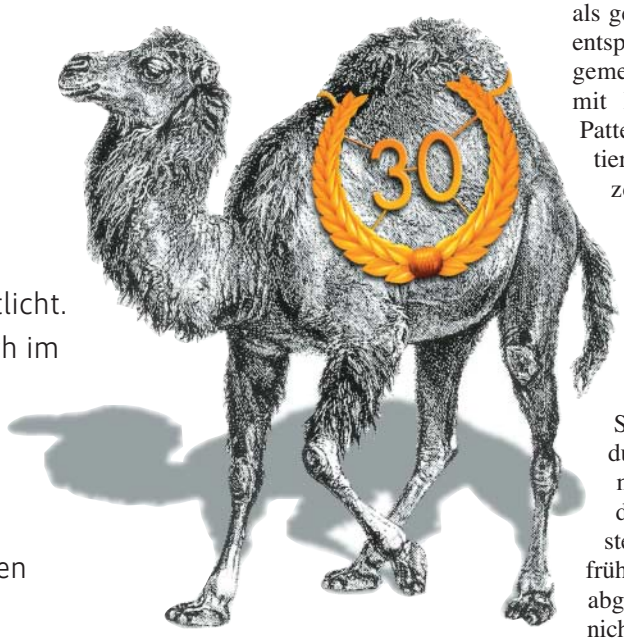
Eine Skriptsprache feiert 30. Geburtstag

Happy Birthday, Perl!

Susanne Schmidt

Vor 30 Jahren wurde Perl als eine der ersten „Skriptsprachen“ veröffentlicht. Einst wohlgelitten, vielfach im Einsatz und ein Antreiber des neuen, hippen WWW, fristet Perl heute ein Nischendasein und muss um seinen Ruf kämpfen.



Vor 30 Jahren, als die Welt noch in Ordnung war und „das Web“ nicht existierte, echte Männer C und Assembler programmierten und sich die Hände mit Shell-Skripten schmutzig machten, erblickte Perl als ziemlich schlaue Mischform aus Unix-Zugriff, besserer Shell und Programmiersprache das Licht der Welt. Ruhige Jahre folgten. Perl wurde angenommen, neu released, verbessert und erweitert und erreichte Version 4.

Dann brach das WWW über die Technikwelt herein. Zunächst verlacht als lächerlich – „Naja, so ’ne Skriptsprache, was kann die schon“ und „hat nicht mal Typen“ –, zeigte sich Perl schnell als eine der handlichsten Sprachen, um CGI-Skripte zusammenzuhacken und die ersten „Webapplikationen“ zu launchen. Viele Firmen der ersten Stunde von Amazon bis zum Sonst-was-Webshop setzten Perl ein, dementsprechend gefragt war die Sprache.

Als Version 5 erschien, die viele als die erste Version zählen, die Perl zu einer „vollwertigen“ Programmiersprache machte, schien der Erfolg unbestreitbar. Perl genoss einen hohen Grad an Einfluss – praktisch jede Programmiersprache hat heute zum Beispiel Perls Geschmacksrichtung von regulären Ausdrücken implementiert, die seinerzeit die beklagenswerte Backslasheritis einfach rausgeworfen hatte und damit das Leben aller vereinfachte.

Perl ist ausgesprochen „funktional“ orientiert – was heute in der JavaScript-Welt schwer gehypt wird, konnten Perl-Entwickler schon vor 20 Jahren anwenden. „Let over Lambda“ adelte Perl gar als gelungene „blub language“ in funktionaler Hinsicht. Der Aufstieg schien fast nicht aufzuhalten – erste Versionen von PHP waren in Perl implementiert; Perl-Bindings für alle möglichen APIs und Bibliotheken und Tools schossen aus dem Boden, Perl-Bücher wurden in Serie publiziert.

Opfer von PHP und Java

Kurz vor der Jahrtausendwende, in einer Technikkultur zwischen Dotcoms, Startups und alteingesessenen Technikfirmen, die kaum wussten, wie ihnen geschah, setzte dann die XMLifizierung und der Siegeszug von Java ein. PHP hatte sich so weit etabliert, dass es eine grauenvolle, aber sehr praktische Implementierung eines Homepage-Baukastens zu sein schien. Zwar voller Sicherheitslücken, aber so praktisch und so viel einfacher zu installieren als Perl, dass viele Websites von Perl zu PHP wechselten.

Auch neuere, alternative (immer noch belächelte) Skriptsprachen wie Ruby und Python existierten bereits. Mit Suns J2EE-Release allerdings setzte eine andere Kultur ein: Webapplikationen wurden als geschäftskritisch begriffen und dementsprechend fand sich die Programmiergemeinde plötzlich in der Pflicht, sich mit Enterprise-Architekturen, Design-Patterns und traditionellen objektorientierten Designfragen auseinanderzusetzen. Perls handgemachtes OO mit „bless“ erschien nicht mehr professionell und zeitgemäß.

