Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/07BE37Baštovanovic

GENERATOR STATIČKIH I DINAMIČKIH WEB APLIKACIJA

STATIC AND DINAMIC WEB APPLICATION GENERATOR

Nikola Baštovanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljen je jezik i generator programskog koda za statičke i dinamičke web aplikacije. Generator kao ulaz koristi model pisan na internom jeziku specifičnom za domen (DSL-u). DSL je implementiran u okviru XML-a. Generisanje koda se vrši uz pomoć T4 tekstualnog obrađivača šablona, kao i korišćenjem parsera jezika koji se koristi na ciljanoj platformi.*
Ključne reči: *Generator koda, C#, .Net, ASP.Net, Sql Server, T4 obrađivač šablona*
Abstract – *This paper presents the language and static and dinamic web application code generator. Code generator uses model written in internal Domain Specific Language (DSL) as an input. Code generation is performed using T4 template engine and also by using parsers for target platform languages.*
Keywords: *Code generator, C#, .Net, ASP.Net, Sql Server, T4 template engine*

1. UVOD

Prilikom razvoja web aplikacija, oslanjamo se na slične ili čak identične koncepte. Kreiranje html stranica koje se jedna od druge razlikuju u fontu, formatiranju, boji teksta i slično, a sve specificirano u css klasama. Imajući sve ovo u vidu, zašto bi iznova i iznova pisali kod koji je već u velikoj meri napisan? Odgovor na ovo pitanje je tema ovog rada. Razvoj generatora koji će pojednostaviti razvoj ovakvih aplikacija.

Osnovni zadatak generatora programskog koda jeste da generiše kod koji je ispravan, dakle kod koji može da se pokrene bez prikazivanja grešaka. Ono što je takođe veoma bitno kod generatora programskog koda je da kod koji je generisan bude što je moguće efikasniji.

U okviru ovog rada biće prikazan generator programskog koda koji kao ulaz koristi model pisan na internom DSL-u (Domain-Specific Language). DSL je implementiran u okviru XML-a (Extensible Markup Language).

Na osnovu ulaznog modela generator će kreirati: skript za kreiranje šeme baze podataka, osnovne operacije nad bazom podataka, kao i predefinisani izgled samih stranica koje korisnik može naknadno menjati u skladu sa svojim potrebama, željama i zahtevima.
S obzirom da generator, pored statičkih, generiše i dinamičke ASP.Net aplikacije, generator će na osnovu ulaznog modela generisati i ASP.Net klase koje omogućavaju pravilan rad generisane aplikacije.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

2 PROJEKTOVANJE I IMPLEMENTACIJA TEHNIČKOG REŠENJA

U okviru ovog poglavlja je opisan jezik specifičan za domen, neke od njegovih prednosti i mana, implementacija internog jezika specifičnog za domen u XML formatu. Takođe, dat je prikaz dijagrama aktivnosti korišćenja sa detaljnim objašnjenjem, Kratak opis i primeri upotrebe T4 tekstualnog obrađivača šablona, osnove Entity Framework-a i Owin-a.

2.1 Jezik specifičan za domen

Jezik specifičan za domen (Domain-specific language - DSL) je programski jezik koji je specifičan za određeni, usko definisan domen primene. Za razliku od jezika opšte namene (General Purpose Language - GPL) koji se široko primenjuju za različite namene i nedostaju mu specijalizovane funkcije za određenu oblast, nude povećanje ekspresivnosti koje se postiže kroz upotrebu koncepata i notacija prilagođenih domenu problema i domenskih eksperata [2].
Jezici specifični za domen se mogu podeliti prema vrsti jezika [1]:

1. - Domenski specifični jezici za obeležavanje;
2. - Domenski specifični jezici za modelovanje i
3. - Domenski specifični programski jezici.

Kada je reč o upotrebi jezika specifičnog za domen, onda treba napomenuti da kreiranje ovakvog jezika ima smisla samo onda kada takav, domenski specifičan jezik, omogućava da se određena vrsta problema ili rešenja izrazi jasnije nego što bi to dozvoljavao neki od postojećih jezika, ali i da se tip problema koji se tim načinom rešava pojavljuje dovoljno često kako bi ovakav metod bio opravdan i isplativ.

