JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 102-107
Wysłane: 2017-11-17
Przyjęte: 2017-11-21

Tworzenie aplikacji internetowych w technologii ASP.NET MVC i JavaSerwer Faces

Mariia Radutina*, Beata Pańczyk

Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule przedstawiono wyniki analizy porównawczej dwóch konkurencyjnych technologii tworzenia aplikacji internetowych: ASP.NET MVC firmy Microsoft oraz JavaServer Faces (JSF) wspieranej przez Oracle. Badania zostały zrealizowane poprzez implementację dwóch aplikacji o takiej samej funkcjonalności, korzystającej z tej samej bazy danych MySQL. Do pracy z danymi wykorzystano najczęściej stosowane narzędzia typu ORM: Hibernate dla JSF i Entity Framework dla ASP.NET MVC. Przy porównaniu brano pod uwagę strukturę aplikacji, łatwość implementacji, wsparcie środowiska programistycznego, wsparcie społecznościowe, komponenty interfejsu graficznego oraz efektywność pracy z bazą danych.

Słowa kluczowe: ASP.NET MVC; JavaServer Faces; aplikacje internetowe

*Autor do korespondencji.

Adres e-mail: mariaradutina@gmail.com

Web application development using ASP.NET MVC and JavaServer Faces

Mariia Radutina*, Beata Pańczyk

Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The paper presents the results of comparative analysis of two competing web application technologies: ASP.NET MVC from Microsoft and JavaServer Faces (JSF) supported by Oracle. The research was done by implementing two applications with the same functionality using the same MySQL database. The most commonly used ORM tools are Hibernate for JSF and Entity Framework for ASP.NET MVC. The research was done by comparison the application structure, ease of implementation, support of the development environment, community support, graphical interface components, and database performance.

Keywords: ASP.NET MVC; JavaServer Faces; web application development

*Corresponding author.

E-mail address: mariaradutina@gmail.com

1. Wstęp

W warunkach szybkiego rozwoju technologii informacyjnych i szybkiej ewolucji systemów zarządzania informacją, najbardziej aktualne z zadań, które nieustannie występuję przed twórcami aplikacji, to wybór podejścia do tworzenia interfejsu. Podczas tworzenia aplikacji, deweloperzy często projektują interfejs, za pomocą specjalnych narzędzi do projektowania, przeciągając zestaw składników lub elementów sterowania na powierzchnię projektową [1].

Ze względu na olbrzymią konkurencję pomiędzy aplikacjami o podobnym zakresie funkcjonalności, użytkownik wybierze tę, która będzie bardziej intuicyjna i przyjazna a programista w prostszy sposób będzie w stanie zmienić interfejs oraz wspierać aplikację.

Do wyboru istniej dużo szkieletów programistycznych, bibliotek i narzędzi przeznaczonych dla różnych języków programowania. Ich nowe wersje pojawiają się na rynku nawet kilka razy w ciągu roku.

W niniejszym artykule do porównania wybrano dwie konkurencyjne technologie. JavaServer Faces (JSF) dla języka programowania Java oraz ASP.NET MVC dla języka programowania C#. Obie technologie mają na celu wsparcie procesu implementacji aplikacji webowych.

ASP.NET MVC, firmy Microsoft, łączy w sobie skuteczność i dokładność architektury model-widok-kontroler (MVC), najnowsze pomysły i wszystko, co najlepsze z istniejącej platformy ASP.NET [2].

JavaServer Faces (JSF) zmienił sposób pisania aplikacji internetowych w Javie. JSF proponuje eleganckie rozwiązania kluczowych problemów, związanych z opracowaniem komercyjnych zastosowań webowych [3]. JavaServer Faces opiera się również na architekturze MVC i jest standardem dla aplikacji Java EE. Technologię aktywnie rozwijają Oracle i IBM, a poza tym utworzono wiele bibliotek, które z nim współpracują. pozwalając wykorzystać niestandardowe komponenty UI, oparte na przykład na jQuery [4].

Obie technologie umożliwiają pracę z bazami danych za pomocą dedykowanych narzędzi ORM (ang. Object Relational Mapping) [5]. Dwa najbardziej popularne rozwiązania typu ORM to Hibernate dla języka Java i Entity Framework dla języka C#.

