Perl, PHP, Python
Ein Vergleich

Vor- und Nachteile von dynamischen Sprachen

Über Alvar Freude

- Freiberuflicher Softwareentwickler, Trainer und Berater
- Schwerpunkt:
  - Die Skriptsprache Perl, SQL (PostgreSQL)
  - Datenbankanwendungen und Web-Applikationen
  - Performance-Tuning; Best Practices
- Seit 2005 Trainer für IBM im Bereich Perl
- http://alvar.a-blast.org/ | alvar@a-blast.org

Einführung

Übersicht über die Eigenschaften und Besonderheiten von Skriptsprachen

Was sind Skript-Sprachen?

* Anfangs für kleine Hilfsprogramme gedacht
  * Ziel: Einfache Probleme einfach lösbar
* Heute: mächtige und vollwertige Programmiersprachen, für nahezu alle Aufgaben nutzbar
  * „Die einfachen Aufgaben einfach halten, ohne die schwierigen unmöglich zu machen“
* Die unterschiedlichen Skriptsprachen haben verschiedene Stärke und Schwächen

Typische Einsatzgebiete

* Traditionell: Systemadministration aller Art
* Web-Applikationen
* Datenbank-Anwendungen
* Bearbeitung von Texten (beispielsweise Logfiles)
* Testing (TAP, Test Anything Protocol)
* Client-Server-Anwendungen
* ...

Charakteristika

- Skript-Sprachen werden i.d.R von einem *Interpreter* ausgeführt
  - Zur Beschleunigung meist als Bytecode
- Skript-Sprachen sind in der Regel dynamisch typisiert
  - Variablen haben keinen festen Datentyp
- Dynamische Anpassungen während der Laufzeit möglich
- Automatische Speicherverwaltung

Vorteile dynamischer Typisierung

* Flexibler und einfacher zu handhaben
  * z.B. beim Aufbau von komplexen Datenstrukturen
* Eingaben (z.B. aus Datei oder Datenbank) können ohne weitere Konvertierung auch als Zahlen behandelt werden
* Erweiterbarkeit, ohne vorhandenen Code bei Typerweiterungen anfassen zu müssen
* Allgemein: die Sprache wählt zur Laufzeit „das Richtige“ von selbst

Nachteile dynamischer Typisierung

- Je nach Implementierung: niedrigere Performance, da Typ-Tests zur Laufzeit geschehen
  - Overhead bei immer gleichen Berechnungen
- Manche Fehler werden erst zur Laufzeit entdeckt
  - Daher: u.U. erschwerte Fehlersuche
- Erlaubt laxen Umgang des Entwicklers
- Auch eine Frage des persönlichen Geschmacks

Vorteile von Skriptsprachen

* Man kann sofort loslegen, Quelltext ändern, wieder starten – alles ohne große Formalitäten
* Je nach Sprache aber auch Formalitäten aktivierbar, wie *use strict;* und *use warnings;* in Perl
* Kurze Entwicklungszyklen und kompakter Code
* Mächtige Konstrukte, reiche Sprachen, hohe Flexibilität
* Testing ist relativ leicht

Nachteile von Skriptsprachen

- Für einige Einsatzgebiete nicht geeignet:
  - Nicht geeignet zum Number-Crunching
  - Weniger geeignet für die systemnahe Programmierung
- Mehr Speicherverbrauch als C-Code
- Laden zum laxen Umgang ein
- Nachteile der dynamischen Typisierung
  - In der Praxis ist das aber meist nicht relevant!

Beispiel (in Perl)

* Zähle die ersten Werte jeder Zeile der übergebenen Dateien, Unix-typisch auch via Pipe; ignoriere Zeilen die mit # beginnen; gebe das Gezählte aus

Bei unsauberem Code kann man sich das sparen

```
#!/usr/bin/perl
use strict;                          # Strikt-Modus für sauberen Code
use warnings;                        # Alle Warnungen einschalten

my %wert_zaehler;                    # Anmeldung der Variable (Optional)

while (<>) {                         # Alle Eingaben einlesen
    next if m(^#);                   # Kommentare ignorieren
    my ($wert) = split;              # Ersten Wert extrahieren
    $wert_zaehler{$wert}++; # Zähler für den Wert erhöhen
}
                                     # Werte sortiert ausgeben
print " Anzahl: $wert_zaehler{$_}, Wert: $_\n"
    foreach sort keys %wert_zaehler;
```

