

HANSER

# Ship it!

von Jared R. Richardson  
und William A. Gwaltney

ISBN 3-446-40425-2

Leseprobe Kapitel 2.3

Weitere Informationen oder Bestellung  
unter <http://www.hanser.de/3-446-40425-2>  
sowie im Buchhandel

## 3 Der skriptbasierte Build-Prozess

---

Der Build-Prozess wandelt den Quelltext in ein lauffähiges Programm um. Abhängig von der Programmiersprache und der Umgebung bedeutet dies, dass Sie Quelltext kompilieren und Bilder und andere Ressourcen zu einem Paket schnüren. Die Skripte, Programme und Technologien dafür stellen das Build-System dar.

---

*Build-Prozess*

Denken Sie daran, dass wir nicht über das Kompilieren und Bauen innerhalb Ihrer IDE reden. In den meisten Fällen sollten Sie Ihre persönliche Entwickler-IDE *nicht* zum Kompilieren des Projekts verwenden, selbst wenn Sie der Einzige sind, der damit arbeitet (wenn Ihnen das eigenartig erscheint, dann lesen Sie noch einmal Abschnitt 2.1, *Entwicklung im Sandkasten*, auf Seite 20). Wir reden hier über einen Build-Prozess auf Ihrem Rechner, der dem „offiziellen“ Build-Prozess auf dem Build-Rechner entspricht. Dazu eine kleine Geschichte, die das erläutert.

Entwickler Billy möchte sein Projekt bauen. Er startet also seine IDE, öffnet das seiner Meinung nach richtige Projekt und weist die IDE an, das Projekt neu zu bauen. Dann kopiert er das fertige Programm in ein separates Verzeichnis und lässt daraus einen Installer erstellen.

Um sich zu vergewissern, dass alles richtig funktioniert, führt er den Installer aus. Dieser stürzt sofort ab. Billy erinnert sich, dass sein Produkt auf die externe `Widget`-Bibliothek angewiesen ist und er vergessen hat, diese in das separate Verzeichnis zu kopieren. Also holt er das nach. Nachdem er den gesamten Vorgang ein zweites Mal durchlaufen hat, fällt ihm ein, dass er ja auch die von seinem Programm verwendeten Grafiken kopieren muss ...

Sieht nach einem Marathon aus, nicht wahr? Billy steht ein langer Abend bevor, an dem er wieder unbezahlte Überstunden macht und die Wiederholungen von *Buffy* verpasst.

Auch Bob baut heute sein Produkt. Bob wechselt in den Ordner, der seinen Quelltext enthält, und führt sein Build-Skript mit einem einzeiligen Befehl (z.B. `ant build_installer` oder `make all`) aus. Das Skript baut das Produkt (wobei es die aktuellen Versionen von allem holt, wovon das Programm abhängt), erstellt einen Installer dafür und testet ihn. Und schon ist Bob fertig.

Sie können an diesen Szenarien sehr gut den Unterschied zwischen einem automatisierten Build-System und der manuellen Erstellung Ihres Produktes erkennen. Billy unterliefen zahlreiche Fehler, die Bob durch die Automatisierung vermeiden konnte. Die Schritte, die Billy vergessen hatte, wurden von Bobs Skript automatisch (und bei jedem Build-Lauf gleich) ausgeführt.

Während Billy – angesichts der vielen Nacharbeit und Überstunden – vielleicht als der engagiertere Entwickler angesehen wird, so würden wir doch lieber mit Bob arbeiten (oder Bob sein!). Indem Sie Ihren Build-Prozess automatisieren, werden die einzelnen Schritte nicht nur genauer und weniger fehleranfällig, sondern Sie kommen auch pünktlich nach Hause – so wie Bob.

Sie haben eine Vielzahl von Möglichkeiten, wie Sie Ihr Produkt bauen können. Die Liste der auszuführenden Schritte könnte z.B. folgendermaßen aussehen:

1. Kompilieren des Quelltextes
2. Kopieren des kompilierten Programms in ein separates Verzeichnis
3. Hinzufügen der neuesten Versionen der benötigten Fremdbibliotheken (z.B. Datenbanktreiber und Parser)
4. Kopieren benötigter Ressource-Dateien, wie z.B. HTML und Grafiken
5. Hinzufügen der Hilfe-Dateien
6. Erstellen des Installers

Wenn Ihr Build-Prozess *irgendwelche* manuellen Schritte enthält, haben Sie ein Problem. Die Frage ist nur, ob Sie sich im Vorfeld die Zeit nehmen und sich mit diesem Problem auseinander setzen, oder ob Sie die Zeit später aufwenden, wenn Sie die einzelnen Schritte immer wieder per Hand durchführen und Sie sich dann mit den unvermeidbaren Fehlern herumschlagen müssen, wenn ein Schritt nicht oder falsch ausgeführt wurde. Die erste Möglichkeit stellt eine kluge Investition in einem frühen Stadium eines Projekts dar. Die zweite ist ein schwarzes Loch; sie wird im weiteren Verlauf Ihres Projekts eine stete Quelle von Frustration und Problemen sein.

***Tipps 4: Verwenden Sie vom ersten Tag an ein Build-Skript***

---

Verwenden Sie zumindest eine Batch-Datei oder ein Shell-Skript (es gibt natürlich bessere Möglichkeiten, und wir werden in Kürze darauf eingehen).