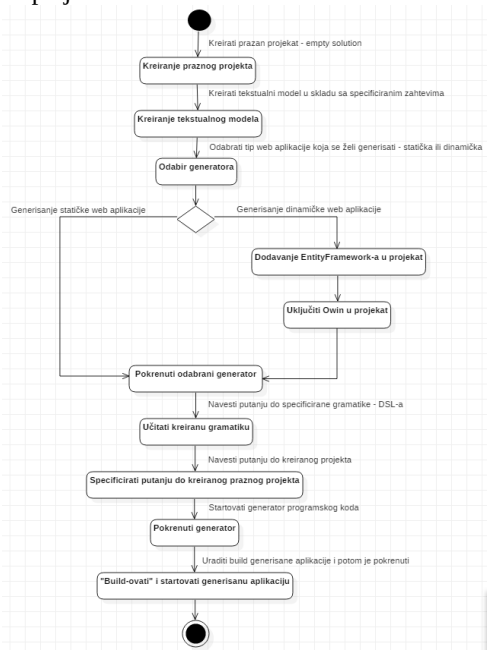
2.2 Dijagram aktivnosti

Na slici 1 je dat dijagram aktivnosti korišćenja sistema koji se razvija u ovom radu.

2.3 T4 Tempalte

U ovom projektu korišćen je C# T4 obrađivač šablona. Šabloni pisani na jeziku datog obrađivača predstavljaju kombinaciju tekstualnih blokova i kontrolne logike pri kreiranju tekstualnih fajlova koji mogu biti različitih ekstenzija. Neki od tipova ekstenzija koji su korišćeni u projektu su: .sql koji predstavlja šablon za generisanje sql skripta koji se pušta nad bazom podataka, .cs što predstavlja šablon za generisanje klasa u C# programskom kodu, .aspx za generisanje ekstenzija

ASP.Net klase, i druge. Blokovi koji sadrže logiku sadrže C# programski kod sa svim svojim pogodnostima koje C# programski jezik pruža [5].
Kada je reč o T4 templejtu, razlikuju se dva osnovna tipa ovog templejta:



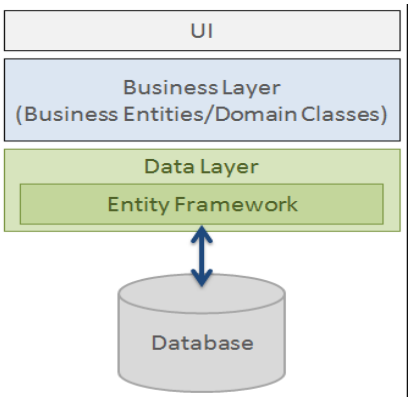
Slika 1: *Dijagram aktivnosti generatora programskog koda*

- RunTime - koji se izvršavaju, kako im i naziv kaže, u vreme izvršavanja aplikacije i generišu kod koji se direktno koristi u vreme izvršavanja aplikacije.
- - DesignTime - generišu kod koji se kasnije koristi u aplikaciji.

2.4 Entity Framework

Entity Framework je delo Microsoft korporacije koji u velikoj meri olakšava razvoj web aplikacija. Kreiranje konekcija, vođenje računa o krajnim rezultatima, upisu u bazu, izmeni podataka i brisanju istih iz pojedinih tabela, o svemu tome sada brine ovaj okvir.

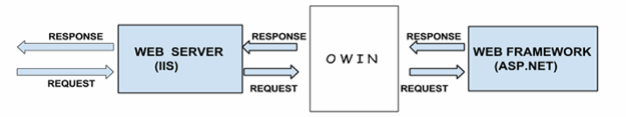
Entity Framework je objektno orijetisan razvojni okvir otvorenog koda koji programerima omogućava rad sa objektima koji imaju vezu sa podacima u bazi podataka bez potrebe fokusiranja na tabele i kolone gde se ti podaci nalaze. Omogućava kreiranje aplikacija sa mnogo manje ručno pisanog programskog koda. Slika 2 prikazuje poziciju razvojnog okvira u aplikaciji [6]:



Slika 2: *Pozicija Entity Framework-a u web aplikaciji* [6]

2.5 Owin

Owin predstavlja standardni interfejs između .Net web servisa i web aplikacije. Glavni cilj Owin-a je da razdvoji server i aplikaciju, podstakne razvoj jednostavnih modula za razvoj .Net web aplikacije i slično. OAuth autorizacija omogućava aplikacijama ograničen pristup Http servisima.
Slika 3 prikazuje cilj Owin-a, a to je da razdvoji server od aplikacije [7].

