# Allgemein

WebExpress ist ein leichtgewichtiger Webserver, welcher für den Einsatz in leistungsarmen Umgebungen (z.B. Rasperry PI) optimiert wurde. Durch die Bereitstellung eines leistungsfähigen PlugIn-Systems und einer umfassenden API, lassen sich Webanwendungen einfach und schnell in einer .Net-Sprache (z.B. C#) entwickeln.

# Lizenz

Die Software steht als Open-Source (MIT) zur freien Verfügung. Die Softwarequellen können über <https://github.com/ReneSchwarzer/WebExpress> bezogen werden.

WebExpress baut auf Komponenten auf, welche als Open-Source zur Verfügung stehen:

* <https://github.com/dotnet/core> (MIT)
* <https://getbootstrap.com/> (MIT)
* <https://www.chartjs.org> (MIT)
* <https://jquery.com/> (MIT)
* <https://summernote.org/> (MIT)
* <https://popper.js.org/> (MIT)

# Architektur

WebExpress ist bewusst sehr einfach gehalten. Es besteht lediglich aus Basis-Funktionalitäten zur Verarbeitung von HTTP-Anfragen, eine API und einem Plugin-System zur Erweiterung der Funktionalitäten. Damit ist WebExpress selbst nicht in der Lage Content zu erzeugen. Hierfür wird das Plugin-System benötigt. Plugins sind .Net-Assembys, welches aufbauend auf der WebExpress-API Inhalte erstellt oder enthält. Die Plugins werden von WebExpress geladen und ausgeführt. WebExpress steuert dabei die Plugins und verteilt die HTTP-Anfragen auf das zuständige Plugin. Diese beantworten die Anfragen, erstellen Inhalte und übergeben diese WebExpress. Das Ausliefern des Inhaltes als HTTP-Antwort erfolgt abschließend über WebExpress.



Im Sprachgebrauch von WebExpress stellen Plugins (Web-)Anwendungen bereit. Eine Anwendung ist der logische Zusammenschluss von Plugins (minimal besteht eine Anwendung aus einem Plugin) für das Erreichen eines gemeinsamen Zieles. Dabei werden die Anwendungen durch die Vergabe einer gemeinsamen ApplicationID gebündelt. Das Plugin-System kann sowohl zur Erweiterung der WebExpress- als auch zur Erweiterung der Anwendungs-Funktionalitäten verwendet werden.

Jedes Plugin kann Inhalte in unterschiedlichen Ausprägungen bereitstellen. Es werden die folgenden Inhaltstypen unterschieden:

|  |  |
| --- | --- |
| Inhalt | Beschreibung |
| Seiten | Webseiten, welche aus HTML-Quelltext bestehen. |
| Assets | Beliebige Dateien, wie zum Bespiel Css, JavaScript, Bilder. |
| Bibliotheken | .Net-Klassen und -Schnittstellen zur Erweiterung der Funktionalität. Die APSs werden in anderen Plugins |
| Rest-APIs | Stellt Schnittstellen bereit, welche hauptsächlich für dir Kommunikation mit (anderen) Systemen eingesetzt werden. |
| Internationalisierungs-Schemas | Stellt Sprachpakete für die Internationalisierung der Anwendungen bereit. |
| Layout-Schemas | Stellt Farb- und Layout-Schemas zur individuellen Anpassung von Anwendungen bereit. |
| Komponenten | Sind Programmteile, welche in definierten Bereichen von Seiten eingebunden werden. Die Komponenten erweitern die Funktionalität oder das Aussehen der Seite. |

## Paketmodell

Um die Anwendung einfacher zu erweitern, wird diese in mehreren Programmbibliotheken aufgespalten. Die Programmbibliotheken WebExpress.dll und WebExpress.UI.dll sind global und werden als Basis in anderen Projekten verwendet. Sie stellen grundlegende Funktionen zum Erstellen von Inhalten und Zusatzfunktionen (z.B. Logging) bereit. Die Programmbibliothek WebExpress.App.exe stellt die Anwendung dar, welche die Steuerung der einzelnen Funktionen und Komponenten übernimmt.