Aber warum, so werden sie einwenden, können Sie die Schritte des Build-Prozesses nicht einfach von Ihrer IDE ausführen lassen? Nun, diese Möglichkeit haben Sie natürlich; Sie werden dabei aber auf folgende Probleme stoßen:

- Es ist unwahrscheinlich, dass Ihr Build-Rechner dieselbe IDE verwendet (die meisten IDEs lassen sich entweder gar nicht oder nur umständlich in einem Batch-Modus betreiben).
- Ihr gesamtes Team – sowohl Experten als auch Anfänger – wäre gezwungen, mit derselben IDE zu arbeiten. Manchmal ist das akzeptabel; meist schaffen Sie sich damit aber mehr Probleme, als Sie lösen.
- Auch wenn alle Teammitglieder dieselbe IDE nutzen, so kann es doch schwierig sein, Änderungen in der Entwicklungsumgebung (wie z.B. eine neue Bibliothek oder geänderte Compilerschalter) zu verbreiten.

Diese Probleme sind nicht unüberwindlich, aber trotzdem unangenehm. Deshalb finden wir es viel einfacher, die Automatisierung des Build-Prozesses von der IDE-Welt zu trennen. Es ist jedoch wichtig, dass der Build-Prozess – in all seiner Komplexität – mit einem einzigen Befehl gestartet werden kann.

Wenn Sie Ihr Projekt nicht mit einem einzigen Befehl bauen können, dann wird sich der Großteil des Teams nicht damit beschäftigen. So wie bei den meisten Aufgaben im Leben gilt auch hier, dass sie nur durchgeführt werden, wenn sie einfach sind. Mit der Möglichkeit, Ihr Produkt einfach und konsistent von jedem Entwicklungsrechner aus zu bauen, verfügen Sie über eines der wichtigsten Kennzeichen einer professionellen Softwareentwicklung.

Leider läuft es aber nicht immer so. In unserer bewegten Vergangenheit gab es eine Firma, in der das Produkt per Knopfdruck innerhalb einer IDE auf einem speziellen Rechner gebaut wurde. Dann verließ der Entwickler, der den Build-Prozess eingerichtet hatte, plötzlich das Unternehmen. Keiner der verbliebenen Teammitglieder wusste, wie das Produkt ohne diesen speziellen Rechner und seinen „magischen Build-Knopf“ gebaut werden konnte. Die Manager waren geschockt, als wir ihnen davon berichteten – das hatten sie nicht gewusst.

Sorgen Sie dafür, dass Sie Ihr Produkt auf jedem Rechner der Firma in der gleichen Art und Weise bauen können. Wenn Sie nicht wissen, wie Sie an Ihrem Rechner den Build-Prozess starten, dann ist die Wahrscheinlichkeit hoch, dass es auch sonst niemand weiß.

### ***Tipps 5: Jeder Rechner kann ein Build-Rechner sein***

Zur Not kann jeder Entwicklungsrechner den Build-Rechner ersetzen, indem er dieselben (eingechekten) Skripte verwendet. Ziel ist es, dass das Ergebnis des Build-Prozesses auf einem beliebigen Entwicklungsrechner mit dem Ergebnis des Build-Rechners identisch ist (außer vielleicht in Sachen wie Zeitstempel, IP-Adresse oder Name des Rechners).

Build-Skripte müssen nicht kompliziert sein. Das folgende Beispiel zeigt ein vollständiges und verwendbares Ant-Skript, das trotzdem lesbar und leicht verständlich ist (trotz der spitzen Klammern von XML):

```
<?xml version="1.0" ?>
<project name="SimpleExample" default="doall">

  <property name="build.dir" value="./build" />
  <property name="dist.dir" value="./dist" />

  <target name="init">
    <mkdir dir="${build.dir}" />
    <mkdir dir="${dist.dir}" />
  </target>

  <target name="compile" depends="init">
    <javac srcdir="." destdir="${build.dir}"/>
  </target>

  <target name="dist" depends="compile">
    <jar destfile="${dist.dir}/SimpleExample.jar"
        compress="true">
      <fileset dir="${build.dir}" />
    </jar>
  </target>

  <target name="doall" depends="dist">
  </target>

</project>
```

## Wie fange ich an?

Wenn Sie gerade ein Produkt geerbt haben, das über keinen automatisierten Build-Prozess verfügt, dann sollten Sie umgehend die folgenden Schritte unternehmen:

1. Lassen Sie Ihr Produkt von einem Teammitglied manuell bauen, und machen Sie sich dabei Notizen.
2. Definieren Sie die einzelnen Build-Schritte.
3. Wählen Sie ein Build-Werkzeug aus, aber seien Sie bereit, auf andere Optionen zurückzukommen, falls sich das gewählte Werkzeug als zu beschwerlich erweisen sollte.
4. Schreiben Sie Ihr Skript; ersetzen Sie dabei die manuellen Schritte einen nach dem anderen.
5. Lassen Sie das Skript auf einem anderen Rechner ablaufen. Auf diese Weise werden Sie unbeabsichtigte Abhängigkeiten von Ihrer Umgebung entdecken.
6. Lassen Sie jetzt das Skript von einem anderen Teammitglied ohne Ihre Hilfe ausführen.

Nachdem Sie diese Schritte durchgeführt haben, werden Sie über ein Skript verfügen, das für jedermann funktionieren sollte.

## Mache ich es richtig?

Wenn Ihr Build-Prozess richtig passt, dann bauen Sie Ihr gesamtes Produkt ...

- mit einem Befehl
- aus Ihrer Versionsverwaltung heraus
- auf jedem Arbeitsplatzrechner
- ohne spezielle Anforderungen an die Umgebung (wie z.B. bestimmte Netzlaufwerke)

Wenn Sie Probleme mit einem dieser Punkte haben, dann sehen Sie sich Ihren Build-Prozess noch einmal genauer an.

## Warnzeichen

- Ihr Build-Prozess enthält noch manuelle Schritte.

- 
- Ihr Build-Skript muss geändert werden, damit es auf anderen Rechnern läuft.
  - Nur wenige Teammitglieder wissen, wie man das Build-Skript editiert.
- 
-