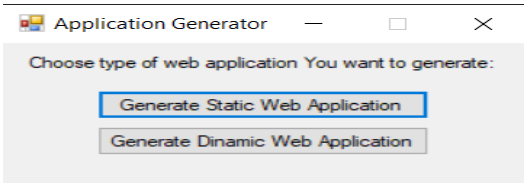


Slika 3: Owin razdvaja server i aplikaciju [7]

3. PRIKAZ IMPLEMENTIRANOG REŠENJA

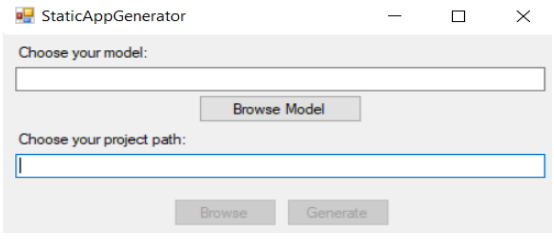
U okviru ovog poglavlja biće detaljno opisan proces generisanja programskog koda za statičke web aplikacije a zatim i za dinamičke web aplikacije. Osnovna razlika statičkog i dinamičkog web sajta je u tome ko ima prava da menja sadržaj samog sajta.

Pod statičkom web aplikacijom se podrazumeva aplikacija čiji sadržaj može da menja i nadograđuje samo stručno lice, kao što je web dizajner ili programer, dok se pod dinamičkom web aplikacijom podrazumeva web aplikacija čiji sadržaj može da menja svaki korisnik računara.
Pokretanjem implementiranog generatora prikazuje se korisnički interfejs kao na slici 4 ispod.



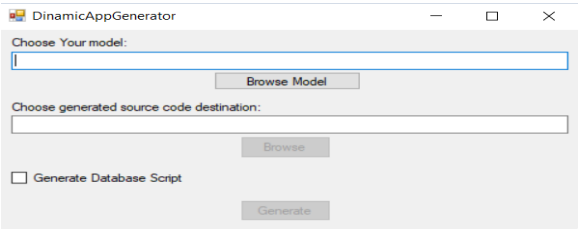
Slika 4: Korisnički interfejs generatora web aplikacija

Kao što se to može videti na slici 4, korisnik može odabrati generisanje statičke ili dinamičke web aplikacije. Klikom na dugme *Generate Static Web Application* otvara se prozor kao što je to prikazano na slici 5.



Slika 5: Generator za generisanje statičke web aplikacije

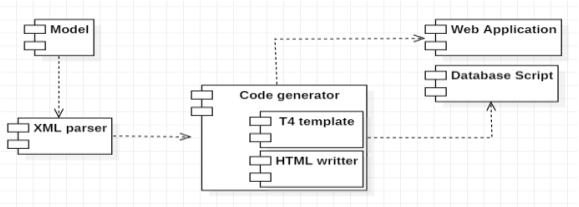
Klikom na dugme *Generate Dinamic Web Application* otvara se prozor kao što je to prikazano na slici 6.



Slika 6: *Generator za generisanje dinamičke web aplikacije*

3.1 Programski kod generatora i generisani programski kod

Najbolji način prikaza funkcionisanja generatora je upoznavanje sa njegovom arhitekturom. Slika 7 predstavlja arhitekturu generatora web aplikacija.



Slika 7: *Arhitektura generatora programskog koda*

Na slici 7 se može videti koje su to komponente koje predstavljaju ulaz u generator, a koje izlaz. Nakon što je generatoru prosleđen model, on prolazi kroz XML parser, budući da se radi o modelu u XML formatu. Kada je isparsiran, prolazi kroz T4 tekstualni templejt i kroz HTML writter kako bi se izvršilo generisanje fajlova potrebnih za pravilan rad web aplikacije. Jedan od izlaznih fajlova može biti i .sql fajl koji predstavlja skript za kreiranje baze podataka.