W artykule przedstawiono wyniki analizy porównawczej opartej na dwóch, identycznych co do funkcjonalności aplikacjach testowych współpracujących z tą samą bazą danych MySQL.

Przyjęto założenie, że przystępując do tworzenia aplikacji webowej, programista zna już podstawy programowania w języku C# i Java.

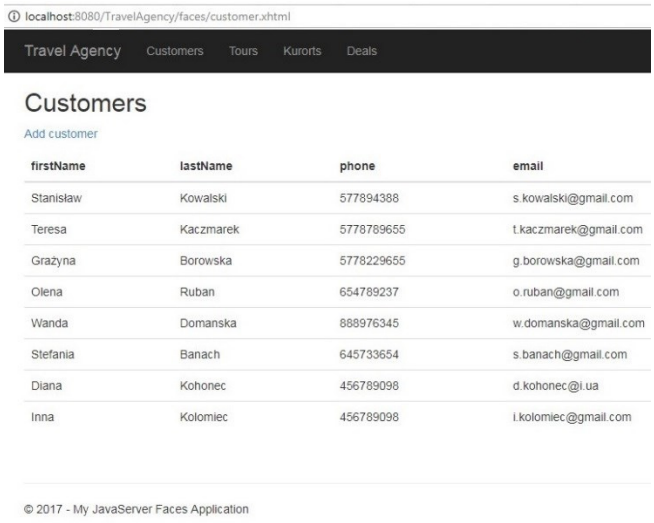
Postawiono następujące tezy:

T1: Obie technologie umożliwiają stworzenie aplikacji współpracującej z bazą danych MySQL o tej samej funkcjonalności.

T2: Technologia ASP.NET MVC jest bardziej przyjazna dla początkującego programisty aplikacji webowych w stosunku do technologii JSF.

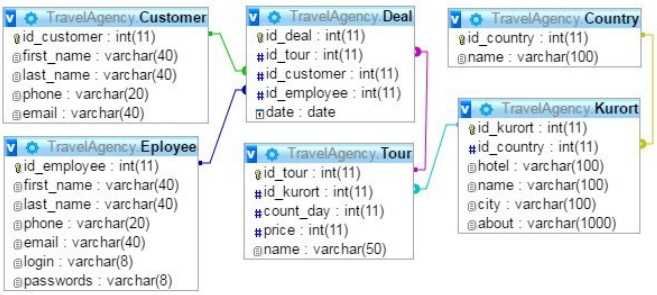
2. Aplikacje testowe

Prosta aplikacja testowa do obsługi biura podróży pozwala na realizację podstawowych funkcjonalności typu CRUD (ang. Create, Read, Update, Delete). Obie aplikacje posiadają jednakowy, prosty interfejs przedstawiony na rysunku 1. Schemat bazy danych przedstawia rysunek 2.



Rys. 1. Strona aplikacji z widokiem klientów biura podróży

Schemat bazy danych przedstawiono na Rys. 2.



Rys. 2. Schemat bazy danych dla aplikacji testowych

3. Ogólne cechy obu platform

Tabela 1 przedstawia podstawowe właściwości analizowanych technologii.

Z danych w tabeli 1 widać przewagę technologii ASP.NET MVC nad technologią JavaServer Faces, w kwestii związanej z rozwojem i ciągłym unowocześnianiem frameworka. Technologia ASP.NET jest na bieżąco aktualizowana. Niemalże każdego roku wychodzą jej nowe wersje, które mają nowe właściwości oraz możliwości. Najpopularniejsza wersja technologii JSF 2.2, z której korzysta największa liczba osób, została wydana w 2013 roku, a ostatnia wersja JSF 2.3 pojawiła się dopiero w roku 2017, po 4 letniej przerwie. Developerzy frameworka ASP.NET MVC dodają nowe funkcje i możliwości, zwiększając tym samym jego konkurencyjność.[6].

