CONVERTING ASP.NET WEB FORM WEBSITES TO ASP.NET CORE 2.0 MVC WEBSITES
**ABSTRACT**

<table>
<thead>
<tr>
<th>Centria University of Applied Sciences</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>December 2018</td>
<td>Jyri Mäkelä</td>
</tr>
</tbody>
</table>

**Degree programme**  
Information Technology

**Name of thesis**  
CONVERTING ASP.NET WEB FORM WEBSITES TO ASP.NET CORE 2.0 MVC WEBSITES

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kauko Kolehmainen</td>
<td>20 + 2</td>
</tr>
</tbody>
</table>

**Supervisor**  
Kauko Kolehmainen

The aim of this thesis is to give guidance on conversion process focusing on converting ASP.NET Web Forms websites to ASP.NET Core 2.0 MVC websites. At the time of writing this thesis, there were no tools for automated conversion, so manual conversion is the only solution. Thesis goes through what is taken into account in conversion process and planning. The thesis gives information about changes in technology, design patterns and code.

Theory part has information about the technologies used in the thesis and example conversion tools. Practical part goes through conversion planning and the actual process. Results are shown at the end of the practical part. Conclusion has the analysis of the success of the thesis and some ideas about continuing the development of the project. Example project is converting a meeting room reservation web application to ASP.NET Core 2.0 MVC. Kosila is the commissioning party for this thesis. Kosila gave the project and the topic for the thesis.

**Key words**  
ASP.NET, Conversion, ASP.NET Core 2.0 MVC.
ABSTRACT

CONTENTS

1 INTRODUCTION ................................................................................................................................ 1

2 TECHNOLOGIES USED IN THIS WORK .......................................................................................... 2
  2.1 ASP.NET ......................................................................................................................................... 4
  2.2 Web Forms ...................................................................................................................................... 4
  2.3 Model-View-Controller .................................................................................................................. 4
  2.4 ASP.NET Core................................................................................................................................ 5
  2.5 Converting code from source language to target language ........................................................ 5
  2.6 Software conversion process ......................................................................................................... 6

3 FINDING TOOLS OR WAYS TO CONVERT THE CODE .............................................................. 8

4 PRACTICAL WORK .......................................................................................................................... 10
  4.1 Installing the development environment .................................................................................... 10
  4.2 Creating a ASP.NET Core 2.0 MVC project in Visual Studio 2017 ....................................... 11
  4.3 Conversion plan for the project .................................................................................................. 13
  4.4 Starting on meeting room reservation web app ........................................................................ 14
  4.5 Conversion process....................................................................................................................... 14
  4.6 Conversion result ......................................................................................................................... 18

5 CONCLUSION .................................................................................................................................. 20

REFERENCES ...................................................................................................................................... 21

FIGURES
FIGURE 1. .NET Standard library architecture .................................................................................... 3
FIGURE 2. Direct Native-Compilation from .NET Source Code ............................................................ 7
FIGURE 3. DMS flowchart ..................................................................................................................... 9
FIGURE 4. Installing workloads in Visual Studio 2017 ...................................................................... 11
FIGURE 5. Creating an ASP.NET Core Web Application project ....................................................... 12
FIGURE 6. Choosing MVC design for the project .............................................................................. 13
FIGURE 7. Part of a scaffold controller showing the database context implementation .................... 15
FIGURE 8. Rooms model showing the get and set functions ............................................................... 16
FIGURE 9. Default website sample ..................................................................................................... 17
FIGURE 10. Startup.cs services configuration ...................................................................................... 17
FIGURE 11. Startup.cs: routing and HTTP request pipeline configuration ......................................... 18
FIGURE 12. User interface for the ReservationWebApp when there are no reservations ................. 19
1 INTRODUCTION

About 13 percent of all websites still use ASP.NET and usage is slowly dropping, according to w3techs.com. Microsoft developed ASP.NET and released the initial version 1.0 on January 5, 2002 and the newest version 4.7.1 released on October 17, 2017. ASP.NET is the successor of the Microsoft’s Active Server Pages (ASP). It is built on Common Language Runtime (CLR), which makes it able to create websites with programming language of one’s choosing, as long as it supports CLR. First supported languages were C# and Visual Basic, now there are many more. ASP.NET’s main website design framework was Web Forms, now there are many other frameworks like MVC (Model-View-Controller), WebAPI and more. (W3Techs. 2018.)