Da bi opis rada generatora bio potpun sledi detaljnije objašnjenje generisanja programskog koda. Najpre će biti prikazan programski kod generatora a zatim generisani programski kod dinamičke web aplikacije. Generator je implementiran na način da najpre generiše HTML stranice. Za generisanje HTML stranica korišćena je klasa *HtmlTextWriter* koja umnogome olakšava generisanje krajnjeg HTML koda korišćenjem C# programskog jezika. Slika 8 daje prikaz koda generatora.

```
using (HtmlTextWriter writer = new HtmlTextWriter(stringWriter))
{
    writer.RenderBeginTag(HtmlTextWriterTag.Html);

    writer.AddAttribute("runat", "server");
    writer.RenderBeginTag(HtmlTextWriterTag.Head);

    writer.RenderBeginTag(HtmlTextWriterTag.Title);
    writer.Write(myAppName);
    writer.RenderEndTag(); //End title tag
    writer.WriteLine();

    writer.WriteBeginTag("link");
    writer.WriteAttribute("rel", "stylesheet");
    writer.WriteAttribute("href", "Styles/StyleSheet.css");
    writer.WriteAttribute("type", "text/css");
    writer.Write(HtmlTextWriter.SelfClosingTagEnd); //End link tag
    writer.WriteLine();

    writer.AddAttribute("ID", "head");
    writer.AddAttribute("runat", "server");
    writer.RenderBeginTag("asp:ContentPlaceHolder");
    writer.RenderEndTag(); //End asp:ContentPlaceHolder tag
    writer.WriteLine();

    writer.RenderEndTag(); //End head tag
    writer.WriteLine();

    writer.RenderBeginTag(HtmlTextWriterTag.Body);
```

Slika 8: *Prikaz programskog koda generatora za generisanje HTML stranice korišćenjem HtmlTextWriter*-a

Pokretanjem ovog koda, generator će izgenerisati HTML programski kod koji je prikazan na slici 9.

```
<html>
    <head runat="server">
        <title>
            DinamicTest
        </title>
        <link rel="stylesheet" href="Styles/StyleSheet.css" type="text/css" />
        <asp:ContentPlaceHolder ID="head" runat="server">
        </asp:ContentPlaceHolder>

    </head>
    <body>
```

Slika 9: *HTML programski kod izgenerisan pokretanjem generatora*

Ovim je dat kratak primer i objašnjenje generisanja koda koji se tiče izgleda stranica. Sledi prikaz i objašnjenje koda generatora koji kreira C# klase. Za ovo je korišćen T4 templejt.

Korišćenje ovog templejta biće objašnjeno kroz generisanje jedne od metoda koja se koristi pri komuniciranju sa bazom podataka. Na slici 10 je dat prikaz tog koda u T4 templejtu.

```
20   foreach (XmlNode xmlNodeTableName in doc.GetElementsByTagName("name"))
21   {
22       string modelName = xmlNodeTableName.InnerText;
23   #>
24   using <#= myAppName #>;
25   using System;
26   using System.Collections.Generic;
27   using System.Linq;
28   using System.Web;
29
30   namespace <#= myAppName #>.Models
31   {
32       public partial class <#= modelName #>Model
33       {
34           public string Insert<#= modelName #>(<#= modelName #> <#= modelName.ToLower() #>)
35           {
36               try
37               {
38                   <#= dbName #>Entities db = new <#= dbName #>Entities();
39                   db.<#= modelName #>s.Add(<#= modelName.ToLower() #>);
40                   BeforeInsert(<#= modelName.ToLower() #>);
41                   db.SaveChanges();
42
43                   return <#= modelName.ToLower() #>.ID + " was succesufully inserted.";
44               }
45               catch (Exception e)
46               {
47                   return "Error: " + e;
48               }
49           }
```