Zur Performance …

- Performance ist meist zu einem Großteil vom Algorithmus oder externen Quellen (z.B. Datenbank) abhängig
  - *http://alvar.a-blast.org/vortraege/perl-workshop/performance-optimierung.pdf*
- Skript-Sprachen haben viele eingebaute, performante Konstrukte wie Reguläre Ausdrücke
- Oft egal, ob etwas 0,1 oder 3 Millisekunden dauert
- 10 Sekunden Ausführzeit nach 30 Minuten Entwicklung, oder 100 Millisekunden nach drei Tagen?

Historisches

Über die Herkunft von Perl, PHP und Python

Geschichte von Perl

- Perl 1.0 wurde am 18. Dezember 1987 von dem Linguisten *Larry Wall* veröffentlicht
- Perl 5.0 erschien am 17. Oktober 1994 und beinhaltete schon alle wesentlichen bis heute genutzten Grundlagen
- Aktuelle Version: 5.10.0 vom 18. Dezember 2007
- Perl 6, eine neue Sprache – nicht die nächste Version von Perl 5 – ist in Entwicklung; auch 5.12 in Entwicklung
- Entwicklung durch die Perl-Community

Geschichte von PHP

- PHP wurde ursprünglich von *Rasmus Lerdorf* entwickelt und entstand als Sammlung von Perl-Skripten
- Die erste Version, PHP/FI, wurde 1995 veröffentlicht
- PHP 3, 1997 veröffentlicht, ähnelte dem heutigen PHP
- Aktuelle Version: 5.2.5 vom 8. November 2007
- Version 6.0 ist in Entwicklung
- Entwicklung vor allem durch die Firma ZEND

Geschichte von Python

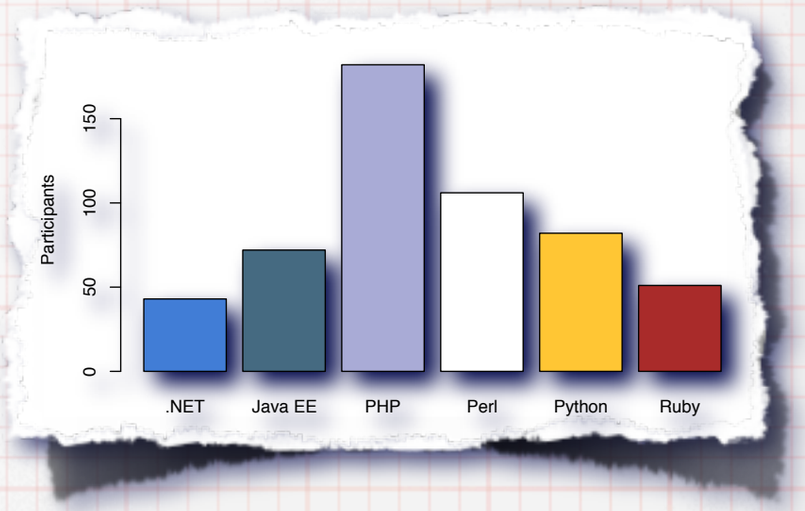
- Python wurde Anfang der 1990er Jahre von *Guido van Rossum* als Programmier-Lehrsprache entwickelt
  - Benannt nach *Monty Python,* nicht nach der Schlange
- Version 1.0 erschien im Januar 1994
- Aktuelle Version: 2.5.2, vom 21. Februar 2008
- Python 3 in Entwicklung; nicht 100% rückwärtskompatibel
- Entwicklung durch die Python-Community

Umfrage zum Sprachen-Vergleich

“Comparing Web Development Platforms Through the Eyes of Professional Developers”

Zusammenfassung der Ergebnisse der Umfrage

Übersicht



- Sprachen in der Umfrage:
  - Ruby, Python, Perl, PHP, Java, .NET
  - „Professionelle Web-Entwickler“ sollten jeweils zwei Sprachen, mit denen sie Erfahrung haben, vergleichen
  - http://www.inf.fu-berlin.de/inst/ag-se/pubs/platforms-surveyTR-2007.pdf

