

Auf einen Blick

TEIL I Die Entwicklungsumgebung kennenlernen	27
1 Erste Schritte im Entwicklungsprozess	29
2 Bewährte Konzepte bei der Entwicklung mit WordPress	57
3 Überall alles verändern: Die Hooks als Grundkonzept in WordPress ...	93
TEIL II Plugins, Themes und Blöcke entwickeln	127
4 Ein Plugin erstellen	129
5 Ein Theme erstellen	175
6 Einstellungsseiten generieren und das Dashboard erweitern	335
7 Performance und Sicherheit	379
8 CSS und JavaScript in WordPress	429
9 Einen Block erstellen	495
10 Internationalisierung	555
11 Fehler finden und beheben: Das Debugging	581
TEIL III Ihr Projekt erweitern und individualisieren	611
12 Schöne URLs mit der Rewrite-API	613
13 Mit eigenen Inhaltstypen arbeiten	641
14 Externe Daten nachladen mit der HTTP-API	697
15 Die Schnittstelle zur Außenwelt: Die REST-API	719
16 Benutzer- und Rechtemanagement	785
17 Zugriff auf die Datenbank	825
18 Dateien auf dem Laufwerk manipulieren	873
19 Zeitgesteuerte Aufgaben (Cronjobs)	897
20 WordPress auf der Kommandozeile: Die WP-CLI	911
21 Mehrere Websites in einer Installation: Die Multisite-Umgebung	937
22 Inhalte dynamisch einfügen mit der Shortcode-API	961
23 Alles gesund? Die Site-Health-API	971

Inhalt

Materialien zum Buch	20
Vorwort	21

TEIL I Die Entwicklungsumgebung kennenlernen

1 Erste Schritte im Entwicklungsprozess	29
1.1 WordPress installieren	31
1.1.1 WordPress downloaden	32
1.1.2 Der Upload zum Webhosting-Anbieter	33
1.1.3 Den Datenbank-Zugang einrichten	33
1.1.4 Die 5-Minuten-Installation	34
1.2 Lokale Laufzeitumgebungen	42
1.2.1 Die lokale Laufzeitumgebung einrichten	43
1.3 Entwickeln mit dem Texteditor?	45
1.4 Software-Entwicklungstools einrichten	47
1.4.1 PhpStorm einrichten	48
1.4.2 Visual Studio Code einrichten	52

2 Bewährte Konzepte bei der Entwicklung mit WordPress

2.1 Der Slug: Den Namen für das Projekt richtig wählen	57
2.2 Den richtigen Lizenzvertrag wählen	59
2.3 Coding-Standards: Welche sollten verwendet werden?	62
2.3.1 Accessibility Coding Standards	64
2.3.2 CSS Coding Standards	65
2.3.3 HTML Coding Standards	67
2.3.4 JavaScript Coding Standards	69
2.3.5 PHP Coding Standards	72
2.3.6 Quellcode automatisch ausrichten	74

2.4 Dokumentationsstandards: Welche sollten verwendet werden?	77
2.4.1 Die PHP-Dokumentationsstandards	77
2.4.2 Die JavaScript-Dokumentationsstandards	80
2.4.3 Dokumentationen automatisch erstellen lassen	82
2.5 Präfixe und Namespaces	83
2.6 Objektorientierung in WordPress	85
2.7 Empfohlene Datei- und Verzeichnisstrukturen	85
2.7.1 Minimal nötige Dateien	86
2.7.2 Empfohlene Verzeichnisstruktur	86
2.8 Online-Nachschlagewerke für die WordPress-Entwicklung	88
2.8.1 Offizielle Dokumentation	89
2.8.2 Am WordPress-Kern mitentwickeln	89
2.8.3 Neuigkeiten	90
2.8.4 Chats und Foren	91

3 Überall alles verändern: Die Hooks als Grundkonzept in WordPress

3.1 Was sind Actions und Filter?	94
3.1.1 Actions	94
3.1.2 Filter	99
3.1.3 Alternative Hook-Funktionen	102
3.1.4 Besondere Hooks	103
3.2 Häufig genutzte Actions und Filter in WordPress	104
3.2.1 the_content	104
3.2.2 the_title	105
3.2.3 init	106
3.3 Filter und Actions finden	107
3.3.1 Die Suchfunktion des Entwicklungstools	108
3.3.2 Offizielle Referenzen	109
3.3.3 Inoffizielle Quellen	109
3.3.4 Plugin »Query Monitor«	110
3.4 Arbeiten mit Filtern und Actions	111
3.4.1 Die Art der Übergabe von Funktionen	111
3.4.2 Filter- und Action-Funktionen entfernen	113
3.4.3 Variable Hooks	119
3.4.4 Schnellrückgabe-Funktionen	120

3.4.5	Habe ich oder habe ich nicht?	121
3.4.6	Wurde der Hook schon aufgerufen?	122
3.4.7	Wenn die falschen Hooks benutzt wurden	124