Tabela 1. Właściwości technologii ASP.NET MVC i JavaServer Faces

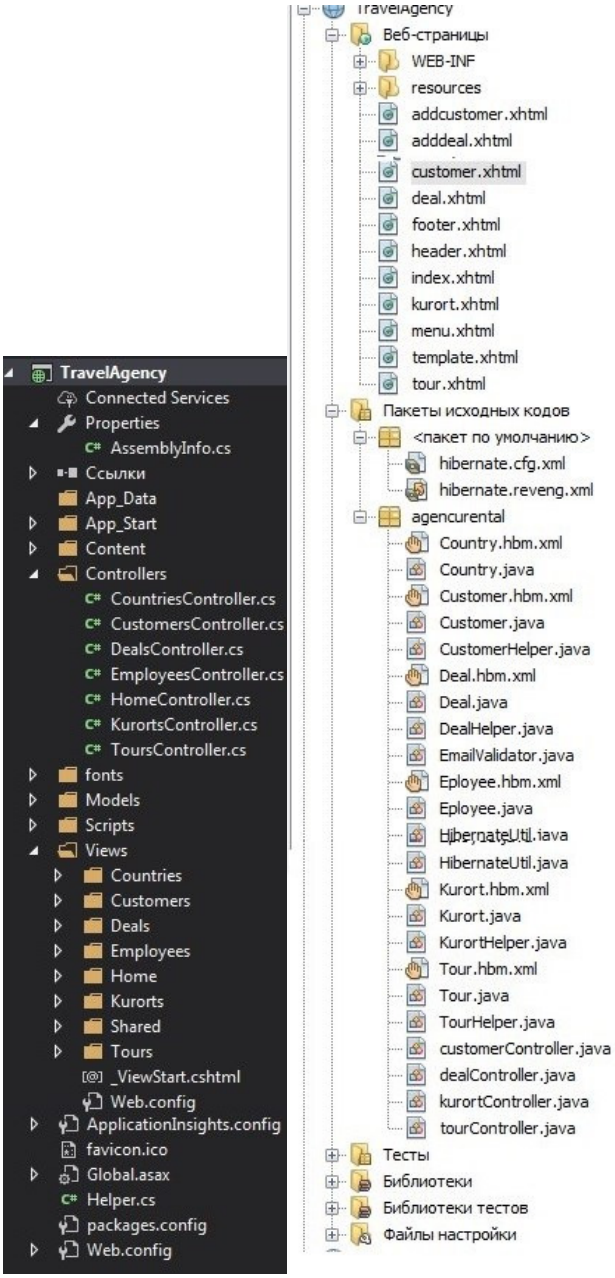
Właściwości	ASP.NET MVC 5	JavaServer Faces 2.2
Data przedstawienia	10 grudnia 2007	1 lipca 2009
Data ostatniej wersji	ASP.NET MVC 6 — 1.0.0 (2016)	JSF 2.3 (2017)
Twórca	Microsoft	Oracle Mojarra, Apache MyFaces
Język programowania	C#	Java
Open Source	Komercyjna	Tak
Środowisko programistyczne	Visual Studio	NetBeans, Eclipse, IntelliJ IDEA
Hosting	IIS	Nie posiada własnego hostingu
Technologia widoku	Razor	Facelet lub JSP
Stylizacja widoku	Wbudowany Bootstrap	Nie posiada wbudowanych narzędzi
Standard widoku	HTML5	XHTML 1.1
Generacja widoku wspomagana przez IDE	Jest możliwość generacji widoku z kontrolera dla jego wybranej metody	Nie posiada takiej możliwości
Tutoriale	Wiele oficjalnych informacji, wiele blogów, przykładów	Dokumentacja na stronie Oracle, mało informacji
Dedykowany serwer	IIS	Tomcat, GlassFish
Współpraca z MySQL i inną bazą danych	Jest możliwość współpracy	Jest możliwość współpracy
ORM wspierające pracę z bazą danych	Entity Framework, NHibernate, DataObjects.NET, ADO.NET Entity Framework	Hibernate, SpringORM
Instalacja bibliotek	Przez silnik NuGet	Dodawanie ręcznie pobranych bibliotek z oficjalnej strony
System operacyjny	Windows	Windows, Linux itp.

4. Struktura aplikacji

Aplikacje wytworzone w obu technologiach posiadają podobne struktury (Rys. 3), podzielone na trzy warstwy: model, widok, kontroler oraz zasoby uzupełniające w postaci plików CSS, skrypty JS.

W ASP.NET MVC w środowisku programistycznym Visual Studio automatycznie są tworzone foldery dla modeli, kontrolerów, widoków, plików stylu itp.