## Klassenmodell

Eine grobe Übersicht der Anwendung und der Plugins wird in der folgenden gegeben.



In der nachfolgenden werden die notwendigen Basisklassen der Plugins aufgezeigt. Jedes Plugin muss dabei mindestens die Factory- und Plugin-Klasse implementieren.



Die Plugins werden durch die Programmbibliothek WebExpresse.dll und von der Bibliothek WebExpress.UI um Steuerelemente und Seitenvorlagen bereitgestellt.

Nachdem die Anwendung und die Plugins definiert wurden, wird in der nächsten Abbildung 8 das Zusammenspiel der einzelnen Komponenten in einem UML-Sequenzdiagramm veranschaulicht. Aus Vereinfachungsgründen wurden Details (wie zum Beispiel das Logging) weggelassen und stattdessen auf die Kernfunktionalität eingegangen.



Abbildung : UML-Sequenzdiagramm

# Konfiguration

Die Anwendung lässt sich mit Hilfe von Konfigurationsdateien und Templates weitestgehend an die zu erwartenden Erfordernisse anpassen. Dabei werden diese Dateien im XML-Format angegeben (siehe Abbildung 9).

<?xml version="1.0" encoding="utf-8" ?>

<config>

<!-- Verbindungsinformationen -->

<credential>

<name>UserPortal</name>

<host>N3083E90\SQLEXPRESS</host>

<port>1433</port>

<user>admin</user>

<password>\*\*\*\*</password>

</credential>

<!-- Mapping -->

<include file="entw.userportal.mapping.userportal.xml"/>

<!-- Spezialattribute -->

<include file="entw.userportal.specialattributs.xml"/>

</config>

Abbildung : Beispielkonfiguration

Konfigurationsdateien können andere Konfigurationsdateien importieren. Somit können Einstellungen, welche in mehreren Plugins verwendet werden, in einer zentralen Konfigurationsdatei festgelegt werden. Die Flexibilität wird durch die Verwendung einer Skriptsprache erhöht, welche Bestandteil der TFA.Substitution-Programmbibliothek ist.

## Ersetzungszeichenfolgen

In der Konfigurationsdatei und in den Templates lassen sich spezielle Bereiche definieren, welche vor der Ausführung ausgewertet und ersetzt werden. Diese Bereiche werden als Ersetzungszeichenfolgen (MPLX) bezeichnet. Es gibt zwei Arten von Ersetzungszeichenfolgen: Variablen- und Funktionsersetzungen. Beide Arten werden im XML-Format angegeben[[1]](#footnote-1). Sie können beliebig geschachtelt werden (siehe BNF aus Abbildung 10).

|  |  |  |
| --- | --- | --- |
| MPLX | ::= | MPLX „<var>“ Variable „</var>“ MPLX |  MPLX „<global>“ Variable „</global>“ MPLX |  MPLX „<“ Funktion Param „/>“ MPLX |  MPLX „<“ Funktion Param „>“  Paramblock  Block  „</“ Funktion „>“MPLX |  MPLX Text MPLX |  ε |
| Variable | ::= | Name Index |
| Funktion | ::= | Name |
| Paramblock | ::= | „<param name=“ Name „ value=“ Wert „/>“ Paramblock |  „<param name=“ Name „>“  MPLX  „</param>“ Paramblock |  ε |
| Block |  | „<“ Name „>“ MPLX „<“ Name „/>“ Block |  „<“ Name Param „>“ MPLX „<“ Name „/>“ Block |  ε |
| Name | ::= | Zeichen Name | Zeichen |
| Param | ::= | Name „= ‘“ Wert „‘“ | Name „=‘“ Wert „‘ “ Param | ε |
| Index | ::= | „[“ Zahl „]“ | ε |
| Zeichen | ::= | Zahl | „A“ | … | „Z“ | „a“ | … | „z“ | „\_“ | „\*“ |
| Zahl | ::= | „0“ | ... | „9“ |
| Wert | ::= | alle Zeichen außer „<“ | „>“ | „‘“ |
| Text | ::= | alle Zeichen |

Abbildung : BNF



1. z.B. <date/> [↑](#footnote-ref-1)