Slika 10: *T4 templejt za generisanje Insert metode*

Slika 10 daje prikaz koda generatora koji kreira Insert metodu koja komunicira sa bazom podataka i vrši upis novih podataka u bazu. Na samom početku slike 10 se vidi *foreach* petlja koja omogućava kreiranje Insert metoda za svaki model iz gramatike. Kod koji se nalazi posle odrednice #> će kao takav da se nađe u izgenerisanoj metodi. Svaka promenljiva koja se nađe u okviru tagova <#= i #> biće zamenjena vrednošću te promenljive u metodi koja se nađe u generisanoj web aplikaciji. Slika 11 potvrđuje to dajući prikaz generisane Insert metode.

```
using DinamicTest;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;


namespace GeneratedDinamicWebSite.Models
{
    0 references
    public partial class ProductModel
    {
        0 references
        public string InsertProduct(Product product)
        {
            try
            {
                GarageEntities db = new GarageEntities();
                db.Products.Add(product);
                BeforeInsert(product);
                db.SaveChanges();

                return product.ID + " was succesufully inserted.";
            }
            catch (Exception e)
            {
                return "Error: " + e;
            }
        }
```

Slika 11: *Generisan programski kod*

Kao što se može videti, generisana je parcijalna klasa. To omogućava korisniku da nakon generisanja programskog koda doda i svoj, ručno pisan kod. Metoda *BeforeInsert*, ovde pozvana, parcijalna je metoda.

Ukoliko korisnik naknadno ne kreira ovu metodu, kompajler će ignorisati taj deo koda i programski kod će biti kompajliran bez problema.

Ukoliko, pak, korisnik kreira tu parcijalnu metodu, kompajler će uzeti u obzir novu implementaciju *BeforeInsert* metode. Nakon ponovnog pokretanja generatora, ručno pisani kod koji je korisnik naknadno pisao neće biti "pregažen" generisanim kodom.

4. ZAKLJUČAK

U ovom radu prikazano je generisanje statičkih i dinamičkih web aplikacija. Kreiranjem ovakvog generatora posao razvoja aplikacija umnogome je olakšan i ubrzan.

Na krajnjem korisniku je da personalizuje generisanu web aplikaciju u skladu sa svojim potrebama i željama ili da je koristi onako kako je generator izgenerisao. Generator daje potpuno funkcionalan programski kod koji je direktno upotrebljiv bez naknadnih intervencija.

Za implementaciju je korišćen interni domenski specifičan jezik razvijen u okviru XML formata koji je detaljno objašnjen. Kroz primer koji je predočen u ovom radu, moglo se videti kako naizgled jednostavan DSL može dovesti do implementacije jedne potpuno ispravne i kompleksne web aplikacije.

Aplikacija koja je izgenerisana na način ovde prikazan omogućava korisniku da kombinuje generisani programski kod i ručno pisani kod ukoliko postoji potreba za daljim izmenama aplikacije.

5. LITERATURA

[1] Wikipedia, Domenski specifični jezici, https://en.wikipedia.org/wiki/Domain-specific_language

[2] Igor Dejanović, Kurs iz predmeta Jezici specifični za domen, http://www.igordejanovic.net/courses/jsd/uvod.html#7

[3] Metodologija razvoja softvera, http://www.tfzr.uns.ac.rs/Content/files/0/MRSLab03%20-%20PDF%20-%20R1.pdf

[4] Dijagram aktivnosti, http://www.vps.ns.ac.rs/Materijal/mat3542.pdf

[5] Microsoft Documents, Code Generation And T4 Templates, https://docs.microsoft.com/en-us/visualstudio/modeling/code-generation-and-t4-text-templates?view=vs-2019

[6] Entity Framework Tutorial, https://www.entityframeworktutorial.net/what-is-entityframework.aspx

[7] Owin, http://owin.org/ Jignesh Trivedi, Token Based Authentication Using ASP.Net Web API, OWIN and Identity With Entity Framework, https://www.c-sharpcorner.com/UploadFile/ff2f08/token-based-authentication-using-Asp-Net-web-api-owin-and-i/

6. BIOGRAFIJA

Nikola Baštovanović rođen je 26. januara 1993. godine u Bajinoj Bašti, Republika Srbija. Osnovnu školu „Rajak Pavićević" i gimnaziju „Josif Pančić" završio je u Bajinoj Bašti sa odličnim uspehom. Fakultet tehničkih nauka u Čačku, smer Informacione tehnologije modul Softversko inženjerstvo upisao je 2012. godine. Akademske studije završio je 2016. godine sa prosečnom ocenom 9.56. Iste godine upisuje master akademske studije na Fakultetu tehničkih nauka u Novom Sadu, smer Softversko inženjerstvo i informacione tehnologije.