W JSF występuje obowiązek samodzielnego tworzenia odpowiedniej struktury katalogowej. Standardem jest podział na pliki Java oraz pliki warstwy web. Pliki w folderze Java zawierają klasy modeli i kontrolerów, a pliki w folderze web - widoki, pliki stylów CSS oraz skrypty JS.



Rys. 3. Struktura testowej aplikacji ASP.NET MVC i JSF

Obie technologie zawierają również pliki konfiguracyjne, ASP.NET MVC w folderze głównym a JavaServer Faces w folderze *web* oraz pliki konfiguracji Hibernate w pakietach Java i plik *Manifest* w folderze konfiguracji.

Przykład 1 przedstawia fragment kodu klasy modelu *Customer* reprezentującej klienta biura podróży dla JSF. Przykład 2 prezentuje analogiczny kod dla ASP.NET MVC.

Z przykładów 1-2 widać, że sposób definiowania podstawowych metod typu get/set (wykorzystywanych w klasach modeli) w C# sprowadzony jest do jednego wiersza, co generalnie wpływa na zmniejszenie liczby wierszy w klasach modeli.

Przykład 1. Fragment modelu Customer dla JSF

```
public String getPhone()
{
    return this.phone;
}
public void setPhone(String phone)
{
    this.phone = phone;
}
```

Przykład 2. Fragment modeluCustomer dla ASP.NET MVC

```
[Required]
[StringLength(20)]
[Display(Name = "Phone")]
public string phone { get; set; }
```

5. Interfejs graficzny

W aplikacji JSF intefejs graficzny został zbudowany za pomocą technologii facelet, natomiast ASP.NET MVC korzysta z silnika widoków Razor. Interfejs obu programów jest jednakowy (Rys. 1), drobna różnica występuje w nagłówku.

Przykład 3 przedstawia kod dodania bibliotek specjalnych, stylów i skryptów JavaScript dla JSF (13 linijek kodu). W celu podłączenia tych plików wykorzystywany jest specjalny znacznik biblioteki JSF, generujący nagłówek dokumentu HTML *<h:head>* oraz znacznik wskazujący źródło dołączanych zasobów *<h:outputStylesheet>*. W JSF występuje konieczność dodatkowego podłączenia bibliotek dla znaczników JSF a każdy plik stylów oraz skryptów dołączany jest oddzielnie.

Przykład 3. Nagłówek pliku widoku dla JSF

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <meta name="viewport" content="width=device-width,
              initial-scale=1"></meta>
        <h:outputStylesheet library="css" name="Site.css"/>
        <h:outputStylesheet library="css" name="bootstrap.css"/>
        <h:outputStylesheet library="js" name="modernizr-2.6.2.js"/>
        <title>Tour agency</title>
    </h:head>
```

Przykład 4 przedstawia sposób dołączenia stylów dla ASP.NET MVC za pomocą funkcji silnika Razor. W tym wypadku nagłówek zajmuje 9 linii kodu.

Przykład 4. Nagłówek pliku widoku dla ASP.NET MVC

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
          initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
```

W ASP.NET wszystkie style podłączane są za pomocą jednej linii kodu. Ten framework bazuje na silniku Razor i czystym kodzie HTML, nie trzeba korzystać z dodatkowych bibliotek znaczników. Razor oferuje natomiast tzw.

'pomocników' (ang. helper) do generowania kodu HTML z poziomu pliku widoku.

Przykład 5 i 6 przedstawiają odpowiednio dla JSF i ASP.NET MVC, fragmenty plików widoku kolumny tabeli z łączem do strony usunięcia elementu *Customer*.

Przykład 5. Fragment widoku pojedynczej kolumny z hiperłączem dla JSF

```
<h:column>
  <h:commandLink
      action="#{customerController.deleteCustomer
               (item.idCustomer)}"
               value="Delete"/>
</h:column>
```

Przykład 6. Fragment widokupojedynczej kolumny z hiperłączem dla ASP.NET MVC

```
<td>
    @Html.ActionLink("Delete",
  "Delete", new { id=item.id_customer })
</td>
```

JSF do generowania widoków korzysta z technologii *facelet* i wykorzystuje specjalne znaczniki (np. *<h:column>* - komórka tabeli z zawartością, *<h:commandLink>* - hiperłącze z możliwością wywołania akcji *Delete*). ASP.NET MVC bazuje na technologii widoków Razor i korzysta z prostych znaczników tabeli HTML (np. *<td>*) a hiperłącza umożliwiające wywołanie akcji *Delete,* wstawiane jest również za pomocą specjalnych „helperów" Razora (np. *@Html.ActionLink*).