Aufwand

- Wieviel Aufwand ist nötig, um eine Aufgabe zu lösen?
  - Bei Skripsprachen deutlich niedriger als bei Java und .NET, mit Ausnahme von PHP
  - Perl und Python benötigen weniger Aufwand als PHP
  - Python und Perl ähnlich bewertet, mit ganz leichtem Vorteil für Perl
  - Uneinheitliche Meinungen bezüglich PHP im Vergleich zu Java und .NET

Aufwand: Interpretation

- Skriptsprachen erleichtern die Entwicklung sowohl einfacher Tools als auch komplexer Applikationen
  - Sie sind ausdrucksstark, haben wenige Formalitäten und viele einfache aber komplexe Konstrukte
- Schlechtes Abschneiden von PHP: Die Umfrage bezog sich vor allem auf umfangreichere Applikationen
- Uneinheitliche Bewertung PHP zu Java und .NET mit unterschiedlichem Wissensstand der Teilnehmer erklärbar

Lesbarkeit

- Die Teilnehmer wurden befragt, welcher Code in welcher Sprache lesbarer ist
  - Klares Ergebnis: Python-Code gilt als der lesbarste
  - PHP- und Java-Code gilt als am schwersten lesbar
  - Perl, oftmals als *Write-Only-Sprache* verschrien, schneidet mittelmäßig ab
  - Das Vorurteil trifft in der Praxis nicht zu

Lesbarkeit: Interpretation

* Lesbarkeit liegt in erster Linie am Programmierstil des jeweiligen Entwicklers
  * Schlechte Entwickler schreiben in jeder Sprache unlesbaren Code
* Dennoch: Python gilt als besonders gut lesbar, was auch an dem weitgehend erzwungenen Stil liegt
* PHP: Meist Mischmasch von Code und HTML
* Java: geschwätziger als Skriptsprachen

Modifizierbarkeit

* In welcher Sprache lassen sich besonders gut modifizier- und erweiterbare Anwendungen schreiben?
  * Python-Anwendungen sind einfacher modifizierbar als Anwendungen in Java oder PHP
  * Perl-Anwendungen einfacher erweiterbar als solche in PHP oder Java
  * Teilweise stark divergierende Meinungen

Modifizierbarkeit: Interpretation

- Sehr stark abhängig von den Kenntnissen der Entwickler sowie dem verwendeten Stil
  - z.B.: viele Tests verhindern Fehler bei der Erweiterung
- Frameworks verbessern die Modifizierbarkeit
- Java gilt bei vielen Entwicklern als überfrachtet (bloated)
- PHP mangelt es an Modularität (z.B. Namensräume)

Robustheit

- Wie robust sind Anwendungen, die mit einer der Sprachen entwickelt wurden?
  - Einhellige Meinung: Anwendungen, die in PHP entwickelt wurden, sind am wenigsten robust, wenn auch nur knapp
  - Ansonsten wieder ein uneinheitliches Bild, mit ganz leichtem Vorteil für Perl

Robustheit: Interpretation

- PHP gilt als Sprache, bei der schlechter Codierungs-Stil gefördert wird
- Insgesamt mehr vom Stil und von den Entwicklern als von der Sprache abhängig
  - Aber: manche Sprachen erleichtern robusten Code
- Interessant: Kein Vorteil für statisch typisierte Sprachen!
- Auch abhängig von den verwendeten Frameworks

Sicherheit

- Wie sicher sind Applikationen, die in den einzelnen Sprachen entwickelt wurden?
  - Deutlich: PHP ist am unsichersten
  - Ansonsten nur relativ geringe Unterschiede
  - Perl leicht im Vorteil

Sicherheit: Interpretation

* Unsicherheit von PHP liegt auch an deren meist relativ unerfahrenen Anwendern
* Aber: die Sprache hat viele Problemstellen
* Ansonsten liegt vieles an den verwendeten Techniken, Frameworks und Kenntnissen der Entwickler
* Perl: Taint-Mode, DBI; CPAN!

