# Konfigurationsdateien – wozu?

## Zusammenfassung

Oft wird die Gesamtbeschreibung eines Programms aufgeteilt in einen Teil der in einer Programmiersprache erstellt wird und einen, hoffentlich kleinen, Teil, der als Konfigurationsdatei zur Laufzeit eingelesen und die Funktionsweise beeinflusst. Hier wird dargelegt, dass man auf solche Dateien verzichten könnte und begründet, warum man das tun sollte.

## Zwingende Gründe für Konfigurationsdateien

### Politische Aspekte

Bei Produkten, die ohne Source ausgeliefert werden, ist eine Konfigurationsdatei zwingend, wenn der Kunde die Funktionsweise auf seine Bedürfnisse anpassen können muss.

Bei Inhouse-Entwicklungen werden Änderungen am Programm anders behandelt als Änderungen an einer Konfigurationsdatei. Erstere müssen ein komplexes Beweilligungsverfahren durchgehen, bevor sie in den Betrieb überführt werden können, währenddem Konfigurationsänderungen durchgewunken werden, selbst wenn ein

```
dbInitCmd = "DELETE * FROM kundenstamm"
```

mindestens so schlimme Konsequenzen haben kann, wie ein Programmierfehler.

### Betrieblicher Aspekt

Leider gibt es so komplexe Programme, deren Dauer eines Neustarts so lange ist, dass man sie lieber ständig laufen lässt. Ein Konfigurationsdatei, die bei Bedarf neu gelesen werden kann erlaubt auf einen Neustart zu verzichten.

## Was sonst, und warum?

Konfigurationsdateien entstehen häufig aus dem Bedürfnis heraus, einige wenige Parameter einfach Ändern zu können. Diese Parameter sollen dann nicht über verschieden Quelldateien verstreut sein, sondern in einer Datei zusammengefasst werden, die so einfach gehalten wird, dass auch ein mit der Programmierung nicht vertrauter Mitarbeiter diese Werte bei Bedarf anpassen kann.

Doch viel zu oft bleibt es nicht dabei. Genauso wie die Anforderungen an ein Programm wächst, wächst sie auch bei der Konfigurationsdatei. Es werden nicht nur immer mehr Parameter hinzugefügt, auch die Anforderungen an die Struktur nimmt zu. Plötzlich wird es notwendig Parameter zu gruppieren. Man möchte die Parameter abhängig von Bedingungen so oder anders setzen. Um Wiederholungen zu meiden möchten man gleichartige Parameter in verschiedenen Gruppen ersetzen können durch einen Verweis auf eine Gruppe die diese Parameter nur einmal erwähnt. Es soll möglich sein den Wert eines Parameters aus dem Wert eines anderen zu berechnen. Und schon hat man die Komplexität einer Programmiersprache erreicht, mit dem Nachteil, dass man sie selbst pflegen muss.

Warum also nicht gleich auf die Konfigurationsdatei verzichten? Das Bedürfnis, die Parameter an einem gemeinsamen Ort zu halten kann man in den meisten modernen Programmiersprachen einfach realisieren. Wird in Java entwickelt, so gibt man vor, dass alle Parameter die man „konfigurierbar“ halten will in eine Klasse Konfiguration.java zusammengefasst sind. Andere

Klassen, welche auf einen solchen Parameter zugreifen sollen, müssen dies per `Konfiguration.getParametername()` tun.

Der Vorteil dieses Ansatzes ist, dass man die gesamte Stärke der Programmiersprache zur Verfügung hat, wenn die Anforderungen wachsen. Aus einem anfänglichen

```
int getMaxThreads() { return 4; }
int getThreadPools() { return 1; }
```

kann wenn notwendig ein

```
int getMaxThreads() { if (isTest()) return 2; else return 10; }
int getThreadPools() { min(1,getMaxThreads()/4); }
```

werden.

Vorteilhaft ist dieses Vorgehen auch deshalb, weil alle Hilfsmittel die eine gute Entwicklungsumgebung bietet, genutzt werden können. So kann es nicht vorkommen, dass man statt eines Integer-Wertes 4.7 angibt, noch kann es vorkommen, dass man sich beim Namen eines Konfigurationsparameter vertippt. In Java und viele anderen Sprachen würde ein solcher Fehler schon bei der Übersetzung des Programms, in einer guten Entwicklungsumgebung sogar schon beim eintippen erkannt.

Alternativ kann die Konfiguration auch als Argument mitgegeben wird, indem, um bei Java zu bleiben, ein interface definiert wird, in dem die Parameter definiert werden. Einer Methode `setConfig(Konfiguration konfiguration)` würde man dann eine Klasseninstanz übergeben, die das interface implementiert.

```
class Main {
 public static void main(String args[]) {
  setConfig(Main.class.forName("Konfiguration").newInstance());
  startApp();
 }
}
```

Eine solche Variante ist auch im Falle einer Applikation denkbar, die man ungern neu starten will. Voraussetzung ist, dass ein Mechanismus existiert, mit dem man zu einem beliebigen Zeitpunkt den Namen der neuen Konfigurationsklasse mitteilen und ein erneutes setzten der Konfiguration erzwingen kann.