Porównując oba pliki widoków - ponownie widać, że zastosowanie silnika Razor skraca i upraszcza kod.

W opinii autorki, napisanie interfejsu w technologii ASP.NET MVC jest łatwiejsze. Środowisko programistyczne Visual Studio pozwala wygenerować widok do wybranej funkcji kontrolera, ale można też dowolnie zmienić stylizację lub wszystkie pliki tworzyć ręcznie. Udostepnienie gotowego szablon (opartego o Bootstrap), zdecydowanie ułatwia i przede wszystkim przyspiesza proces projektowania layoutu strony W przypadku JSF udostępniane są bardzo podstawowe szablony i cały layout strony należy projektować manualnie.

Jeśli chodzi o komponenty GUI to również ASP,NET MVC oferuje więcej kontrolek. Natomiast JSF wymaga tu wsparcia dodatkowych bibliotek np. Prime Faces [7], RichFaces [8].

6. Współpraca z bazą danych

Dla aplikacji JSF wymagane jest pobranie jednej biblioteki MySQL Connector Java z oficjalnej strony Oracla oraz dodanie jej do projektu.

Dla aplikacji ASP.NET MVC dodawanie bibliotek MySQL jest łatwiejsze, ale potrzebne są dwie biblioteki: *MySql.Data*, *MySql.Data.Entity* Wystarczy w środowisku programistycznym otworzyć pakiet NuGet oraz wybrać do zainstalowania te biblioteki.

Przykład 3 przedstawia konfigurację połączenia z baza danych przy pomocy Entity Framework dla ASP.NET MVC. W pliku *Web.config* w znaczniku *connectionString* umieszczono parametry połączenia z serwerem bazy danych: *nazwa użytkownika, nazwa serwera, parametry bazy danych.*

Przykład 3. Konfiguracja połączenia z bazą danych w Entity Framework

```
<connectionStrings>
  <add name="Master"
       connectionString="user id=root;
       password=;
       server=localhost;
       database=TravelAgency;
       Convert Zero
       Datetime=True;
       persistsecurityinfo=True"
providerName="MySql.Data.MySqlClient" />
</connectionStrings>
```

Przykład 4 przedstawia konfigurację połączenia z baza danych przy pomocy Hibernate. Plik *hibernate.cfg* jest tworzony automatycznie przy tworzeniu pustej aplikacji. W tagach *property* jest wskazana konfiguracja połączenia do bazy danych: typ dialektu, link do serwera, użytkownik oraz pliki mapowanych tabel z bazy danych.

Przykład 4. Konfiguracja połączenia z bazą danych przy pomocy Hibernate

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">
        org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">
        com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">
        jdbc:mysql://localhost:3306/TravelAgency?
        zeroDateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.current_session_context_class">
        thread</property>
    <mapping resource="agencurental/Eployee.hbm.xml"/>
    <mapping resource="agencurental/Country.hbm.xml"/>
    <mapping resource="agencurental/Deal.hbm.xml"/>
    <mapping resource="agencurental/Kurort.hbm.xml"/>
    <mapping resource="agencurental/Customer.hbm.xml"/>
    <mapping resource="agencurental/Tour.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

W kwestii podłączenia MySQL, można stwierdzić, że z jednej strony lepsza jest technologia JSF (wymagana jest dodatkowo jedna biblioteka, która zajmuje 2 razy mniej pamięci). A z drugiej strony w technologii ASP.NET MVC łatwiejszy jest proces instalacji bibliotek, które jednak zajmują więcej pamięci.

W obu aplikacjach przez narzędzia ORM (Hibernate dla JSF, Entity Framework dla ASP.NET MVC), automatycznie mapowane są tabele bazy danych na odpowiednie klasy modeli.

W obu technologiach mapowanie jest niemal jednakowe. Hibernate tworzy dodatkowe dwa pliki konfiguracyjne. Po mapowaniu tworzą się klasy z polami tabel bazy danych, z funkcjami dostępowymi get/set. Oba frameworki automatycznie generują odpowiednie kody w plikach konfiguracyjnych, ustawiając parametry połączenia do bazy danych.

