DOI 10.34185/1562-9945-1-126-2020-12
UDC 004.4

A.V. Kravets, I.V. Ponomarev

THE METHODOLOGY FOR DEVELOPING WEB APPLICATIONS ON THE PLATFORM ASP.NET CORE

Abstract. The stack of web technologies today is very diverse. When choosing tools for developing a highly loaded web portal, it is important to pay attention to many factors, for example: modularity, the ability to deploy in the cloud, dependency injection, and so on. The technique of using the cross-platform .NET Core environment for developing large web portals and web services is considered.
Keywords: ASP.NET Core, .NET Core, Cross-Platform, MVC, Dependency Injection, Razor, Architecture, Entity Framework.

Formulation of the problem. ASP.NET Core framework works on the basis of the cross-platform environment .NET Core, which can be deployed on the main popular operating systems: Windows, Mac OS, Linux. And thus, using ASP.NET Core it is possible to create cross-platform applications. In addition to this, Microsoft has created the Kestrel cross-platform web server.

ASP.NET Core is modular and extensible. The framework is built from a set of relatively independent components [1]. The ASP.NET Core platform includes an MVC template that controls the form of an ASP.NET web application and the interactions between its components.

Purpose of the research. ASP.NET Core provides a huge selection of tools for implementing web applications. It is necessary to develop a methodology for constructing websites using the main features of this framework.

Main part. The main advantages of ASP.NET Core are as follows (Fig. 1).

• MVC architecture.

The ASP.NET Core framework helps with the development of web applications that can be more precisely tested, providing a clear separation of

functionality. At the same time, development, compilation and testing in a model, view or controller is simplified.

• Functionality of Razor Pages.

Razor Pages is a new ASP.NET Core element that makes web-based scripting software more productive.

Razor views are HTML templates that contain C # logic, which is used to process model data to generate dynamic content that responds to changes in the model.

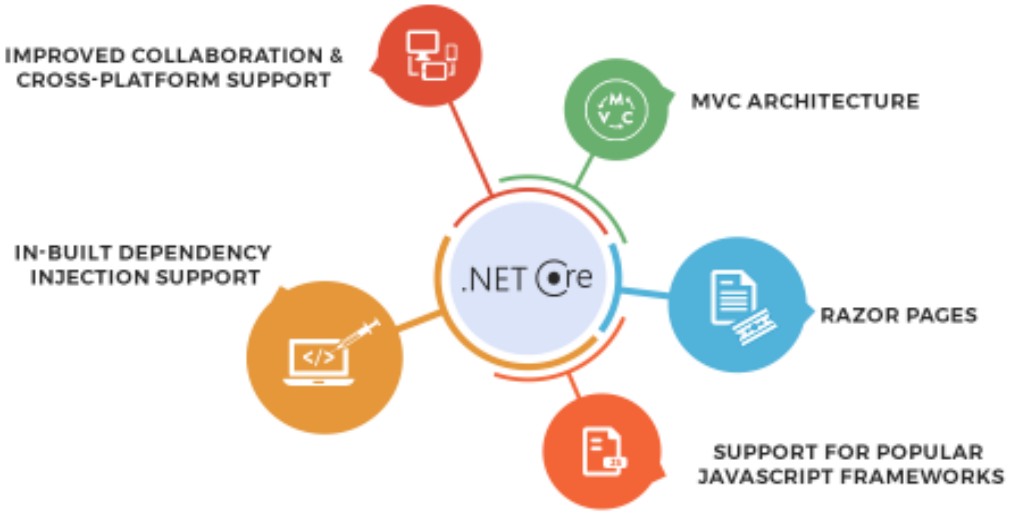

Figure 1 – Key Benefits of ASP.NET Core

• Providing support for popular JavaScript frameworks.

Unlike ASP.NET MVC, the .NET Core framework provides built-in templates for the two most popular JavaScript frameworks - Angular and React.

• Improved team development and cross-platform support.

ASP.NET Core is a cross-platform framework. Applications built using this framework can run on Windows, Linux, and Mac operating systems. Developers can work in different operating systems and at the same time they can still work together on the same project.

• Native support for dependency injection.

The ASP.NET Core framework provides built-in support for Dependency Injection. This means that you no longer need to use third-party frameworks such as Ninject or AutoFactor. Dependency Injection is, in fact, a tem-

SSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

plate that can help a developer isolate different parts of their application, while improving testing capabilities and scalability [1].

