Engineer

Generator für statische Websites

Programm, das alle Seiten im Voraus als Dateien schreibt

Was es ist

Ein Generator für statische Websites ist ein Build-Werkzeug, das einen Ordner mit Quelldateien, meist in Markdown geschriebenen Seiten und einem Satz Vorlagen, in eine fertige Website umwandelt, die nur aus Dateien besteht: HTML, Stylesheets, Skripte und Bilder, die ein Webserver unverändert ausliefert. Der Begriff wird oft mit SSG abgekürzt. „Statisch“ beschreibt dabei die Auslieferung, bei der alle Anfragen an eine Adresse dieselben Bytes erhalten, und nicht das Verhalten einer Seite, nachdem sie im Browser angekommen ist.

Dem steht der dynamische Ansatz eines Content-Management-Systems oder einer Webanwendung gegenüber. Dort setzt ein Programm auf dem Server jede Seite bei der Anfrage zusammen, meist durch eine Datenbankabfrage und das Ausfüllen von Vorlagen. Ein Generator verarbeitet dieselbe Art von Inhalten mit derselben Art von Vorlagen, aber einmal pro Änderung statt einmal pro Besuch. Das Ergebnis liegt in einem Ordner, den ein gewöhnlicher Webserver, ein Speicher-Bucket oder ein Content Delivery Network ausliefern kann, ohne zu wissen, wie die Seiten entstanden sind.

Hugo ist ein solcher Generator. Seine Dokumentation stellt ihn als „einen in Go geschriebenen Generator für statische Websites, auf Geschwindigkeit optimiert und auf Flexibilität ausgelegt“ vor, der „eine vollständige Website in Sekunden rendert, oft noch schneller“. Üblicherweise bearbeitet man die Inhalte, während der eingebaute Webserver läuft. Dieser baut die Website bei jeder Änderung neu und lädt die geöffneten Browserseiten erneut. Anschließend baut man die Website und stellt sie auf einem Host bereit:

```
hugo server     # build, serve, rebuild and reload on every save
hugo --minify   # write the finished site into public/
```

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Generator für statische Websites?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

Full-Stack-GIS-Kartenentwicklung geleitet Kontakt aufnehmen →

Warum es wichtig ist

Die Entscheidung betrifft den Zeitpunkt der Arbeit. Ein Generator verlagert sie von der Anfrage auf den Build. Fast alles Weitere ergibt sich daraus.

Eine Datei zu senden gehört zu den günstigsten Aufgaben eines Webservers. Deshalb antwortet eine generierte Website schnell und kann auf einem sehr kleinen Rechner laufen. Da jeder Besucher einer Adresse dieselben Bytes erhält, ist Caching einfach. Ohne laufenden Anwendungscode oder eine Datenbank hinter den Seiten gibt es wesentlich weniger abzusichern und zu aktualisieren. Das Ergebnis ist ein Ordner; ein Wechsel auf einen anderen Host bedeutet daher, Dateien zu kopieren. Der Quelltext liegt in der Versionsverwaltung, in der sich jede Änderung an jeder Seite prüfen und nachvollziehen lässt.

Die Nachteile haben dieselbe Ursache. Jede Änderung, auch ein korrigierter Tippfehler, erfordert einen neuen Build und eine Bereitstellung. Mit wachsender Seitenzahl wird die Build-Geschwindigkeit deshalb wichtiger. Alles, was von der anfragenden Person abhängt, etwa Suche, Kommentare, Anmeldung oder ein Formular, das seine Eingaben speichert, braucht Browsercode, einen separaten Dienst oder eine Lösung zur Build-Zeit: Auf dem Server läuft kein Programm, das diese Arbeit übernehmen könnte. Zwischen Builds steht die Zeit still. Hugo lässt standardmäßig Seiten mit einem Datum in der Zukunft aus. Eine für Montag geplante Seite erscheint daher mit dem ersten Build am Montag oder danach. Selbst Weiterleitungen nehmen eine andere Form an. Ohne ein Programm, das eine HTTP-Weiterleitung sendet, legt Hugo bei einer verschobenen Seite standardmäßig eine kleine HTML-Datei an der alten Adresse ab, die den Browser weiterschickt. Die Dokumentation bezeichnet diesen Ansatz als auf alle Hostinganbieter übertragbar.

