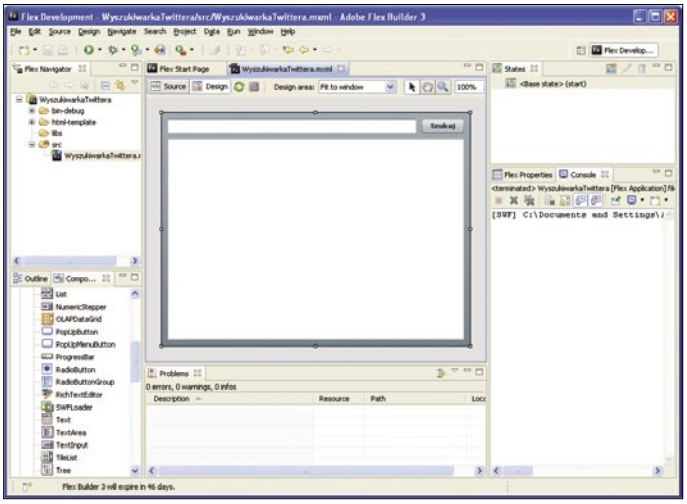


Rysunek 4. *Widok graficznej edycji aplikacji*

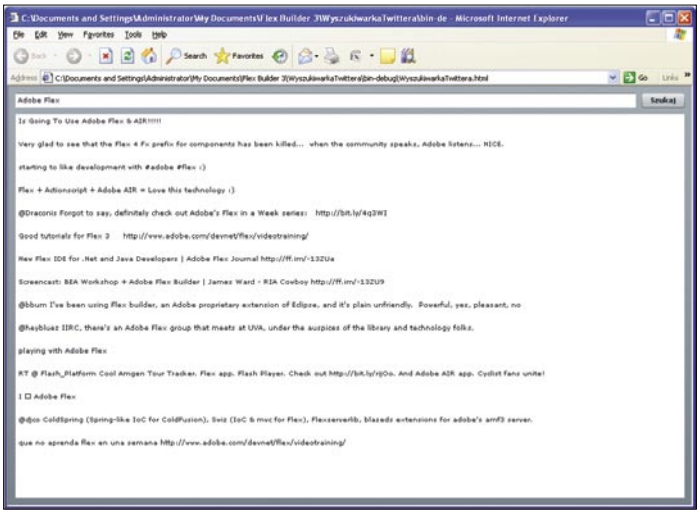
należy zwrócić uwagę na bardzo ważną rzecz, czyli sposób, w jaki przekazywana jest wartość z pola tekstowego *txtSearch*, które dodaliśmy w trakcie definiowania części wizualnej aplikacji. Referencja do tego pola i jego własność text jest umieszczona w nawiasach klamrowych, taka notacja wskazuje kompilatorowi wiązanie (z ang. tzw. *binding*), które będzie ewaluowane w trakcie działania aplikacji. Dalej pozostaje nam wywołanie metody `send()` na obiekcie klasy HTTPService. Możemy to zrobić również z poziomu komponentów wizualnych, obsługując zdarzenie click przycisku



Rysunek 5. *Widok właściwości komponentów graficznych aplikacji*



Rysunek 6. *Projekt wizualny aplikacji*



Rysunek 7. *Działająca aplikacja*

Szukaj. W tym celu wystarczy, że dodamy atrybut click i ustawimy mu wartość `twitterService.send()`.

Ostatnim krokiem, jaki pozostał, jest pobranie zwróconych wyników przez obiekt *twitterService* i umieszczenie ich w naszej liście *lstResults*. Można to zrobić na parę sposobów, również z poziomu komponentów wizualnych, jednak w tym przypadku pokażę, jak to zrobić za pomocą prostej logiki zaimplementowanej w języku ActionScript. Dobra praktyka wskazuje, aby część aplikacji implementowanej w ActionScript umieszczać powyżej definicji komponentów wizualnych. W tym momencie znowu pomocny będzie nam mechanizm podpowiadania, który po wybraniu komponentu *<mx:Script>* i zamknięciu nawiasu ostrokątnego generuje odpowiedni fragment CDATA, wewnątrz którego będziemy mogli umieszczać kod naszej aplikacji. Pierwszą rzeczą będzie zdefiniowanie zmiennej *resultsList* typu *ArrayCollection*. Tutaj warto również skorzystać z mechanizmu podpowiadania, gdyż po

