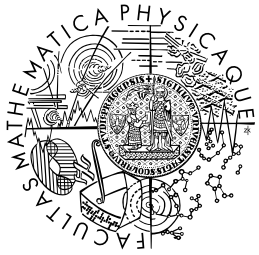

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Tomáš Husák

Client-side execution of PHP applications compiled to .NET

Department of Software Engineering

Supervisor of the bachelor thesis: RNDr. Filip Zavoral, Ph.D.

Study programme: Computer Science (B1801)

Study branch: ISDI (1801R049)

Prague 2021

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Client-side execution of PHP applications compiled to .NET

Author: Tomáš Husák

Department: Department of Software Engineering

Supervisor: RNDr. Filip Zavoral, Ph.D., Department of Software Engineering

Abstract: Blazor is a new technology enabling to run .NET applications directly in the browser using WebAssembly, a recently created binary instruction format adopted by major web browsers. Whilst PHP is the most popular language in the realm of web applications, it cannot run directly in the browser. The Peach-Pie compiler provides a way to compile projects written in PHP into Common Intermediate Language (CIL), enabling them to run on the .NET platform.

This thesis aims to design and implement a convenient interface between Blazor and compiled PHP, enabling developers to create client-side PHP applications. These applications would be able to utilize the specifics of the client-side paradigm, such as fast response times, the possibility to preserve the application state between the requests more efficiently and the direct access to the Document Object Model (DOM) of the page. To demonstrate the usability of the implementation and the specific benefits of the solution, a pilot interactive application will be created.

Keywords: PHP .NET Blazor Peachpie

Contents

Introduction

PHP is the most popular server-side language. Websites are more interested in rich graphic interface in nowdays. Javascript has been developed for interact with Browser and DOM. Since developers have to understand two programming languages, there are more space for mistakes. One of goals of this thesis is designing a comfortable way, how developers can write a whole web application using only PHP. There already exists project Pib which uses WebAssembly directly. Since it is a WebAssembly, the code should run like in native enviroment. But there is an another way how to be able to write a php script and run it in browser. This thesis aims to make integration between PeachPie and Blazor which results in a same goal as previous mentioned project. Peachpie is able to compile PHP into CIL and Blazor runs CSharp code on clien-side. As a consequence of these technologies is a possibility to use PHP in an existing Blazor project, which can be beneficial for a large amount of PHP libraries. So this approach has an extra feature that PHP and CSharp developers will be able to colaborate with their programming languages with minimum knowing of the integration. Another advantage is CSharp, PHP and JavaScript interop which offers more options for developer.

1. Existing technologies

1.1 WebAssembly

Web is a new code format which can be run in today's browsers. It has compact byte format and it's performance is near to native code. WebAssembly is designed to be a compiling target of popular low-level languages like C or C++ due to it's memory model. It should be able to support languages with carbage collector in the future. Advantage of this format is similarity with Javascript module ES2015 after compilation into machine code. This enables browsers to execute it by Javascript runtime. So it's security is same as code written in Javascript. Because of the same runtime WebAssembly can call Javascript and vice versa.

Despite of supporting to run WebAssembly in browser, the browser can't load it as a normal ES2015 module yet. WebAssembly Javasrcipt API was created in order to be able to load a WebAssembly to browser using JavaScript.

1.2 Mono

Mono is a .NET runtime which aims to mobile platforms. Recently, they started to support com into WebAssembly. This support allows executing CIL inside browsers. The compilation has two modes. One of the modes compiles only Mono runtime which then can executes .dll files without further compilation of them into WebAssembly. A consequence of this compilation is enabling to call Javascript and WebAPI from CIL.

1.3 Blazor

Blazor is a framework which provides a convinient way how to write dynamic web pages in CSharp. It offers two **?** how to build your websites.

The first one consists of two parts. Server part cares about serving web pages to client via SignalR. The thesis uses the second way which Microsoft refers as Blazor WebAssembly.

Blazor WebAssembly uses server and client part as well. The main difference is in where the webpage is put together. The client side cares about rendering the page. This is possible due to Mono and WebAssembly Javascript interop which enables to modify DOM from CSharp using Mono Webassembly. For this purpose, there are constructs which can configure the behavoir of the web application. The behavoir is meant as which pages to show or services to offer. This all settings is done by client project. The server is used for serving resources of the wep application. Recources are static assets, .dll libraries including web applivation, mono runtime and so on.

Content of the web application is composed from components. Component is a class which stands for generating a part of content. Balzor presents own virtual DOM to reduce changing DOM directly in browser in order to its demending performance.

Rendering is based on diff algorithm. When the page is rendered for the first time, the diff algorithm is not necessary and DOM is updated according to RenderBatch which is generated from RenderTreeBuilder. After page update, the diff algorithm is executed before DOM update. This algorithm gereratech RenderBatch which only modifies elements, which was updated.

The algorithm uses sequence of numbers to identify which elements was modified.

Because interleaving of HTML with other language turns out to be useful, the Razor language was introduced. This Razor differs from Razor used in .cshtml files. It is adapted to Blazor WebAssebmly enviroment which provides additional features and settings of this application. Antoher reason for using this format is to free developer from difficult using of mechanism for rendering a page content.

The process of bootstrapping Blazor app to browser folows these steps. Server gets a request for Blazor app. The server responses with html page, which contains references to Javascript responsible to load the app. The Javascript code fetches remaing resources like .dll libraries and runs Mono module. The runtime initialise the application using user defined .dll libraries.

Remaing interaction is maintained by event handling. I divides it into two type of actions.

The first type is a navigation. The nav can be triggered by an anchor, form or filling up the address bar. Address bar is handled seperated by browser. The remainings ways is handled by Javascript. Anchor is the one one which are is handled by Blazor app by default. Javascript tries to invoke navigation handler in Blazor app using a Mono WebAssembly gateway. This handler can be modified by user, but a default behavoir is implemented by a specialized component Router. The Router finds out all components, which implements an IComponent interface and tries to render the page according to path matching. The navigation can be redirected to server.

The second type is events invoked by UI like onchange. These events are registred by RenderTreeBuilder with their callbacks. When the event is triggered, Javascript call right callback.

1.4 PeachPie

TODO: What is PeachPie

TODO: Compiling PHP to .NET

1.5 PHP

TODO: Classic web page in php

2. Problem analysis

Conclusion

Bibliography

The project pib. URL https://github.com/oraoto/pib?fbclid=IwAR3KZKXWCC3tlgQf886PF3GT_Hc8pmfCMI1-43gdQEdE5wYgpv070bRwXqI.

Webassebmly. URL https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts.

compilation. URL https://www.mono-project.com/news/2017/08/09/hello-webassembly/.

URL https://chrissainty.com/an-in-depth-look-at-routing-in-blazor/.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment