ВВЕДЕНИЕ В ASP.NET CORE

и. т. н. ***Бакунова О. М.,***
Тимофеев Д. О.,
Уласович В. Ю.,
Халецкий Д. М.,
Ухналев Р. Ю.,
Михаленко В. А.

Республика Беларусь, г. Минск, Институт информационных технологий
Белорусского государственного университета информатики и радиоэлектроники

ARTICLE INFO
Received 01 March 2018
Accepted 15 March 2018
Published 12 April 2018

KEYWORDS
.NET Core,
ASP. NET,
library,
application,
development,
the modern platforms of
programming

ABSTRACT
Ways of developing various programs and web applications are becoming newer. Historically, there were several types of platforms (Windows, Linux, Mac OS) for development, and developers created their products separately for each platform, because restrictions prevented them from developing everything at once. Now, there is an active trend to make work of developers easier for large number of users. And even Microsoft, who didn't particularly welcome cross-platform and Open Source, changed their opinion on that. Their .NET platform, since its introduction in 2002, has come a long way, becoming one of the best solutions in the enterprise. But it offered to development only for devices running Windows. A new round of the platform .NET - .NET Core has become a serious step for the consolidation of development methods.

Введение. Новым шагом в эволюции ASP.NET является ASP.NET Core, чья разработка велась гораздо более открыто, по сравнению с предшественниками, и распространялась свободно на GitHub. Такой способ разработки должен обеспечить высокую конкурентоспособность ASP.NET по сравнению с наиболее актуальными фреймворками современности (Go, Elixr, Node). ASP.NET Core серьезно связан с другой инициативой по модернизации исполнительной среды .NET, называемой .NET Core. Разработка .NET Core направлена на огромный ряд усовершенствований, важнейшим из которых является поддержка .NET Core не только в Windows, но и в других операционных системах. Таким образом, вы можете разработать свое приложение в Linux или macOS.

Результаты и обсуждение. .NET Core имеет модульную архитектуру. То есть, компилятор, библиотеки и сама исполняющая среда являются отдельными сущностями, взаимодействующими через четко спроектированные интерфейсы, что позволяет гибко взаимодействовать с компонентами для конкретных задач. .NET Core является производительным, и он создает издержки только за действительно используемые абстракции.

На данный момент существует четыре варианта написания кода под .NET Core: кроссплатформенные web-приложения ASP.NET, кроссплатформенные консольные приложения, кроссплатформенные библиотеки и инфраструктуры, UWP-приложения.

Одним из наиболее важных нововведений .NET Core по сравнению с его предшественниками является полноценная поддержка macOS и различных дистрибутивов Linux. В этом плане перед огромным числом разработчиков открылась возможность разрабатывать кроссплатформенные приложения, одинаково работающие на всех платформах.

Различия между кроссплатформенными библиотеками и инфраструктурами заключаются в степени масштабирования. Библиотеки являются одним из возможных способов программирования в .NET Core. При этом, при больших масштабах инфраструктуры для распределенных вычислений тоже являются хорошим вариантом для разработки.

У кроссплатформенных консольных приложений область применения шире, чем кажется многим разработчикам. Те же UWP-приложения, выполняемые в .NET Core, ориентируются на устройства с Windows 10. С расширением инструментария разработчики смогут создавать еще больше программ, направленных на решение различных задач.

ASP.NET Core включает фреймворк MVC. Непосредственно сам MVC также включает различные функциональности, такие как: Web Pages, Web API, и непосредственно самого шаблона MVC. Применение этого шаблона в разрабатываемых программах не считается строго обязательным условием, можно использовать чистый ASP.NET Core, на котором и будет выстраиваться вся логика, но для более понятного разделения логики программы, и самой реализации программы, использование шаблона MVC является стандартом при разработке веб-приложений ASP..NET Core.

Сама структура шаблона MVC разделяется на три следующих компонента: модель (model), представление (view), контроллер (controller).

Модель берет на себя часть ответственности логики, где описываются используемые в приложение данные, а также саму логику данных. Также в модели применяется логика валидации. Сама модель включает в себя два подвида: модель представления и модель домена. Модель представления необходима для отображения и передачи данных, а непосредственно модель домена описывается управления данными.

Представление представляет собой пользовательский интерфейс. Часто используется язык-разметки html для отображение данных.

Контроллер обеспечивает связь между пользователем и приложением. Также он является хранилищем и представлением данных. Контроллер также содержит логику обработки запроса пользователя, то есть в него приходят данные, которые вел пользователь через представление (view) и обрабатывает их. В зависимости от результатов, контроллер отправляет пользователю определенный вывод, на представление с наполненными данными из моделей.

Данный шаблон помогает разделить ответственность логики, что позволяет разрабатывать более гибкие приложения с помощью ASP.NET Core MVC.

В .NET Core появилась возможность упаковки и развертывания приложения с использованием пакета целевой среды. Хранилище пакетов среды выполнения существенно экономит дисковое пространство и повышает производительность. Для использования всех преимуществ нового хранилища используется метапакет Microsoft.AspNetCore.All. Изменена конфигурация экземпляр IConfiguration по умолчанию добавляется в контейнер служб.

Изменения коснулись ведения журналов. В .NET Core 2 по умолчанию ведение журнала включено в систему зависимостей. Поставщики и фильтрация выполняются в файле Program.cs.

Была упрощена сборка веб-API удостоверений в ASP.NET Core. Маркеры доступа веб-API теперь можно получить с помощью библиотеки проверки подлинности MSAL. Включено создание QR-кодов проверки подлинности приложений. Используются два фактора проверки пользователя, с помощью синхронизированного одноразового пароля TOTP алгоритма.

Сервер Kastl приобрел новые функции. Добавлено свойство Limits для класса KastelSereverOtions, который позволяет ограничить число клиентских подключений, размер запроса и скорость передачи содержимого запроса.

Осуществлено расширение поддержки заголовков HTTP. Переименован пакет WebListener в HTTP.sys.

Выводы. Действия, направленные на некоторую стандартизацию .NET API, очень важны, отчего они и активизировались. Предполагается, что наличие кроссплатформенности приведет к принятию .NET в тех секторах рынка, которые традиционно не являлись дружественными к технологии .NET. Многие недоброжелатели, не приветствующие политику закрытого исходного кода, свойственную для Microsoft, вынуждены пересматривать свои взгляды, так как большая часть стека технологий (включая компилятор и BCL), распространяется с открытым кодом. Теперь операционная система, под которой работает или только разрабатывается ваше приложение, не является определяющим фактором при выборе .NET. А сама платформа .NET Core стала новым шагом эволюции .NET Framework, уже начав вытеснять старые методы разработки.

ЛИТЕРАТУРА

1. Адам Фримен ASP.NET Core MVC с примерами на C# для профессионалов, 6-е издание: Пер. с англ. — СпБ. : ООО “Альфа-книга”, 2017 - С.19-24

2. Джеймс Чамберс, Дэвид Пэкетт, Саймон Тиммс ASP.NET Core. Разработка приложений — СпБ.: Питер, 2018 - С.38-39, 130-142