Przykład 5 przedstawia metody tworzenia listy danych z tabeli *Deals* (korzystając z modelu Deals i języka zapytań LINQ) przy pomocy Entity Framework.

Przykład 5. Formowanie listy Deals przy pomocy Entity Framework

```
var deals = db.Deals.Include(d => d.customer)
    .Include(d => d.eployee).Include(d => d.tour);
return View(deals.ToList());
```

Przykład 6 przedstawia metody tworzenia listy danych z tabeli *Deals* przy pomocy Hibernate.

Przykład 6. Tworzenie listy Deals przy pomocy Hibernate

```
List<Deal> dealList = null;
org.hibernate.Transaction tx = session.beginTransaction();
Query q = session.createQuery ("from Deal  ");
dealList = (List<Deal>) q.list();
```

Tworzenie listy umów z klientami w ASP.NET MVC bazuje na odpowiednim zapytaniu LINQ wykonanym za pomocą jednej instrukcji. W JSF najpierw należy zadeklarować listę umów, następnie, uruchomić transakcję Hibernate i dopiero zapytanie do baz danych

Przykład 7 przedstawia kod dodawania nowego klienta do tabeli *Customer* w ASP.NET MVC za pomocą Entity Framework. Przykład 8 prezentuje analogiczny kod dla Hibernate i JSF.

Przykład 7. Dodawanie nowego klienta w ASP.NET MVC

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include =
    "id_customer,first_name," +
    "last_name,phone,email")] Customer customer)
{
    if (ModelState.IsValid)
    {
        db.Customers.Add(customer);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(customer);
}
```

Przykład 8. Dodawanie nowego klienta w technologii JSF

