

# Funktionale Programmierung verstehen

Konzepte und Entwurfsmuster für guten Code

# DAS INHALTS- VERZEICHNIS

» Hier geht's  
direkt  
zum Buch

# Inhalt

Vorwort .....	9
<b>1 Was ist funktionale Programmierung?</b> .....	<b>11</b>
<b>1.1 Unveränderlichkeit</b> .....	<b>13</b>
<b>1.2 Referenzielle Transparenz</b> .....	<b>16</b>
<b>1.3 Funktionen höherer Ordnung</b> .....	<b>18</b>
<b>1.4 Lazy Evaluation</b> .....	<b>19</b>
<b>1.5 Funktionale Denkweise</b> .....	<b>21</b>
<b>1.6 Die Vorteile der funktionalen Programmierung</b> .....	<b>22</b>
1.6.1 Funktionale Programmierung kann die Produktivität steigern .....	23
1.6.2 Funktionale Programmierung macht Spaß! .....	24
1.6.3 Scala .....	25
<b>1.7 Fazit</b> .....	<b>26</b>
<b>2 Mathematische Grundlagen</b> .....	<b>27</b>
<b>2.1 Mengenlehre</b> .....	<b>27</b>
2.1.1 Funktionen .....	28
2.1.2 Funktionstypen .....	30
<b>2.2 Grundlagen der Informatik</b> .....	<b>32</b>
2.2.1 Anonyme Funktionen .....	32
2.2.2 Funktionen als First-Class-Objekte .....	33
<b>2.3 Fazit</b> .....	<b>34</b>

### 3 Kategorientheorie und Entwurfsmuster 35

---

<b>3.1</b>	<b>Kategorientheorie und Funktionsmuster</b> .....	37
	3.1.1 Ein kurzer Abriss .....	38
	3.1.2 Objekte und Morphismen .....	39
	3.1.3 Ein Beispiel für eine Kategorie .....	40
	3.1.4 Die Scal-Kategorie .....	44
	3.1.5 Funktoren .....	45
	3.1.6 Funktoren in der Programmierung .....	49
<b>3.2</b>	<b>Funktionsmuster</b> .....	51
	3.2.1 Das Funktor-Muster .....	51
	3.2.2 Monoide .....	53
	3.2.3 Natürliche Transformationen .....	55
	3.2.4 Monaden .....	57
<b>3.3</b>	<b>Fazit</b> .....	60

### 4 Funktionale Datenstrukturen 61

---

4.1	Die Option-Datenstruktur .....	62
4.2	Die Try-Datenstruktur .....	67
4.3	Die Either-Datenstruktur .....	68
4.4	Funktionen höherer Ordnung .....	70
4.5	Monaden und Scala-Abstraktionen .....	72
4.6	Traditionelle Datenstrukturen .....	74
	4.6.1 Unveränderlichkeit und Historie .....	74
	4.6.2 Trägheit .....	75
4.7	Fazit .....	75

### 5 Unveränderlichkeit im Detail 77

---

5.1	Veränderliche und unveränderliche Variablen .....	77
5.2	Rekursion .....	78
	5.2.1 Ein Beispiel für eine verkettete Liste .....	79

<b>5.3</b>	<b>Endrekursion</b> .....	86
<b>5.4</b>	<b>Weitere Beispiele für die Leistungsfähigkeit der fold-Funktion in Scala</b> .....	91
<b>5.5</b>	<b>Verbindung zwischen »fold« und Monoiden</b> .....	92
<b>5.6</b>	<b>Vertiefende Informationen zu Funktionen höherer Ordnung</b> .....	96
5.6.1	Von »map« zu »flatMap« .....	98
<b>5.7</b>	<b>Fazit</b> .....	101

## **6 Nebenläufigkeit** 103

---

<b>6.1</b>	<b>Streams</b> .....	107
<b>6.2</b>	<b>Akka-Streams</b> .....	108
6.2.1	Quelle .....	108
6.2.2	Fluss .....	109
6.2.3	Senke .....	110
<b>6.3</b>	<b>Weitere Informationen zu Streams</b> .....	111
<b>6.4</b>	<b>FS2: funktionale Streams für Scala</b> .....	112
<b>6.5</b>	<b>Fazit</b> .....	114

## **7 Wie geht es weiter?** 115

---

<b>7.1</b>	<b>Den reinen Weg einschlagen</b> .....	115
7.1.1	Die IO-Monade .....	117
7.1.2	Den Mittelweg beschreiten .....	120
7.1.3	JVM-Sprachen .....	120
7.1.4	.NET-Sprachen .....	121
7.1.5	Typklassen .....	121
<b>7.2</b>	<b>Fazit</b> .....	128

<b>Anhang</b>	129
<hr/>	
<b>A</b>	
<b>Scala</b> .....	129
A.1 Annahmen .....	130
A.2 Überblick .....	130
A.3 »var« und »val« .....	132
A.4 Klassen und Objekte .....	132
A.5 Funktionen .....	133
A.6 Case-Klassen .....	134
A.7 Fazit .....	149
<b>B</b>	
<b>Python-Special</b> .....	151
B.1 Interne und externe Bibliotheken .....	152
B.2 Typsicherheit .....	153
B.3 Sprachelemente .....	155
B.4 Codebeispiele zu Kapitel 1, »Was ist funktionale Programmierung?« .....	160
B.5 Codebeispiele zu Kapitel 3, »Kategorientheorie und Entwurfsmuster« .....	161
B.6 Codebeispiele zu Kapitel 4, »Funktionale Datenstrukturen« .....	167
B.7 Codebeispiele zu Kapitel 5, »Unveränderlichkeit im Detail« .....	173
B.8 Codebeispiele zu Kapitel 6, »Nebenläufigkeit« .....	176
B.9 IO-Monaden .....	179
B.10 Wie geht's weiter? .....	180
 Index .....	 183