ASP.NET Core 2.0 is the newest major version that was released in August 14, 2017, first version was introduced on June 27, 2016. Its framework is completely rewritten from ASP.NET MVC and ASP.NET WebAPI into a single programming model. ASP.NET Core is faster than its predecessor and natively supports cross-platform ASP.NET Core applications. Therefore, it can create applications on Linux and run them on Windows or Mac as long as they are running ASP.NET Core. ASP.NET Core is open-source. Many companies are looking into updating their websites to ASP.NET Core from ASP.NET. Migration of the old working websites can be costly, especially large websites, if it must do them from scratch to a new framework. Aim of this thesis is to find out if there are any tools or ways to make the migration faster, easier or to save costs. Quality of the conversion is dependent on the quality of the conversion tool. With good quality tools, it does not have to manually fix any part of the code.

Kosila is a Finnish software company with offices located in Kokkola, Vaasa, Helsinki and Tampere. Kosila is part of the Keski-Pohjanmaan Kirjapaino PLC, which employed 349 workers in 2017. Kosila employs thirty workers. Kosila provides websites, web applications, mobile applications, project planning and designing and many more services using Microsoft technologies. They are focused on providing digital business solutions. Kosila is the commissioning party of this thesis.

This thesis consists of theory part, practical part and conclusion. Chapters two to three are part of the theory, forth chapter is practice part and fifth chapter is the conclusion. Conclusion is that conversion takes huge amount of manual conversion work, since there are no tools available. References are found at the end of the thesis.
2 TECHNOLOGIES USED IN THIS WORK

.NET is a mature development environment; it is developed by Microsoft. It is also free, cross-platform and open source. .NET has multiple integrated developer environments (IDE), most popular ones are Visual Studio, Visual Studio Code and .NET Core Command-line interface (CLI). It has tools for making web, desktop, mobile, games, libraries, machine learning, artificial intelligence (AI) and internet of things (IoT) apps. .NET also supports multiple programming languages, main ones are C#, F# and Visual Basic. More languages can be added through extensions, as long as they support Common Intermediate Language (CIL). Some of the tools support cross-platform; some are for specific operating systems (OS), like Xamarin for mobile app development. .NET Core provides main support for cross-platform web, server or console apps on Windows, Linux or Mac. Internet of things are small devices that provide software utility for users, like wireless light switches or wireless weather stations. (Microsoft. 2018.)

.NET Standard gives a common base set application-programming interface (API) for .NET implementations. It makes developing in .NET environment feel uniform. Libraries written according to .NET Standard are usable with many of the tools (FIGURE 1). .Net Standard was released on June 27, 2016. Newest version of the standard is 2.0. (Microsoft. 2018.)
.NET uses CLR to compile the source code into Intermediate Language (IL). After the source code has been compiled into IL, it is then compiled to platform-specific native code by Just-In-Time (JIT) compiler. Compiler translates the source code into a code that the machine can understand. .NET Core uses CoreCLR and .NET Framework uses CLR. Current versions of CLR and CoreCLR use new version of JIT, called RyuJIT. The new compiler is faster and adds more debugging capabilities to Visual Studio. All versions of the CLR have Garbage Collector (GC). GC cleans memory from unused data. Threads are also handled by the runtime. (Nagel. 2018. 16).