ASP.NET Core platform architecture (Fig. 2).

As part of the clean ASP.NET Core architecture, the user interface layer works with the interfaces that are defined in the application kernel at compile time, and ideally should not know anything about the types of implementation defined in the infrastructure layer. However, at runtime, these types of implementation are necessary for the application to run, so they must exist and be bound to the application kernel interfaces through dependency injection [3].

Creating an application on ASP.NET Core.

Creating an ASP.NET Core web application using the MVC pattern consists of the following sequence of steps.

1. Creating the components of an MVC application. In an MVC project, you must follow the conventions of project structuring and file naming.

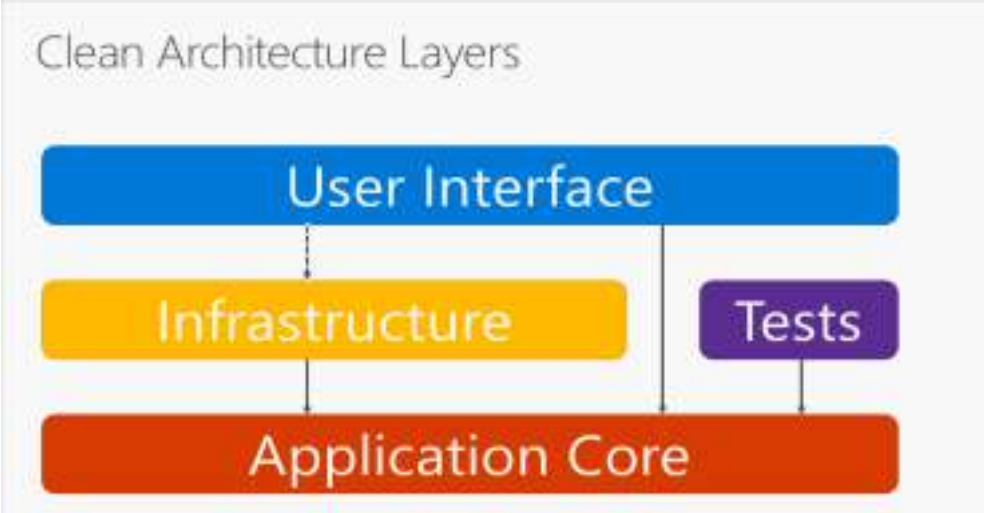

Figure 2 – ASP.NET Core Clean Architecture Diagram

1) Designing data models.

The data store for these models is typically MS SQL Server. To work with MS SQL Server it is recommended to use the Entity Framework ORM technology. The advantage of the Entity Framework is that it allows you to abstract from the structure of a specific database and conduct all operations with data through the model. To interact with the database, you must define the data context. Entity Framework Core uses the Code First approach, which

first defines the models and context of the data, and then, based on these models and the context class, the database and all its tables will be created. To connect to the database, connection parameters are set in the appsettings.json file.

2) Creating a controller.

The whole point of the web application development platform is to design and display dynamic output. In the framework of MVC, the controller's work is to prepare the data and transmit it to the presentation, which is responsible for their visualization in the form of an HTML markup.

3) Creation and visualization of the presentation.

When you create a project, a client-side development package Bootstrap is installed, which is a convenient CSS infrastructure. The view is styled after importing the Bootstrap stylesheets by using its CSS classes.

2. Application configuration.

The Program and Startup classes and JSON files are used to configure the operation of the application, as well as specify the packages on which it depends. The configuration system allows you to tailor applications to their environments and manage package dependencies. By default, the Program class starts application execution:

```
public class Program
{
  public static void Main(string[] args)
  {
   CreateHostBuilder(args).Build().Run();
  }
 public static IHostBuilder CreateHostBuilder(
      string[] args) =>Host.CreateDefaultBuilder(args)
      .ConfigureWebHostDefaults(webBuilder =>
     {
       webBuilder.UseStartup<Startup>();
     });
}
```

To run an ASP.NET Core application, an IHost object is required within which the web application is deployed. To create an IHost, the IHostBuilder object is used. In the default program, the static method CreateHostBuilder

SSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

just creates and configures IHostBuilder. Direct creation of IHostBuilder is done using the Host.CreateDefaultBuilder (args) method. This method sets the configuration of the host, application, adds logging providers. Next, the ConfigureWebHostDefaults() method is called. This method is designed to configure host parameters.