Weiteres

- Skalierbarkeit: Java leicht vorne.
  - Von Hause aus viele Tools vorhanden
- Performance: uneinheitlich, Perl leicht vorne.
  - Kommt meistens auf viele Faktoren an, z.B. Datenbank
  - Java und .NET haben mehr Tools, aber sind auch sehr abhängig davon
  - Skriptsprachen: weniger Tools, auch kaum abhängig

Fazit

- Umfrageergebnisse sind natürlich subjektiv, aber:
  - Skriptsprachen haben Vorteile, insbesondere beim Erstellungs- und Pflege-Aufwand
    - Wurde auch beim Plat_Forms Contest bestätigt
  - PHP steht in vielen Punkten unter Kritik
  - Perl – gelegentlich totgesagt, da kaum Hype und PR – schneidet gut ab
  - Frameworks können die Arbeit erleichtern

Vor- und Nachteile

Perl, PHP und Python

Vorteile Perl

* CPAN: sehr umfangreiches Perl-Modul-Archiv
* Variablendeklaration erzwingbar, Taint-Mode
* Sehr ausdrucksstark und flexibel
* Mächtige Tools für Dokumentation, Testing, *Perl::Tidy*, . . .
* Für fast alles geeignet
* Viele Vereinfachungen

Vorteile PHP

* Für kleine Web-Projekte: sofort loslegen
* Günstiges Hosting
* Viele günstige Entwickler
* Alles notwendige für die Web-Entwicklung ist meist dabei
* Einfacher Einstieg, aber sehr viele Fallstricke
* Größere Projekte zwar möglich, aber sehr viel Disziplin nötig

Vorteile Python

- Es wird viel Wert auf lesbaren Code gelegt
- Relativ stringentes und umfangreiches OO
- Umfangreiche Standardbibliothek mitgeliefert („Batteries Included“)
- Gutes Exception-Handling bereits eingebaut
- Java-Integration mit Jython relativ einfach

Unterschiede Perl <=> Python

- Perl: TIMTOWDI, *There is more then one way to do it*
- Python: *There should be one – and preferably only one – obvious way to do it*

- Perl: viel Freiheit; Freiheit bedeutet Verantwortung
- Python: weniger Verantwortung, weniger Freiheit
- Features: relativ ausgewogen
  - z.B. Taint- und Strict-Mode in Perl, OO in Python

Unterschiede Perl/Python <=> PHP

* PHP hat viele Probleme bei den Sprachstandards:
  * großer Wildwuchs, Inkonsistenzen, (bisher) keine Namensräume, viele Altlasten
* Mit PHP kann man (für kleine) Web-Anwendungen sofort loslegen, ohne Framework; nur fürs Web gut einsetzbar
* PHP mangelt es vielen Features, Sicherheitsprobleme (z.B. Nullbytes und POSIX Reguläre Ausdrücke)
* Schneller Einstieg, später hoher Pflegeaufwand bei PHP

Verfügbare Erweiterungen

* Stand 29. April 2008
  * PHP, PECL: 185; PEAR: 480; Summe: 665 Packages
    * *http://pecl.php.net/* | *http://pear.php.net/*
  * Python, PyPI: 3927 Packages
    * *http://pypi.python.org/pypi*
  * Perl, CPAN: 15136 Distributionen
    * *http://search.cpan.org/*

Schlussrunde

Zusammenfassungen, Kriterien zur Sprachwahl, Tipps & Tricks

Kriterien für die Sprachauswahl

* Sprach-Features: Ausreichend und zur Aufgabe passend?
* Verfügbarkeit von Erweiterungen, Modulen, Frameworks
* Support, Community-Unterstützung?
* Welche Kenntnisse haben die vorgesehenen Entwickler?
* Umfang des Projektes, evtl. spätere Erweiterbarkeit
* Externe Libraries benötigt?
* Zeitkritische Berechnungen?

Tipps zum Schluss

- Wichtig: Sauberer Code
  - Dokumentation
  - Testing
  - Namenskonventionen
  - Planung
  - Module nutzen (z.B. CPAN)
  - Für Perl: *Perl Best Practices* von Damian Conway

Tipps zum Schluss (2)

- Möglichkeiten der Sprache nutzen, Beispiele:
  - Prozedural oder Objektorientiert, oder gemischt
  - Vereinfachungen nutzen: *q, qq, qw, qr; map, grep;* <>
  - Modularisieren, Spaghetti-Code vermeiden …
- Keinen Code in HTML einbetten, Templates nutzen!
  - schwer lesbar, wenig robust, schwer erweiterbar
  - Umgekehrt gilt das gleiche

Danke fürs Zuhören!

Fragen?

Alvar Freude
http://alvar.a-blast.org/
alvar@a-blast.org

http://www.perl-blog.de/