TEIL II Plugins, Themes und Blöcke entwickeln

4 Ein Plugin erstellen

129

4.1	Grundwissen zu Plugins	129
4.1.1	Was ist ein WordPress-Plugin?	129
4.1.2	Die Vorteile von Plugins	130
4.1.3	Wo Plugins zu finden sind	132
4.1.4	Wie Plugins installiert und aktiviert werden	134
4.1.5	APIs: So interagieren Plugins mit dem WordPress-Kern	135
4.1.6	Die verschiedenen Plugin-Typen	142
4.2	Das erste eigene Plugin	144
4.2.1	Das Grundgerüst eines Plugins	145
4.2.2	Die Aktivierungs- und Deaktivierungsfunktionen nutzen	150
4.2.3	Einen Cronjob durchführen	154
4.2.4	Keine Spuren hinterlassen: Die Deinstallationsroutine	156
4.2.5	Inhalte anzeigen	159
4.3	Pfadangaben	162
4.3.1	Der Pfad zum Plugin-Verzeichnis	163
4.3.2	Die URL zum Plugin-Verzeichnis	164
4.3.3	Die Basisbezeichnung	165
4.3.4	Weitere nützliche Pfadangaben	166

5 Ein Theme erstellen

175

5.1	Grundwissen zu Themes	175
5.1.1	Was ist ein Theme?	175
5.1.2	Abgrenzung zu Plugins	179
5.1.3	Abgrenzung zu Blöcken	181
5.1.4	Wo Themes zu finden sind	183
5.1.5	Häufig heruntergeladene Themes bei w.org	185
5.1.6	Wie Themes installiert und aktiviert werden	186

5.2 Crashkurs: Gutes Webdesign	188
5.2.1 Was muss eine Website können?	188
5.2.2 Wahrnehmungsprinzipien und deren Einsatz im Webdesign	189
5.2.3 Allgegenwärtige Definitionen und Normen	193
5.2.4 Die richtigen Farben wählen	195
5.2.5 Die richtigen Schriften wählen	199
5.2.6 Kommunizieren mit Bildern, Videos und Icons	204
5.2.7 Zusammenfassung	207
5.3 Ein Classic-Theme erstellen	208
5.3.1 Die Datei style.css	209
5.3.2 Theme-Funktionen festlegen: functions.php	214
5.3.3 Design und Layout festlegen: theme.json	216
5.3.4 Einstellungen für Themes festlegen	222
5.3.5 Das WordPress-Template-System	235
5.3.6 Erforderliche Templates erfassen und erstellen	239
5.3.7 Funktionen für Templates und der Loop	240
5.3.8 Konditionale Abfragen	267
5.4 CSS für Themes	269
5.4.1 Pfadangaben für Themes	269
5.4.2 CSS-Dateien in das Theme einbinden	272
5.5 Seitenleisten und Widgets erstellen	286
5.5.1 Seitenleisten registrieren	287
5.5.2 Widgets zu Seitenleisten hinzufügen	289
5.5.3 Sidebar im Theme anzeigen	290
5.5.4 Ein eigenes Widget erstellen	294
5.6 Menüs erstellen	301
5.6.1 Menüpositionen registrieren	301
5.6.2 Ein Menü im Frontend ausgeben	302
5.6.3 Beispiel-Theme mit CSS ergänzen	303
5.7 Wenn Themes Kinder bekommen: Child-Themes	306
5.7.1 style.css: Die Verbindung zum Eltern-Theme	307
5.7.2 Die functions.php eines Child-Themes	307
5.7.3 Pfadangaben für Child-Themes	307
5.7.4 Die Template-Dateien des Child-Themes	307
5.8 Ein Block-Theme entwickeln	308
5.8.1 Was ist ein Block-Theme?	308
5.8.2 Hauptkonfiguration mit der theme.json	311
5.8.3 Template-Editor und Template-Übersicht	317
5.8.4 Full-Site-Editing	319

5.8.5	CSS in Block-Themes	324
5.8.6	Style-Variationen	328
5.8.7	Block-Vorlagendateien	329
5.9	Themes testen	329
5.9.1	HTML und CSS validieren	330
5.9.2	Visuelles testen	332
5.10	Zusammenfassung	333

6 Einstellungsseiten generieren und das Dashboard erweitern

335

6.1	Haupt- und Untermenüs erstellen	335
6.1.1	Ein Hauptmenü erstellen	336
6.1.2	Ein Untermenü erstellen	340
6.2	Konsistenz im Layout: Die WordPress-UI nutzen	342
6.2.1	Integrierte Icons: Inhalte aufpeppen mit den Dashicons	342
6.2.2	Schickes Box-Design mit der Metabox-API	343
6.2.3	Richtig, falsch oder Warnung: Hinweisnachrichten in WordPress	350
6.2.4	Das Layout von Überschriften, Buttons, Formularfeldern, Tabs und Tabellen	352
6.3	Einstellungsfelder generieren	358
6.3.1	Werte speichern und abrufen mit der Options-API	358
6.3.2	Einstellungsfelder generieren mit der Settings-API	362
6.3.3	Fehlermeldungen ausgeben	371
6.4	Dashboard-Widgets erstellen	373
6.4.1	Registrieren eines Dashboard-Widgets	373
6.4.2	Konfigurieren eines Dashboard-Widgets	376