The application start class is the Startup class, from which the processing of incoming requests will begin.

Dependency injection allows you to make objects interacting in the application loosely coupled. Objects are interconnected through an abstraction layer, for example, through interfaces, which makes the entire system more flexible, more adaptable and expandable. The ConfigureServices() method is responsible for installing services in the application:

```
public void ConfigureServices(IServiceCollection services)
{
     // Add framework services.
 services.AddDbContext<ApplicationDbContext>(
     options =>options.UseSqlServer(Configuration
     .GetConnectionString("DefaultConnection")));
services.AddIdentity<ApplicationUser,IdentityRole<Guid>>()
   .AddEntityFrameworkStores<ApplicationDbContext, Guid>()
   .AddDefaultTokenProviders();
 services.AddMvc();
     // Add application services.
 services.AddTransient<IEmailSender, AuthMessageSender>();
 services.AddTransient<ISmsSender, AuthMessageSender>();
}
```

After creating the services, ASP.NET calls the Configure() method. The Configure() method configures the query pipeline, which is a set of components (called middleware) used to process incoming HTTP requests and generate responses to them.

3. Creation and registration of routes.

Routes are created using a lambda expression, passed as an argument to the UseMvc() configuration method. They are defined using the MapRoute() extension method, which receives explicitly named arguments name, template, and defaults. The name argument specifies the name of the route, the

template argument is specified in the template argument, and default values are specified in the defaults argument.

4. Application deployment.

Web applications are increasingly hosted in small and simple containers on cloud platforms. With Visual Studio, an application can be easily deployed to Azure [2].

Conclusions. Today, ASP.NET Core contains many tools that make it possible to create a flexible, modular, easily maintained and cross-platform web application. And constant updates of independent modules and the addition of new functionality makes this framework a leader in its field of web development. A methodology for creating web applications using the main features of the ASP.NET Core framework has been developed.

REFERENCES

1. Choosing between .NET Core and .NET Framework for server apps / [Electronic resource]. – Access mode: https://docs.microsoft.com/en–us/dotnet/standard/ choosing–core–framework–server.

2. Publish an ASP.NET Core app to Azure with Visual Studio [Electronic resource]. – Access mode:
https://docs.microsoft.com/en–us/aspnet/core/tutorials/ publish–to–azure–webapp–using–vs?view=aspnetcore–3.1.

3. Common web application architectures Studio [Electronic resource]. – Access mode: https://docs.microsoft.com/en–us/dotnet/architecture/modern–web–apps–azure/common–web–application–architectures.

Received 24.01.2020.
Accepted 28.01.2020.

Методика розробки веб-додатків на платформі ASP.NET Core

На сьогоднішний день для розробки великих веб-порталів і веб-сервісів Microsoft пропонує відмінне рішення - кроссплатформенную середу .NET Core. Запропоновано алгоритм дій, який спрощує розробку гнучкого, модульного, легко супроводжуваного і кроссплатформенного веб-додатку.

Методика разработки веб-приложений на платформе ASP.NET Core

На сегоднящний день для разработки крупных веб-порталов и веб-сервисов Microsoft предлагает отличное решение - кроссплатформенную среду .NET Core. Предложен алгоритм действий, который упрощает разработку гибкого, модульного, легко сопровождаемого и кроссплатформенного веб-приложения.

SSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Кравец Андрей Викторович - студент гр. КИ-16-1 кафедры ЭВМ Днепров-ского национального университета имени Олеся Гончара.
Пономарев Игорь Владимирович - доцент, к.т.н., доцент кафедры ЭВМ Днепропетровского национального университета им. О. Гончара.

Кравець Андрій Вікторович - студент гр. КІ-16-1 кафедри ЕОМ Дніпровського національного університету імені Олеся Гончара.
Пономарьов Ігор Володимирович – доцент, к.т.н., доцент кафедри ЕОМ Дніпропетровського національного університету ім. О. Гончара.

Kraves Andrey - student gr. KI-16-1, Computer Engineering Department, Oles Honchar Dnipro National University.
Ponomarev Igor - candidate of technical sciences, associate professor of the department of electronic computers of the faculty of physics electronics and computer systems of the Oles Honchar Dnipro National University.