Wie es in der Praxis aussieht

Diese Website wird mit einem solchen Generator gebaut. Die Credits-Seite nennt Hugo als Generator für statische Websites und schreibt ihm die Vorlagen, Taxonomien, sämtliche Sprachen und Feeds zu. Zum Ergebnis im Browser erklärt sie, dass der HTML-Code vollständig geschrieben wird, bevor ihn jemand anfragt: Jede Seite ist eine Datei auf dem Datenträger statt einer bei jeder Anfrage zusammengesetzten Vorlage. Sogar das Aussehen wird beim Build festgelegt, denn das Hintergrundbild des hellen Designs wird während des Seitenbaus ausgelost und die Website bei jeder Veröffentlichung neu eingekleidet. Die Qualitätsprüfungen laufen lokal auf dem Rechner, auf dem die Website bearbeitet wird, ohne gehosteten Build-Dienst in der gesamten Kette. Jede Diagnose gilt als Fehler; im Oktober 2026 erzeugte die Website 2.268 Seiten. Ausgeliefert werden sie von einem Server mit einem Kern und 512 MB Arbeitsspeicher. Dort lesen der Webserver und die drei anderen für die Website zuständigen Dienste denselben Dateiordner, sodass eine Bereitstellung alle vier aktualisiert.

Der Portfolioeintrag zum Aufbau dieser Website mit Hugo dokumentiert den Implementierungsstand vom August 2026 und erläutert die Gründe. Drei Sprachen, kein gesondert berechneter Anwendungsdienst und keine an Dritte weitergereichten Besucher schlossen nahezu jedes Content-Management-System aus: Ein laufendes Backend muss abgesichert, aktualisiert, bezahlt und auf einer Datenschutzseite erklärt werden. Die Website hatte damals 108 Vorlagendateien, davon 53 Partials, veröffentlichte jede Seite in vier Darstellungen aus einem Inhaltsbaum und baute diesen Baum sieben weitere Male für fokussierte Subdomains, indem der Generator auf einen anderen Inhaltsordner verwiesen wurde. Der Eintrag beschreibt den Preis dafür, Funktionen zur Build-Zeit statt im Browser hinzuzufügen: Abgesehen von der dokumentierten, begrenzten JavaScript-Ausnahme für einen Portfoliofilter mussten interaktive Funktionen beim Build gelöst werden oder entfallen. Deshalb wurde ein Suchindex durchgerechnet und zurückgestellt, statt ihn auszuliefern.

Wo es schiefgeht

Eine gelöschte Seite bleibt online. Eine aus den Quellen entfernte Seite wird weiter unter ihrer alten Adresse ausgeliefert, und Suchmaschinen finden sie weiterhin. Hugo leert seinen Ausgabeordner vor einem Build nicht: Vorhandene Dateien werden überschrieben, aber nie gelöscht. Deshalb kann `public/` Seiten früherer Builds behalten, darunter Entwürfe und abgelaufene Seiten. Alles, was

diesen Ordner auf einen Server kopiert, kopiert auch sie. Der `resources/` -Cache hat ein verwandtes Problem: Die Designregeln dieser Website warnen davor, dass er ein veraltetes CSS-Bündel liefern kann, wenn sich nur das Quell-CSS geändert hat. Bauen Sie in einen sauberen Ordner, mit Hugos Option `--cleanDestinationDir` oder einem eigenen Löschschritt. Der Build dieser Website entfernt sowohl `public/` als auch `resources/` , bevor er beginnt.