Listing 1. *Kod źródłowy części wizualnej aplikacji*

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
                    layout="absolute">
   <mx:Button right="10" top="10" id="btnSearch"
                    label="Szukaj"/>
   <mx:TextInput left="10" top="10" right="82" id="txtSearch"/
                    >
   <mx:List id="lstResults" bottom="10" left="10" right="10"
                    top="40"></mx:List>
</mx:Application>
```

Listing 2. *Kod źródłowy całej aplikacji*

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
                    layout="absolute">

   <mx:Script>
      <![CDATA[
         import mx.collections.ArrayCollection;
         import mx.rpc.events.ResultEvent;

         [Bindable]
         private var resultsList:ArrayCollection;

         private function onResult(event:ResultEvent):void
         {
            resultsList = event.result.feed.entry as
                    ArrayCollection;
         }
      ]]>
   </mx:Script>

   <mx:HTTPService id="twitterService"
      url="http://search.twitter.com/search.atom?q={txtSearch
                    .text}"
      result="onResult(event)"/>

   <mx:Button click="twitterService.send()" right="10"
                    top="10" id="btnSearch" label="Szukaj"/>
   <mx:TextInput left="10" top="10" right="82" id="txtSearch"/
                    >
   <mx:List id="lstResults" dataProvider="{resultsList}"
                    labelField="title" bottom="10" left="10"
                    right="10" top="40"></mx:List>
</mx:Application>
```

wybraniu z listy klasy *ArrayCollection* automatycznie zostanie dodany tzw. import uwzględniający całą ścieżkę pakietu. Przykład przedstawiony jest na Listingu 2. Bardzo istotny w przypadku tej zmiennej jest atrybut *[Bindable]*, poprzedzający całą definicję. Atrybut ten pozwala na utworzenie wiązań, które automatycznie będą notyfikowały związane komponenty o zmianach w *resultsList*. W rzeczywistości wiązania te to odpowiedni kod, który jest automatycznie generowany w trakcie procesu kompilacji, takie rozwiązanie bardzo odciąża programistę i skraca kod całej aplikacji. Teraz możemy użyć naszej zmiennej jako wartość atrybutu *dataProvider* komponentu *lstResults*. Oczywiście nie zapominajmy o nawiasach klamrowych. W komponencie *lstResults* należy jeszcze wskazać nazwę własności elementów listy *resultsList*, zróbmy to za pomocą atrybutu *labelField* i wartości *title*. W tym przypadku elementy dokumentu XML w formacie ATOM o nazwie title będą wyświetlane jako treść każdej linii listy wyników. Ostatnią rzeczą, jaką zaimplementujemy w części Script, będzie metoda obsługująca zdarzenie result komponentu HTTPService. Zdarzenie to następuje asynchronicznie, zwracając rezultat wywołania zdalnego serwisu HTTP. Rezultat ten przekazywany jest w parametrze metody `onResult`. Parametr ten jest typu *ResultEvent* o nazwie event. Następnie, już w samej metodzie obsługującej zdarzenie, ustawiana jest wartość zmiennej resultsList. Tutaj ujawnia się bardzo ważna cecha języka ActionScript, czyli dynamiczność. Korzystając z mechanizmu podpowiadania można zauważyć, że typ zmiennej event posiada właściwość result, ale jej typ nie zawiera już właściwości feed, a jednak aplikacja się buduje i działa poprawnie. W tym przypadku feed jest to element główny dokumentu XML, zwracanego przez serwis *twitter.com*. Kolejne wywołanie właściwości entry zwraca już listę elementów, dlatego też rzutowanie na typ *ArrayCollection* nie powoduje żadnego wyjątku w trakcie działania aplikacji. Aby lepiej zrozumieć, jak wygląda struktura zwracanego dokumentu w formacie ATOM wystarczy, że otworzymy adres: *http://search.twitter.com/search.atom?q=Adobe+Flex* w oknie przeglądarki i podejrzymy źródło zwróconej treści. Po zaimplementowaniu metody onResult ,ustawiamy referencję do niej jako wartość atrybutu result komponentu HTTPService.

Teraz pozostało nam już tylko uruchomić aplikację, wpisać w pole tekstowe szukaną frazę, np. Adobe Flex i naciśnięcie przycisku *Szukaj*, a rezultaty powinny pojawić się po chwili w dolnej liście.

Podsumowanie

Jak widzimy, wykorzystując technologię Flex możemy tworzyć bardzo ciekawe rozwiązania za pomocą niewielkiej ilości kodu. Również bardzo istotne jest to, że domyślny wzorzec separujący warstwę widoku w postaci komponentów MXML od warstwy logiki aplikacji w języku ActionScript, opiera się na najlepszych praktykach programistycznych w tego typu rozwiązaniach. Dodatkowo pełna obiektowość, a zarazem możliwość definiowania dynamicznych typów, przy jednoczesnym wsparciu dla typowania statycznego, czynią tę technologię bardzo nowoczesną.

Mam nadzieję, że wprowadzenie to będzie dla Was inspiracją do stworzenia wielu nowoczesnym rozwiązań typu RIA.

Adobe Systems Incorporated

Adobe Systems Incorporated to jeden z największych i najbardziej wszechstronnych producentów oprogramowania na świecie. Znany zarówno z doskonałych programów graficznych oraz edytorskich, jak i zaawansowanych rozwiązań do bezpiecznego tworzenia, udostępniania i zarządzania dokumentami w biznesie oraz administracji publicznej.

PIOTR WALCZYSZYN
Adobe Systems Poland & Baltics
Platform Evangelist
Kontakt z autorem: piotr.walczyszyn@adobe.com