

Report Title

REPORT SUBTITLE

Brian Munksgaard & Jens Christian Rasch | EAAA Backend | December 2016

Indholdsfortegnelse

[INTRODUCTION 1](#_Toc469057953)

[ASP.NET MVC 1](#_Toc469057954)

[Asp.Net MVC Components 1](#_Toc469057955)

[Conventions 3](#_Toc469057956)

[IMPLEMENTATION 4](#_Toc469057957)

[Sketching 4](#_Toc469057958)

[Uml Diagrams 4](#_Toc469057959)

[Domain Model 4](#_Toc469057960)

[Mapping between controllers and views 4](#_Toc469057961)

[Layout and Views 4](#_Toc469057962)

[Model validation 4](#_Toc469057963)

[Entity Framework 4](#_Toc469057964)

# INTRODUCTION

Arcane Tinmen are in the process of updating their online presence and as part of this work, they want to split their current website up, so that they have different websites for each of their brands. To begin this work, they have asked for an updated website for the brand Board Game Sleeves, to which they have acquired a corresponding domain name (boardgamesleeves.com).

This paper will contain two parts where the first will go through the aspects of an ASP.NET MVC website, the components and conventions that ASP.NET MVC is built upon as well as some theory behind building a website using this framework. The second part of the paper will be a detailed description of how we have used these components and conventions in our implementation of a new website for boardgamesleeves.com as well as a walkthrough of the functionality that we have implemented.

Along with this paper, we have included the code for the website (in a Visual Studio solution) as well as the database with some default data (for testing purposes).

# ASP.NET MVC

As mentioned in the introduction the website will be created by using the ASP.NET MVC framework. In the following sections, we will describe the components that comprise the framework and some conventions that the framework builds upon.

## Asp.Net MVC Components

Below is a list of the components that are part of the ASP.NET MVC framework and that we will be using in the implementation of the finished product. We will not go into details with each component type in the list, but will only briefly describe them to explain their usage and the reason for them being in the framework.

In the following section about our implementation, we will go into more detail about each component we use and why we use them.

|  |  |
| --- | --- |
| **Component** | **Description** |
| Controller | Controllers are the binding between Views and Models. The controllers handle the incoming URL requests, fetch the data through the models and send the relevant data through to the Views. The controllers can also contain logic for choosing which View to show or mappings between Models and ViewModels. |
| View | Views are templates for creating dynamic content. They define how the pages look and how the different components should be placed relative to each other. |
| Partial View | A partial view contains small parts that can be included inside a View. It is especially useful for different parts of a website that is used multiple places, like a shopping cart. |
| Helpers: HTML, Inline & Custom |  |
| Child Actions | Child actions are for Controllers what ViewModels are for Models.  A child action is a controller method that you can call from inside a View and you can set it up so you can only call this type of method from Views and not directly through a URL, as is the case for other Controller methods/actions. |
| Routes | The routing system in ASP.NET MVC specifies how incoming URLs will be handled and directed to the different controllers. |
| Models & ViewModels | Models represent the data that the website is working on, along with any business logic there might be.  The difference between Models and ViewModels is that ViewModels are centered on what is to be shown on a View where the Models are centered on the persisted data in the domain. |
| Layouts | It is possible in an ASP.NET MVC to specify layouts as part of a single page, or have pages use a common shared layout. No matter what is used, the layout specifies how the different components are positioned on the individual HTML pages that the controller send back to the browser. |
| Model Binding (IModelBinder)->State | Model binding is the data from the HTTP Request to parameters used in the action methods of a Controller. Binding can be done to both simple and complex types. It is possible to create your own model binder logic in classes that extends the IModelBinder interface. |
| Areas | Areas are a way to organize a large project or website into smaller and manageable units. An application can contain multiple areas each defining a small portion of the entire application. An example of areas could be the shopping system or the administration system of a website. |

## Conventions

The ASP.NET MVC framework also works with a couple of naming and folder conventions that makes the building of ASP.NET MVC websites easier. <insert more here!!>

By default

# IMPLEMENTATION

## Sketching

During the sketching phase we came up with the following.

Grid

## Uml Diagrams

From the site requirements, the sketches and the web stuff from Arcane Tinmen, we have built the following models and diagrams:

### Domain Model

### Mapping between controllers and views

## Layout and Views

## Model validation

## Entity Framework