20.06.2022

CodeSnipper

M151

- Arnold Tim
- Eugster Andrin

Inhalt

Ziel von CodeSnipper

CodeSnipper ist eine Plattform, welche das einfache Speichern und Veröffentlichen von CodeSnippets ermöglicht.

Als Entwickler gibt es vielmals Code, welchen man wiederholt schreiben muss. Um diese lästige Arbeit zu vernichten, können solche Ausschnitte in CodeSnipper notiert werden.

Die Software bietet die Möglichkeit, Snippets als privat oder öffentlich abzuspeichern, womit andere ebenfalls profitieren können. Zusätzlich kann man einem Snippet einen Titel und Kategorien zuweisen, wobei das Snippet schnell gefunden werden kann.
CodeSnipper verzichtet bewusst auf eine Beschreibung, da die Ansicht so simpel wie möglich gehalten wird, und das Snippet so schnell wie möglich kopiert werden kann.

Beispielsweise können ConnectionStrings, die Methode "RaisePropertyChanged" und weitere nützliche Funktion mit CodeSnipper kopiert werden.

Architektur

In diesem Kapitel wird erläutert, wie genau die Web-Applikation gebaut wurde. Es werden Dinge über die Architektur beschrieben, Themen, welche mit Datenbanken zu tun haben, aufgegriffen, sowie manche Dinge zu Sicherheitsaspekten, wobei das meisten im Kapitel Sicherheit dokumentiert ist.

Allgemeines

Um unsere Vision zu realisieren, benutzten wir gewisse Frameworks, welche hier kurz erwähnt werden:

- C# als Haupt-Programmiersprache
- .NET 6.0
- ASP.NET MVC
- Entity Framework
- Standard-Login von ASP.NET MVC

4-Tier Architektur

Unsere Web-Applikation hat eine 4-Tier Architektur implementiert. Dies bedeutet, dass es 4 Teilbereiche gibt, welche zusammen harmonieren. Unsere verschiedenen Layers sind:

- Presentation Layer
- Business Layer
- Data Access Layer
- Data Layer

Diese werden in Unterkapiteln erläutert.

Presentation Layer

Im Presentation Layer geht es um die Darstellung von Inhalt. Der Layer interagiert mit dem Business Layer, von dem es seine Daten erhält.

Das MVC Design Pattern wird innerhalb des Layers angewendet. Die Daten, welche von dem Business Layer geholt werden, werden in Models (ViewModels) abgefüllt, mit denen dynamisch HTML erzeugt werden. Das erzeugte HTML wird mit weiteren, statischen Ressourcen dem Client zugesendet.
Das Layer ist somit zuständig für die Präsentation des Inhaltes.

Business Layer

Im Business Layer wird Funktionalität definiert, welche genutzt werden kann. Der Layer interagiert mit dem Presentation und dem Data Access Layer.

CodeSnipper stellt hierbei verarbeitete Daten für die Präsentationsschicht zur Verfügung.

Data Access Layer

Der Data Access Layer vereinfacht den Zugriff auf den Data Layer. Der Layer interagiert mit dem Data Layer und dem Business Layer.

Hierbei spielt in unserem Projekt Entity Framework eine grosse Rolle. Es regelt den Zugriff auf die Datenbank.
Zusätzlich gibt es ein C#-Projekt, indem die Models liegen und die Verbindung mit der Datenbank definiert ist.

Data Layer

Im Data Layer geht es hauptsächlich um abgespeicherte Daten. Genauer gesagt ist dies die Datenbank. Die Datenbank, welche CodeSnipper benutzt ist eine MSSQL-Datenbank.

CodeSnipper speichert die Snippets und die Benutzer persistent in die Datenbank.

Datenbank

In der Datenbank sind die Daten abgespeichert, welche CodeSnipper benötigt. Sie ist eine MSSQL-Datenbank.
CodeSnipper speichert die eingetragenen CodeSnippets und die erstellten Benutzer in die Datenbank ein.

Die Daten werden mithilfe des Entity Frameworks ausgelesen und verarbeitet.

Sessions

CodeSnipper verwendet Sessions zur Anmeldung von Benutzern. Damit diese Session nicht sofort verfällt, werden Cookies benutzt.
Für das Anmelden von Benutzern wird das Standard-Login von ASP.NET verwendet. Da dies bereits sicher implementiert wurde und es in unserem Beispiel keinen Sinn ergibt, etwas Eigenes zu machen.

SSL/TLS

Das Verwenden von SSL / TLS ermöglicht uns, eine verschlüsselte und somit gesicherte https Verbindung zu gewährleisten.

Natürlich muss dies auch beim Server aktiviert werden, unsere Applikation ist aber schon mal dafür vorbereitet.

Dieses Feature haben wir nicht selbst eingebunden, da dies bereits von ASP.NET bereitgestellt wird.

Arbeitsjournal

Das Arbeitsjournal wurde in Workjournal.md geführt. Zu dieser Datei steht auch eine PDF-Version zur Verfügung.