Eine fertige Seite erscheint nie. Ein Beitrag wird geschrieben, committet und bereitgestellt, der Build meldet Erfolg, und auf der Website fehlt die Seite. Hugo lässt standardmäßig als Entwurf markierte Inhalte aus, ebenso Inhalte mit einem zukünftigen Datum oder Veröffentlichungsdatum sowie Inhalte mit einem überschrittenen Ablaufdatum. Nichts davon gilt als Fehler. Ein versehentlich um einen Tag zu weit in die Zukunft gesetztes Datum genügt. Prüfen Sie zuerst diese vier Felder, beschränken Sie `--buildDrafts` , `--buildFuture` und `--buildExpired` auf Vorschauen und sorgen Sie bei einer geplanten Seite für einen Build nach ihrem Datum.

Alle Seiten behaupten, gleichzeitig geändert worden zu sein. Die Sitemap weist fast jeder Adresse denselben Zeitpunkt der letzten Änderung zu. Das sagt einem Crawler, die ganze Website habe sich geändert, und verrät nichts darüber, welche Seiten tatsächlich betroffen sind. Ein Generator muss das Änderungsdatum jeder Seite irgendwoher beziehen. Der Zeitstempel der Datei ist die falsche Quelle, wenn ein Schritt Dateien mit unverändertem Inhalt neu schreibt. Die Konfiguration dieser Website hält fest, dass die Synchronisierung bei jedem Lauf alle 189 Inhaltsdateien neu schreibt. So teilten 172 der 176 Sitemap-Adressen einen Zeitstempel. Beziehen Sie das Datum stattdessen aus der Versionsverwaltung: Mit aktiviertem `enableGitInfo` liest Hugo das Datum der letzten Git-Revision einer Datei. Eine erneute Synchronisierung mit identischen Bytes erzeugt keinen Commit, sodass eine Synchronisierung nach einem Monat Bearbeitungen nur 8 von 186 Dateien berührte.

Dateien werden veröffentlicht, die niemand veröffentlichen wollte. Wer den Namen errät, kann eine Datei abrufen, die andere Dateien auflistet. Ein Generator veröffentlicht alles aus den gelesenen Ordnern, abzüglich dessen, was eine Ausschlussregel entfernt. Erreicht eine Regel nicht jeden Ordner, bleibt der Rest offen. Die Konfiguration dieser Website hält fest, dass ihre allgemeine Ignore-Liste einen Modul-Mount nicht erfasste: Der Build veröffentlichte auf jedem Host alle sechs `.DS_Store` -Dateien, die der macOS Finder unter icons/, images/ und webmanifest/ hinterlassen hatte. Jede listete die Namen der Einträge im selben Ordner auf; die Datei unter images/ nannte deshalb einen Ordner, den die Website zurückhält. Ein Filter, dem nach seinem `!` ein

Leerzeichen fehlte, schloss einmal stattdessen jedes Bild eines Mounts aus. Schließen Sie unerwünschte Dateien in jedem veröffentlichenden Mount aus und prüfen Sie danach die gebaute Ausgabe auf Dateien, die dort nicht hingehören. Zwei Prüfungen dieser Website lassen einen Build scheitern, der eine davon ausliefert.

Weiterführende Lektüre

- Hugo: Introduction — was Hugo ist, wozu es dient und wie der Ablauf vom Schreiben über den Build bis zur Bereitstellung funktioniert.
- Hugo: Basic usage — eine Website bauen und ausliefern, weshalb `public/` alte Dateien behält und welche Inhalte ein Build auslässt.
- Hugo: Configure front matter — wie Hugo die Datumsangaben einer Seite auswählt, einschließlich der Token `:git` und `:fileModTime`.
- Hugo: Configure modules — Mounts, ihr jeweiliger `files` -Filter und die Regel, dass ein definierter Mount den Standard-Mount entfernt.
- Hugo: URL management — Aliasse und die Weiterleitungsdateien, die ein statischer Host anstelle einer Serverweiterleitung ausliefert.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Generator für statische Websites?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/ssg/