C# (C sharp) programming language will be the main language used in this thesis. Before C# was created, Microsoft was trying create J++ (Java plus plus) programming language, but the company that created Java filed a court order to disallow Microsoft’s attempts. Then Microsoft created C# and released it with .NET Framework 1.0 back in 2002. Java, C++ and Pascal influenced C#, but since C# was created after these programming languages, Microsoft was able to do things differently and fix many of the errors these other languages had. C# has evolved throughout the years. Newest version 7.3 was released in May 2018. (Nagel. 2018. 5).
2.1 ASP.NET

Microsoft started developing ASP.NET in the late 1990s. It was released in 2002 with .NET framework 1.0. It was developed for desktop developers to make it easier for them to create websites. ASP.NET was a big improvement compared to its predecessor ASP. Web Forms was the first framework for creating websites, later on other frameworks were developed. These other frameworks are WebAPI, SignalIR and Single Page Apps, the focuses on this thesis are Web Forms and MVC frameworks, In March 2009, Microsoft released MVC 1.0 for ASP.NET. MVC was Microsoft’s response to the criticism they had about Web Forms. Latest release of ASP.NET was released on October 2017, with version 4.7.1. (Evjen, Gaylord, Wenz, Rastogi, Miranda, Hanselman & Hunter. 2013. 3-5).

2.2 Web Forms

Web Forms is the oldest framework for developing websites in ASP.NET. With Web Forms they tried to make web development feel like developing a desktop application. Microsoft tried to hide Hypertext Transfer Protocol (HTTP) and Hypertext Markup Language (HTML) from developers, because back then not that many developers knew about these technologies. Web Forms was trying to be stateful while HTTP is stateless. Stateless does not need keep up with session information or status during the duration of multiple requests. Stateful on the other hand sends data every time a state changes, this caused large amounts of data being transferred between the client and the server, which caused longer response times and increased bandwidth usage on the server. Hiding of the HTML can be seen from the server controls. Server control renders the HTML page for the user. This causes some unwanted HTML pages, since you could not edit HTML or Cascading Style Sheets (CSS), you had to reverse-engineer the postback event mechanism or try to use different tricks. Web Forms lacked unit testing which is essential for software development. (Freeman. 2016. 3-4)

2.3 Model-View-Controller

MVC is a design pattern for creating websites in ASP.NET. It was first developed for graphical user interfaces (GUI) in the late 1970s. When GUIs became the norm, it was clear that MVC pattern does not work well with GUIs. Since MVC is more of an event-based design pattern, patterns like Model-View-Presenter (MVP) were better for new systems. Today MVC design pattern is widely used in web development. MVC was first introduced in 2007 for ASP.NET and it was released in 2009. Release added
many improvements that Web Forms lacked, like unit testing which was in high demand by the developer community. Latest version 5.2.5 was released in May 2018. (Galloway, Allen & Matson. 2014. 31-33)

Controllers are responsible for user input and making changes to the model according to the user input. Controllers handle the flow of the application. They take care of data coming in, data processing and sending to the relevant view. View provides user interface for the user. Controller processes user given data and gives information for the requested Uniform Resource Locator (URL), then send it to view to display it to the user (Galloway, Allen & Matson, 31-33). Models are for plain code and classes. Models have the objects of the code. Objects are used for sending information between databases, render views and calculations. (Galloway, Allen & Matson. 2014. 50, 76).

2.4 ASP.NET Core

Microsoft announced ASP.NET Core in 2015 and first released in June 2016. It is built on top of the .NET Core. ASP.NET Core supports cross-platform, so you can develop and use ASP.NET Core applications on Linux and Mac instead of just Windows like ASP.NET Web Forms. ASP.NET Core is completely rewritten from ASP.NET, uniting ASP.NET MVC and WebAPI into one ASP.NET Core MVC. It is Open Source, so you can see and edit the source code, this gives ability to make your own version of ASP.NET Core and make more advanced applications and better debugging. ASP.NET Core supports all ASP.NET Core version apps, before you could not run different .NET framework version apps at the same time on the same machine. (Freeman. 2016. 5-8) (De Oliveira & Bruchet. 2017. 13-17) (Nagel. 2018. 13).