```
public void addCustomer() throws IOException
{
      Customer customer = new Customer();
       customer.setFirstName(first_name);
       customer.setLastName(last_name);
       customer.setEmail(email);
       customer.setPhone(phone);
       CustomerHelper customersClass = new CustomerHelper();
       customersClass.insertCustomer(customer);
       HttpServletResponse response =
       (HttpServletResponse) FacesContext.getCurrentInstance().
             getExternalContext().getResponse();
       response.sendRedirect("/TravelAgency/faces/customer.xhtml");
}
```

Dodawanie nowego obiektu w ASP.NET MVC zajmuje mniej linii kodu. W JSF należy uzupełnić pola obiektu, korzystając z funkcji *set*. W ASP.NET MVC jest to realizowane automatycznie za pomocą mechanizmu Data Model Binding i Entity Framework - dlatego wystarczy jedna instrukcja do pobrania danych klienta z formularza, utworzenie nowego obiektu *Customer* i dodania go do bazy danych.

Przykład 9 przedstawia metody usuwanie klienta z tabeli *Customer* w ASP.NET MVC przy pomocy Entity Framework.

Przykład 9. Usuwanie klienta w ASP.NET MVC za pomocą Entity Framework

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Customer customer = db.Customers.Find(id);
    db.Customers.Remove(customer);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Przykład 10 przedstawia metody usuwanie klienta z tabeli *Customer* w JSF przy pomocy frameworka Hibernate.

Przykład 10. Usuwanie klienta w JSF za pomocą Hibernate

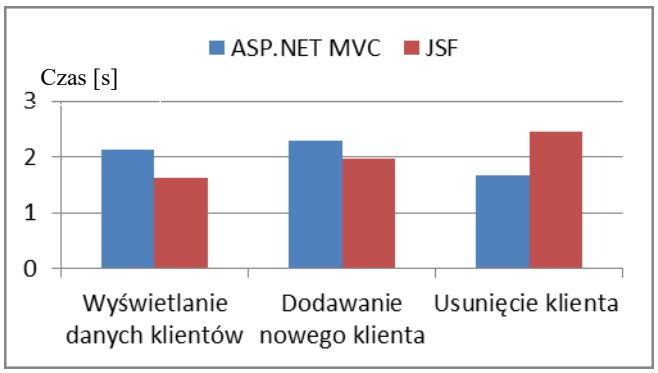
```
public void deleteCustomer(int id)
{
    SessionFactory sessionFactory;
    sessionFactory =
            new AnnotationConfiguration().
                    configure().buildSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();
    Query q = session.createQuery("from Customer where "
            + "id_customer = :id_customer ");
    q.setParameter("id_customer", id);
    Customer stock = (Customer)q.list().get(0);
    session.delete(stock);
    session.getTransaction().commit();
    session.close();
}
```

Usuwanie obiektów w obu technologiach jest proste i w zasadzie podobnie kodowane.

Ważnym aspektem pracy z bazą danych jest czas wykonywania operacji bazodanowych. Tabela 2 i rysunek 4 przedstawiają czas wykonywania podstawowych operacji na tabeli *Customer* w obu technologiach.

Tabela 2. Czas wykonywania operacji na bazie danych

	ASP.NET MVC	JSF
Pierwsze uruchamianie (sekundy)	34,15	74,27
Wyświetlanie danych klienta (sekundy)	2,13	1,63
Dodawanie nowegoklienta (sekundy)	2,3	1,98
Usuwanie klienta (sekundy)	1,67	2,46



Rys. 4. Wydajność operacji bazodanowych [s]

7. Metryki kodu aplikacji

Ostatnim analizowanym kryterium porównania obu aplikacji były podstawowe metryki kodu, które zostały zestawione w tabeli 3.

Aplikacja w technologii ASP.NET MVC zajmuje do 5 razy więcej pamięci, i wykorzystuje więcej bibliotek.

Porównując fizyczne linie kodu, można stwierdzić, że aplikacja JavaServer Faces zajmuje więcej linii kodu. Tylko dla widoku w obu technologiach liczba linii kodu jest jednakowa.

Dla pliku modelu JavaServer Faces zajmuje półtora razy więcej linii kodu. Wynika to z faktu, że w ASP.NET MVC metody get oraz set są pisane zwykle w jednej linii.

Tabela 3. Wybrane metryki kodu obu aplikacji

	JavaServer Faces	**ASP.NET MVC**
Liczba plików aplikacji	164	384
Pliki konfiguracyjne	5	5
Liczba bibliotek	18	26
Waga bibliotek(MB)	11.5	68.7
Kontroler - liczba linii kodu	180	115
Widok - liczba linii kodu	50	50
Model - liczba linii kodu	69	43
Waga projektu (MB)	22.3	103

8. Wnioski

W artykule przedstawiono najważniejsze aspekty analizy porównawczej technologii ASP.NET MVC oraz JavaSerwer Faces. Stworzone aplikacje testowe potwierdziły tezę 1, że obie technologie umożliwiają stworzenie aplikacji współpracującej z bazą danych MySQL o tej samej funkcjonalności.

Na podstawie doświadczenia nabytego w trakcie tworzenia aplikacji w obu środowiskach i przedstawionych wyników, można wnioskować, że to ASP.NET MVC jest dla początkującego programisty łatwiejszy i bardziej przyjazny. Oferuje możliwość automatycznego generowania layoutu, tworzenia widoków do wybranego modelu danych, czy też do wskazanych akcji kontrolerów. Dodatkowo istnieje wiele materiałów pomocniczych udostępnianych na stronach Microsoft, blogach i forach dyskusyjnych. Nie wymaga też instalowania dodatkowych bibliotek z komponentami GUI.

Technologia JavaServer Faces jest silnym konkurentem dla ASP.NET MVC, jednak oferuje zdecydowanie mniejsze wsparcie, jeśli chodzi o generowanie kodu dla aplikacji webowej. Jej niezaprzeczalną zaletą jest dużo mniejsza waga gotowej aplikacji.

Literatura

[1] Chris Sutton Programming ASP.NET MVC; Москва, 2008.

[2] http://smarly.net/pro-asp-net-mvc-4/introducing-asp-net-mvc-4/what-is-the-big-idea/key-benefits-of-asp-net-mvc [03.11.2017]

[3] https://professorweb.ru [12.08.2017]

[4] Kito D. Mann: JavaServer Faces in Action, Manning Publications Company, 2005.

[5] Дэвид Гири, Кей С. Хорстманн. Core JavaServer Faces. Вильямс, 2011.

[6] David Geary, Cay Horstmann: Core JavaServer Faces, Second Edition, Prentice-Hall, 2007.

[7] http://showcase.richfaces.org/, [18.11.2017]

[8] https://www.primefaces.org/showcase/, [18.11.2017]