PHP2CFML und wieder zurück

Erfasst am : 15. März 2007 10:55 | Erfasst von : Andreas Schuldhaus
Verknüpfte Kategorie(n): ColdFusion, PHP2CFML

Nein, dies ist keine neue Kategorie für Beiträge mit dem Thema: Warum ColdFusion? Warum PHP? Warum doch? Warum nicht? Und wieso überhaupt? Dazu gibt es wohl genügend Beiträge und Streitschriften im Internet.

Hier wird nicht mehr die Frage nach dem **"Warum?"** gestellt, sondern nach dem **"Wie?"** einer Portierung von Applikationen in beide Richtungen.

Dennoch sehe ich schon, wie sich zwei religiös fanatische Entwicklergemeinden hämisch grinsend an die Stirn tippen und lachend auf mich zeigen. ColdFusion Applikationen zu PHP migrieren? Warum in aller Welt sollte das Irgendwer wollen? PHP Programme zu ColdFusion portieren? Wozu soll denn das gut sein?

Lasst mich etwas ausholen. Es gibt Programmiersprachen, Frameworks, Design Patterns, Applikationsserver, Systemumgebungen, Philosophien, Applikationsdesigner, Gurus, geniale Entwickler ... - und Kunden. Oops! Fast hätten wir diese Kleinigkeit übersehen.

Da programmiert man gerade eine Web 5.0 Killer-Applikation, aber die Kunden sind viel weniger daran interessiert, dass wir im Rahmen eines von ihnen bezahlten Projektes den IT-Nobelpreis bekommen, als daran, dass die Aufgabenstellung für sie optimal gelöst wird und ihnen die Software zukünftig das Leben erleichtert und Kosten sparen hilft.

Wie schön wäre die Vorstellung: hier der Entwickler als Künstler, dort der Kunde als großzügiger Mäzän. Freie Wahl der Stilmittel und Plattformen. Wir wissen alle, dass die Praxis anders aussieht. Wieviele ColdFusion Applikationen sind nicht verkauft worden, weil Kunden ColdFusion nicht einmal buchstabieren konnten, also gar nicht kannten? Wieviele kompliziert heterogene Systeme entstanden nur deshalb, weil plötzlich eine passende PHP Applikation auftauchte, die künstlich in eine ColdFusion dominierte Applikationslandschaft eingepflanzt werden musste?

An diesem Punkt wollen wir ansetzen und Erfahrung von ATGInfotech auf diesem Gebiet einbringen.

Wozu soll Migration nun gut sein? Nun ja. Wir haben ja bereits die optimale Lösung, aber wir stehen vor dem Problem, dass ein weiterer potentieller Auftraggeber so gar nicht mit der Plattform kompatibel ist, auf der unsere Lösung läuft. Der Kunde hat möglicherweise Berater, die gerne Ihre eigenen Lösungen verkaufen wollen, auch wenn diese erst nach kostenintensiven Anpassungen zu 100% dem Kundenbedürfnis entsprechen. Ich brauche nicht zu erwähnen, dass deren Applikationen natürlich immer in der bestehenden Systemumgebung lauffähig sind. Also wieso nicht über den Tellerrand hinausschauen und "Augen zu und durch!".

Wer es wagt, der hat ja schon die komplette Lösung, das fertige Pflichtenheft und einen voll funktionsfähigen, bereits produktiv eingesetzten "Prototyp" zur Verfügung. So erspart man sich einen ganz wesentlichen, wichtigen und vor allem zeitaufwändigen Abschitt in der Applikationsentwicklung und kann sofort zur Migration und Codierung

schreiten. Und vielleicht hat man bald auch einen stark erweiterten Absatzmarkt für die Software erschlossen.

Also los gehts mit dem ersten Beispiel, hier der hochkomplizierte PHP-Code:

```
<?php

// dings auf 1 setzen

$dings = 1;

?>
```

Und jetzt ab in CFML damit:

```
<!--- dings auf 1 setzen --->
<cfset dings = 1>
```

Wow. Damit ist für den Anfang schon Einiges geschafft. Viel vorteilhafter erweist es sich jedoch, wenn man in ColdFusion Gebrauch von <cfscript> macht. Dann sieht es nämlich wie folgt aus und ist noch viel einfacher zu implementieren:

```
<cfscript>

// dings auf 1 setzen

dings = 1;

</cfscript>
```

Na, das sieht doch gleich viel besser und ähnlicher aus. So können ganze Blocks von Code übernommen werden. Nachträglich entfernen wir alle (PHP)$-Zeichen aus dem Bereich und bereinigen/ändern die Operatoren, dort wo es erforderlich ist. Dasselbe gilt natürlich umgekehrt ($-Zeichen hinzufügen), wenn es von CFML nach PHP gehen soll.

Aus diesem Vorgang leiten wir die Migrationsregel Nr. 1 ab:

- Für höchstmögliche Portabilität sollten Script-basierte Code-Teile den Vorzug vor Tag-basierten Code-Teilen erhalten.

Wir werden hier laufend Code-Snippets und Beispiele unter dem Gesichtspunkt der Portabilität veröffentlichen. Es ist auch geplant, die komplette Beispielapplikation QTasker (ein rudimentärer Taskplaner) aus einem unserer Vorträge, sowohl in CFML als

auch in der portierten PHPQTasker Version hier zu veröffentlichen und Schrittweise die Lösungsansätze darzustellen.