2.5 Converting code from source language to target language

Software has a life cycle. During this life cycle, software developers support and maintain the software as usable as possible. Life cycle of the software is estimated on release. Many variables are taken into account when estimating software life cycle, like usage, upkeep costs, risks and importance. Either when software comes to the end of its life cycle, support ends or it is extended. End of life cycle means that
its developers no longer support software, no more additional features or bug fixing. Bugs are errors in software when software does not run as intended. (Wagner. 2014. 10).

In some cases during and at the end of the life cycle, software source code is converted to another language for better support. Software life cycle might be so long that new technologies have been released. These new technologies might give better features, performance and better compatibility. The cost of the conversion varies according to size and complexity of the software. (Barbier & Recoussine. 2015. 49-50)

2.6 Software conversion process

Software conversion process differs from case to case, there are many programming languages and they differ from each other even between versions. Some code converters try to convert the source code to target language in 1:1 ratio. This causes problems when target language uses strict grammar and you have to fix these problems manually. A compiler is a translator that translates code into machine code or assembly language so the computer understands it. Software modernization is converting legacy code to more modern programming languages and models. Old languages like COBOL are being modernized using modeling techniques like unified modeling language (UML). Old legacy software is first modelled to an UML, and then converted to the target technology or language. BLU AGE is an architecture-driven modernization (ADM) computer-aided software engineering tool (CASE) for COBOL and COBOL like languages. BLU AGE uses Eclipse modeling framework. (Barbier & Recoussine. 2015. 183-184)

.NET Core RunTime is a good example of code translation. .NET IL is processed in the CoreRT Native CodeGen or IL to C++ code that runs on platform specific compiler for C++ code. This allows .NET Core to be run on many current platforms and future platforms. .NET IL is first run in its own compiler, then translated to C++ in CoreRT to the specific platform and then executed on that platform in CoreRT. CoreRT does not translate in 1 to 1 ratio. Since C++ does not support .NET features fully, you need to have an interpreter to determine the correct mappings for C++ features. CoreRT is an ahead of time (AOT) compiler while CoreCLR, .NET Core default compiler, is a JIT compiler (FIGURE 2). They both run RyuJIT. (Fritz. 2016)
FIGURE 2. Direct Native-Compilation from .NET Source Code. (Fritz. 2016)
3 FINDING TOOLS OR WAYS TO CONVERT THE CODE

There are many different tools to convert code to a different language. Some are free to use, and some are commercialized. Quality of the converter is a big factor, because it saves money. If the quality of the conversion is well done, it is only needed to do minor fixes to the code. Before doing any conversion, you have to have a cost analysis. Project needs to estimate the time needed to code the program from scratch versus conversion and fixing possible problems. Some of the free converters are online tools thus are very simple and only convert a single file or source code at a time. Commercial converters have ability to convert whole projects at once. Automated software migration saves time and money compared to manual migration. Size and complexity of the source code and understanding of the source code mandates whether you should migrate automatically or manually. Modular source code is easier it is to convert to the target language. (Red Hat. 2018.) (Semantic Systems. 2016)

Design Maintenance System (DMS) Software Reengineering Toolkit is an automated conversion tool. It supports over 20 languages like HTML, C#, COBOL, SQL and Java. Semantic Designs develops DMS Software Reengineering Toolkit. DMS has a customizable tool builder; you can create your own tool to use for the automated conversion. This helps with quality of the conversion and save time and money. DMS provides source code analysis and modification, and code generation. DMS does not only convert programming languages. It converts markup languages, data descriptions, design notations and domain-specific languages. DMS uses generalized compiler for the conversion process. The compiler has parser that sends parsed code to analyzer that analyzes the parsed code according to the compiler ruleset (FIGURE 3). Analyzer also creates reports. Parsed code is also sent to transformer. Transformer converts the parsed source code to target language, using target languages formatting rules. DMS is mainly aimed at converting software programs that have several million lines of code. (Semantic Systems. 2016)
FIGURE 3. DMS flowchart. (Semantic Systems. 2016)
4 PRACTICAL WORK