Perl hat exzellente Features und Fähigkeiten. Schon seit Version 5 ist das Scoping aufgeräumt. Grundsätzlich kann Perl alles, was man von einer zeitgenössischen Sprache derzeit erwartet; einen Modul-Installer gibt es seit anno dunnemals und das Konzept von Modulen, die online gesammelt und bereitgestellt werden, hatte sich Perl schon frühzeitig mit CPAN von TeX (CTAN) abgeguckt. Andere Sprachen hatten das nicht oder erst sehr spät.

CPAN war seiner Zeit voraus

CPAN erledigt übrigens auch automatisierte Tests auf diversen Plattformen, die Modulentwickler als Reports griffbereit vorfinden. Es gibt ein (wenn auch altmodisches) Unified Interface für Datenbanktreiber (DBI), ORMs (DBIx::Class), Webframeworks (Dancer, Mojolicious, Catalyst) und nahezu jedes Modul, das man sich vorstellen kann – inklusive der wichtigen „Langweiliges Zeug“-Module wie Excel, die man dringend haben will. Perl hat inzwischen OOP (Moose) und kann alternativ genau wie JavaScript in einer sehr funktionalen Art und Weise benutzt werden. Closures und Callbacks, Asynch-IO – alles da. Nicht zu reden von exzellenten Handbüchern, sehr guter Dokumentation und einer Community, die als etwas verrückt, aber kompetent gilt.

Diversity der Community? Kein Thema – auch wenn Perl wie viele IT-Communitys an mangelnder Attraktivität bei Frauen leidet, ist die Perl-Gemeinde in der ihr eigenen exzentrischen Art selbst immer offen für schwul, lesbisch, trans, queer, non-binary.

Und doch – Perls Ruf ist ruiniert. Perl gilt als „Write once, never read again“-Sprache, gefüttert durch jahrelange „Ob-

fuscated ist cool“-Kultur und durch die Propagierung der schlimmsten Seiten von Perl. Schlechter Stil ist weiterhin gängig – als ob jede Vernunft und jede Best Practice mit Perl aus dem Fenster gekippt wird. Das hat die Sprache eigentlich nicht verdient – aber eine Dekade von Versäumnissen der Community hat nichts zur Rufverbesserung beigetragen.

Und die Kommunikationskultur? Auch Perl hat es wie viele Technik-Communitys versäumt, seine angriffslustigen Alphatierchen und die alte Nerd-Kultur der „tough love“, die heute so vollständig deplatziert wirkt, aufzuräumen. War ich noch vor einigen Jahre der Überzeugung, ein Code of Conduct sei hier nicht notwendig, belehrten mich die hässlich ausgetragenen inneren Streitigkeiten und eine insgesamt aus dem Ruder gelaufene Aggressionskultur eines Schlechteren.

Die lange Geburt von Perl 6

Im Jahr 2000 kam – zunächst als Aprilscherz – „Perl 6“ als spannende, neue Idee auf. Ein schöneres Perl, ein aufgeräumtes Perl, mit einigen Ecken und Kanten, geglättet und um einige Absurditäten verschönert. Das war die Idee. Über die letzten 17 Jahre hat sich dann Perl 6 – dessen Release im Dezember 2015 inzwischen niemand mehr interessierte – zu einer monströsen Programmiersprache entwickelt, die alles kann. Wirklich alles. Ganz viele Elemente sind gut und richtig (Unicode done right) und die Features rund ums Handling von Listen beispielsweise sind schnuckelig. Trotzdem ist Perl 6 eine riesige Sprache, ein kaum zu durchschauendes Feature-Monster.

Alles, was viele Leute an Perl immer gehasst haben – der Syntax-Clutter, die Sigils et cetera –, wurde nicht bloß aufgegriffen, sondern etwa mit „Trigils“ noch auf die Spitze getrieben. Inzwischen haben alle Programmiersprachen ihre Tüttelchen und Kringelchen und Pfeile, die für alle möglichen syntaktischen und deklarativen Zwecke eingesetzt werden – aber Perl 6 nutzt das so exzessiv, dass sich die Großrechnersprache APL aus den 1960ern als Vergleich aufdrängt.

Und Perl 6 kam zu spät. Viel zu spät. Lange nach dem Siegeszug des „sauberen“ Python, dem Aufstieg von Ruby nebst beispielgebenden Rails, nach der Vielfalt neuer Sprachen von Julia über Rust, Go bis zu Elixir und JavaScripts grandiosem Erfolg kam dann endlich Perl 6 als echte Release. Naja. Who cares. Braucht heute keiner mehr. Und bisher verfehlt Perl, sich neu zu positionieren, sich unentbehrlich zu machen in einer Nische, obwohl die Fähigkeiten der Sprache dem in keiner Weise entgegenstehen.