Sicherheit

SQL Injection

Bei einem SQL Injection Angriff versucht der Angreifer die eigentliche SQL-Abfrage zu manipulieren, sodass sensible Daten ausgegeben werden.
Die Manipulation der SQL-Abfrage erfolgt über Eingabefelder wie zum Beispiel ein Suchfeld oder ein Registrier- & Anmeldeformular.

In unserem Projekt wirken wir solchen Angriffen entgegen, indem wir das Entity Framework mit Linq verwenden. Somit wird aus dem Userinput keine SQL-Abfrage generiert, sondern die Ergebnisse von Entity Framework gefiltert.

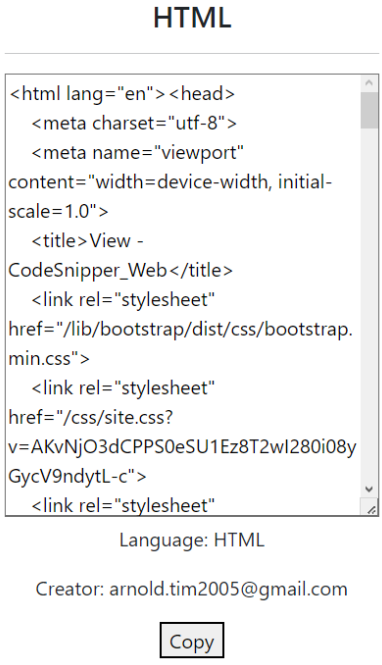
Die Benutzereingabe hat somit nichts mehr mit SQL zu tun und es können keine ungewollten Datenabfragen getätigt werden.

Cross Site Scripting

Cross Site Scripting kann benutzt werden, um eigenen JavaScript-Code auf einer Webseite auszuführen.

Bei beispielsweise einer Blog-Seite können neue Beiträge erstellt werden. Ist die Web-Applikation schlecht programmiert, wird beim Abbilden des Inhalts den angegebenen Text als HTML abgebildet, womit JavaScript-Code ausgeführt werden kann. Beispielsweise können "<script>" Tags oder iframes benutzt werden.

CodeSnipper behandelt alle Eingaben so, dass sie wortwörtlich abgebildet werden:



Directory Traversal

Hierbei werden auf Dateien, welche nicht im Web-Verzeichnis enthalten sind, zugegriffen. Dies können wichtige Dateien wie beispielsweise Sourcecode sein.

Systemdateien können bei CodeSnipper nicht eingesehen werden, da nirgends die Möglichkeit besteht, eine Datei mit einem Namen angegebenen herunterzuladen.

DoS/DDoS

DoS oder DDoS Attacken bedeutet, dass sehr viele Anfragen auf einen Webserver gesendet werden, sodass der Server überlastet wird und keine Ressourcen mehr für eine Antwort von Anfragen zur Verfügung stehen.

Die meisten Einstellungen gegen solche Attacken werden im Webserver selbst definiert. Es können auch dienste von z.B. Cloudflare benutzt werden.

Man in the Middle

Bei einem Man in the Middle Angriff wird eine vermeintlich sichere und private Kommunikation zwischen zwei Parteien über einen Angreifer geführt, der die Anfragen und Antworten der Opfer zwar weiterleitet und sie damit denken lässt sie wären sicher, jedoch die komplette Kontrolle über den Austausch hat.
Somit kann der Angreifer an jegliche sensiblen Daten gelangen.

Diese Angriffe können durch Endpunkt Authentifizierung verhindert werden, die sicherstellen, dass auch nur der richtige Empfänger die Nachricht entschlüsseln kann.

In unserem Projekt wird dies durch den Einsatz von TLS / SSL gewährleistet.

Man in the Browser

Ein Man in the Browser Angriff beschreibt einen Trojaner im Browser des Opfers, der selbstständig die Darstellung von Webseiten verändern und Transaktionen durchführen kann.

Für das Opfer ist es nahezu unmöglich diesen Trojaner zu identifizieren.

Eine mögliche Massnahme gegen diese Attacken ist ein Antiviren Programm. Als Entwickler kann man jedoch kein keine weiteren Massnahmen vornehmen.

Session Hijacking

Bei Session Hijacking werden die Cookies von einem angemeldeten Client ausgelesen. Damit wird eine Session hergestellt, ohne Anmeldedaten zu benutzen. Der Angreifer kann somit eine Session aufbauen.

Um dies bestmöglich zu verhindern, werden alle Cookies auf HttpOnly gesetzt, wobei Cookies nicht von JavaScript eingelesen werden können.

Social Engineering

Als Entwickler können wir grundsätzlich nichts gegen Social Engineering vornehmen, da solche Attacken auf menschlichem Versagen basieren und keine Softwarelösung möglich ist. Wir können jedoch den Benutzer sensibilisieren und vor solchen Angriffen warnen.

In unserem Projekt versuchen wir dieser Aufgabe in Form von einem kleinen Text über dem Footer gerecht zu werden.