Main aim of this thesis is to find out if it is possible to convert from ASP.NET Web Forms to ASP.NET Core 2.0 MVC. I did not find any tools that would do the conversion automatically. (Microsoft, 2018). Only possible way I found is to convert to ASP.NET MVC or ASP.NET WebAPI from Web Forms first (Vogel, 2013). Then it is possible to convert to ASP.NET Core 2.0 MVC. You would have to do every step of the conversion manually. That is not logical to do for small projects, but as project size scales up, it might be worth it to do the conversion. Converting from ASP.NET Web Forms to ASP.NET Core 2.0 MVC is not an easy task to succeed. Straight conversion does not work since ASP.NET Core does not support ASP.NET Web Forms, Web Forms and MVC are two completely different design patterns. Second option is to do the project from the start on ASP.NET Core 2.0 MVC (Levin, 2018). Both options take a lot of time. Although programming the project from the start give you better understanding of the new code. The example project used in this thesis is simple. A meeting room booking web application for Kosila’s intra network. The web application uses a database to keep track of bookings.

4.1 Installing the development environment


To install Microsoft Visual Studio 2017 Community edition, you have to download installation file from https://visualstudio.microsoft.com/free-developer-offers/. After downloading the installation file, run it and install Visual Studio. The Installation installs basic components by default, installer asks what components are needed during installations, and more components can be added later on. For this task, the “ASP.NET and web development” is needed (FIGURE 4), components to install can be selected from
the right. When workloads and components are selected, installation is started by clicking “Install” button from bottom right corner of the window.


#### 4.2 Creating a ASP.NET Core 2.0 MVC project in Visual Studio 2017

Launch Visual Studio, open File menu, and select New Project. In the New Project window, choose Web under Visual C# and choose ASP.NET Core Web Application (FIGURE 5). The project can be renamed to any title on the bottom of the window, when ready click OK. In the next window, choose Web Application (Model-View-Controller) (FIGURE 6). It can be also chosen if authentication is wanted on the application. Click OK once correct project type is chosen. Now programming can be started.
4.3 Conversion plan for the project

Since ASP.NET is not supported with ASP.NET Core MVC, conversion has to be planned. Learning to the code works is critical, before conversion can be started. Since these use different design patterns, it has to be designed for the MVC. The new conversion needs to have the same functionality and features. In this thesis, project is a meeting room reservation web application. It has a calendar feature where date is chosen to see if a meeting room is available and make reservation for that day. When you choose the date, table is shown with the meeting room names as headers and room can be reserved for minimum 30 minutes starting from 8.00 to 21.00. Table shows the name of the reservation, if there is a reservation and if there is no reservation it is blank. When blank area is clicked reservation form opens. In the reservation form, you need to fill out the name of the reservation, start time of the meeting, end time of the meeting and name of the one making the reservation. There needs to be a way to edit reservations and remove them in case of cancellations. Since this project is for the Kosila’s intra network, it does not need to have login features. It is only accessible in the local network at Kosila.
4.4 Starting on meeting room reservation web app

New project can be started couple different ways. It can be started with the database first or the code. Since the old project already has a database set up, database first option is better for this case. This means that you start by doing database related codes first. In MVC, models handle getting the data from a database or sending data to a database. After creating a model, variables are needed for holding data. If database is already set up, there are tools for reverse engineering the database and getting the variables for the model and the database context for the connection. If database is not set up, then you can add the variables in the model and then use scaffolding to create a controller. It generates the controller for the model. Controller can be chosen with create, update, edit and delete functions. Scaffolding will generate these functions into the controller and generate views for them. Once scaffolding is done, migration must be created using Entity Framework. NuGet Package Manager Console has to be accessed to do this in Visual Studio 2017.

4.5 Conversion process

Conversion starts with studying the source code of the original website. Better understanding is gotten from studying the source codes. ASP.NET uses code behind files for the backbone of the code, code behind files is used to create the logic of the website. Aspx files can be checked to find out which code behind file it uses.