Perl hat es verpasst, sich mit dem Beginn des neuen Jahrtausends sinnvoll zu platzieren. Man hat sich endlos aufgehalten, bis Moose geboren war, und dann wars nicht schön genug und dann gabs Mouse (schneller), dann Moo (ohne Meta), dann Mo. Demnächst gibts dann ganz ehrlich wirklich das MOP von Stevan Little als Teil der Sprache in Perl 5. Inzwischen hat sich auch die Perl-5-Community trotz Überschneidungen an vielen Stellen von der Perl-6-Gemeinde gelöst. Vorschläge zum Umbenennen einer der beiden Schwestersprachen führen nur noch zu genervtem Augenrollen.

PHP hat die Nische des einfachen Homepage-Bauens bei Webhostern besetzt, weil die PHP-Community zügig massiv geliefert hat: Webshops, Foren, Blog-Software, CMS. Egal, wie schrecklich der Code im Inneren sein mochte, die PHP-Welt hat Bedürfnisse erkannt und erfüllt.

Ruby hat (hoffentlich) die gesamte Wahrnehmung von Codeästhetik gewandelt – dreckiges Perl ist einfach nicht mehr akzeptabel. Die Verhöhnung als Write-once-Sprache ist natürlich Unsinn. Man kann problemlos sauberes, lesbares Perl schreiben. Ruby hat mit Rails das Webframework geliefert, dessen Einfluss gar nicht überschätzt werden kann. Python hat sich direkt in Schulen, Universitäten und Einsteigerseminaren als die „einfachere, saubere“ Sprache platziert und ist inzwischen unangefochtener Sieger in vielen Bereichen, die Perl hätte besetzen können: Big Data und DevOps beispielsweise.

Perl ist einfach uncool

Wer sich heute die gerade angesagte Tool-Landschaft anschaut, wird dort kein Perl mehr finden. Go und Python, ein bisschen Ruby hier und dort und eine Reihe von Nicht-Perl-Sprachen bestimmen die gesamte Automatisierungs- und DevOps-Landschaft. Perl hätte jede einzelne dieser Nischen aufnehmen und füllen, die „Killer-App“ liefern können. Stattdessen haben sich Perl 5 und 6 eine Dekade lang in einer Art von Nabelschau aufgerieben – und komplett verloren. Der Siegeszug von PHP und JavaScript zeigt, dass die „Sauberkeit“ einer Sprache im Endeffekt irrelevant für die Popularität ist. „Worse is better“ (Richard P. Gabriel) zeigt sich auch hier sehr deutlich.

Quelle: Klapi, CCL

Perl-Vater Larry Wall auf der Fosdem im Februar 2015. Perl 6 kam dann doch erst zum Jahresende.

Es gibt keine inspirierenden Perl-Bücher mehr, kaum Artikel, die „cool stuff“ in Perl zeigen. Programmier-Neulinge wählen heute JavaScript oder Python. Initiativen wie „Rails Girls“, denen massenhaft die Bude eingerannt wird, gibt es für Perl nicht. API-Bindings existieren heute für Python, nicht für Perl. Interessante Jobs mit Perl? Die deutsche Job-Landschaft zeigt in der Regel entweder eine Dekade alte Webapplikationen zum Aufräumen oder enthält nur noch die bloße Formulierung „Perl-Kenntnisse hilfreich“ – aber eigentlich ist alles in Python.

Perl ist heute eine Nischensprache, verfolgt von Liebhabern in ihrer Freizeit. Auch das Aufzeigen der Handvoll Firmen, die heute noch Perl verwenden oder neu verwendet haben, der Qualität und Stabilität vieler Tools hilft offenbar nicht mehr, neue Leute zu locken. Ich halte Mojolicious für eines der besten Webframeworks, die ich kenne. Und doch wird es Perl als Ganzes nicht mehr zum Erfolg verhelfen. Neue Moden – egal ob gerechtfertigt oder nicht – werden von Perl nicht mehr aufgegriffen. Es wird wohl nie das ultimative Super-Framework für Machine Learning in Perl geben und der Zug im DevOps-Bereich ist damit dann auch abgefahren. (Ja, es gibt Rex.)

Traurig eigentlich – aber vielleicht reißen die nächsten 10 Jahre das Ruder noch herum. Ich gebe die Hoffnung für Perl noch nicht auf. (js)

Susanne Schmidt

ist Politologin und arbeitet bei SysEleven GmbH als Senior-Developerin.

Alle Links: www.ix.de/ix1712108