After learning how the code works, designing of the conversion can be started. In this case, database is reusable since SQL is still widely used and supported. Design can be started with the models and creating database context. Since database already exists, database first approach can be taken. Scaffolding can be used to reverse engineer the database tables and generate models and database context. This automatically creates the models and database connection string for the database context. This reduces the amount of designing time from model design. After models are done, then it is time to start creating controllers for the project. Controllers handle getting data through models and sending it to a view. Controllers take care of the HTML GET and POST functions as well. Function can be forced to use these using data annotations. At the beginning of the controller, it is required to call for a database context that is going to be used in the controller. If scaffolding is used to create a controller, scaffolding will generate database context and functions based on controller type that is chosen (FIGURE 7). Controller needs to have the
“using ReservationWebApp.Models;” part, so the controller knows that the models exist. These functions give out basic views using data from database created. Views can be used to add, edit and delete data from database. These views can be viewed by running the project in Visual Studio.

Models will mostly have get and set functions for database operations (FIGURE 8). Example rooms model was reverse-engineered from existing database. RoomId is the primary key in the database table. Each table has its own model class. Models are united using database context. Get function reads data from database and set function writes data to database.
Views are cshtml files, so they mainly support HTML, C# and scripts like JavaScript. Using C# in views requires @-tag in front of the C# line. Curved brackets can be used with @-tag for multiple lines of code. Controllers call views for the user to interact with it. User cannot see the code implemented inside the cshtml file. User only sees HTML code when inspecting the page. Original code is converted to HTML. Views use bootstrap and layout file for the style of the website.
Visual Studio has a built-in default website as a sample when you create a new project (FIGURE 9). It uses default layout and bootstrap. Default index page is defined in default controller. You can change this in the default HomeController or create a new controller and change default map route in Startup.cs. Default index page is configured inside app.UseMvc routing configuration.

A controller is chosen and default function inside the controller as action. Startup.cs is the settings file to pipeline all the services and routes for your project (FIGURES 10-11). In services, things are handled like cookie handling, database connection string, routing and many more. These services are also known as middleware.

```csharp
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
    app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();

    app.UseMvc(routes =>
    {
        routes.MapRoute(
            name: "default",
            template: "[controller=Reservations]/[action=Reservation]");
    });
}
```


4.6 Conversion result

Converted website has similar functions as the ASP.NET version. It is a single page web application using MVC. Due to lack of time, Sacrifices had to be done on the features. Web application is missing calendar function for choosing dates. It has the main function of getting reservations for the today and making reservations. Creating, deleting and editing rooms are hidden so only authorized persons can do changes. Authorized persons can edit or delete reservations as well.

User interface is simple to use for the user (FIGURE 12). When user clicks “Make a Reservation”, it takes user to the form where he chooses the room, gives name of the reservation and gives start time and end time. Reservation is saved in the database. Reservation is then shown in the table for other users. Authorized users are able to remove and edit reservations and rooms.
5 CONCLUSION

Conversion process takes a lot of manual conversion work. At the time of writing this thesis, there are no tools for automated conversion. There are two ways to do the conversion from ASP.NET Web Forms, manually converting to ASP.NET Core 2.0 MVC or manually converting ASP.NET MVC or WebAPI and then converting to ASP.NET Core 2.0 MVC. In most cases, it is better to do manual conversion to ASP.NET Core 2.0 MVC, since both take huge amount of manual converting. Doing conversion this way also gives better understanding of the newer technology and how it differs from the older technology. Before starting the conversion, it is good to value, if the conversion is actually needed. Converting a working website to ASP.NET Core 2.0 MVC might cause problems on the way and in the future. Manual converting also takes up more workhours to do which makes the cost higher.

Converted project is not completely done. It is missing features like calendar and other quality of life features. Commissioning party can continue developing the project. Developing an automated conversion tool is useful, if there are big demand for converting ASP.NET Web Form websites. ASP.NET Core 2.0 provides many improvements over ASP.NET, like being able to host websites with different ASP.NET Core versions on a single server.

It was not an easy topic to write about since most of the written material about conversion is about COBOL or other legacy programming language. There are only a couple of articles about converting ASP.NET to ASP.NET Core 2.0 MVC and they are vague. This research can be continued to produce more material.
REFERENCES