7 Performance und Sicherheit

379

7.1	WordPress-Projekte absichern	379
7.1.1	Was bedeutet Sicherheit?	379
7.1.2	Benutzerrechte überprüfen	381
7.1.3	Cross-Site Request Forgery Protection (CSRF)	384
7.1.4	Datei-Uplands validieren	389
7.1.5	Benutzereingaben neutralisieren	393

7.1.6	Deserialisierung der vom Benutzer bereitgestellten Eingaben	398
7.1.7	Sichere SQL-Abfragen erzeugen	400
7.1.8	Verwendung bestimmter PHP-Funktionen mit benutzerdefinierten Eingaben	403
7.1.9	Offenlegung sensibler Informationen	405
7.1.10	Schwachstellen bei Dateizugriff und -nutzung	406
7.1.11	Weitere Risiken	409
7.1.12	Das Security-Mindset schulen	410
7.2	Den Turbo aktivieren: Mehr Performance in WordPress-Projekten	411
7.2.1	Daten zwischenspeichern mit der Cache-API	413
7.2.2	Temporären Speicher von WordPress nutzen: die Transient-API	420

8	CSS und JavaScript in WordPress	429
8.1	Skripte registrieren, einhängen und entfernen	431
8.1.1	Skriptdateien registrieren	432
8.1.2	Skriptdateien einbinden	435
8.1.3	Skripte aushängen und entfernen	438
8.1.4	Inline-JS und -CSS: Skripte direkt in HTML einbetten	438
8.1.5	Zusatzdaten für JavaScript- und Styles-Dateien	440
8.2	Integrierte JS- und CSS-Dateien in WordPress	441
8.2.1	Integrierte JavaScript-Dateien in WordPress	441
8.2.2	Integrierte CSS-Dateien in WordPress	443
8.3	Backbone und Underscore	445
8.4	Crashkurs jQuery	446
8.4.1	Was ist jQuery und wofür wird es benötigt?	446
8.4.2	Einbinden von jQuery in WordPress-Projekte	447
8.4.3	Das jQuery-Objekt	450
8.4.4	Einsatzbereitschaft	450
8.4.5	Arbeiten mit Selektionen	452
8.4.6	Ajax	453
8.4.7	Hilfsmethoden	458
8.4.8	Verkettung	459
8.4.9	Tree-Traversing in jQuery	460
8.5	Crashkurs React	460
8.5.1	Was ist React und wofür wird es benötigt?	461
8.5.2	Einbinden von React in WordPress-Projekte	462
8.5.3	JavaScript Syntax Extension	465

8.5.4	Funktions- und Klassenkomponenten	467
8.5.5	Die Props	467
8.5.6	Der State	468
8.5.7	Der Lifecycle	469
8.5.8	Ajax in React	471
8.5.9	Events	477
8.5.10	React Developer Tools	483
8.6	Internationalisierung	485
8.6.1	Der alte Weg zur Internationalisierung	485
8.6.2	Der neue Weg zur Internationalisierung	487
8.6.3	Strings in JavaScript übersetzen	488
8.7	Hintergrundaktivitäten mit der Heartbeat-API	489
8.7.1	Daten an den Server senden	490
8.7.2	Daten auf dem Server verarbeiten	490
8.7.3	Zurückgesendete Daten im Browser empfangen	491

9	Einen Block erstellen	495
9.1	Grundwissen zu Blöcken	496
9.1.1	Vergleich: Classic- und Block-Editor	497
9.1.2	Blöcke sind WordPress-Plugins	499
9.1.3	Der Technologie-Stack bei Block-Plugins	500
9.2	Einen Block erstellen	501
9.2.1	Die Entwicklungsumgebung einrichten	501
9.2.2	Einen Block registrieren	502
9.2.3	Die block.json-Datei	505
9.2.4	Der Build-Vorgang	507
9.2.5	Der Einstieg in JavaScript	511
9.2.6	Den Block clientseitig registrieren	511
9.2.7	Block-Attribute	513
9.2.8	Einstellungsfelder generieren	518
9.2.9	Einstellungen speichern	520
9.2.10	Speichern der HTML-Repräsentation	523
9.2.11	Dynamischer Block-Inhalt	526
9.2.12	Dynamischen Inhalt in den Block-Editor laden	527
9.2.13	Arbeiten mit zusätzlichen Paketen	530
9.2.14	Arbeiten mit CSS-Styles von Blöcken	531
9.2.15	Styles vom Block-Editor übernehmen	532
9.2.16	Eigene Styles definieren	535

9.2.17	Noch mehr Design mit Block-Variationen	538
9.2.18	Veraltete statische Blöcke	542
9.2.19	Blöcke in Seitenleisten	546
9.3	Block-Vorlagen erstellen	546
9.3.1	Eine Block-Vorlage registrieren	547
9.3.2	Kategorien für Blockvorlagen	550
9.3.3	Vorlagen im Theme	552
9.3.4	Vorlagen und Kategorien entfernen	553
10	Internationalisierung	555
10.1	Grundwissen	556
10.2	Übersetzungsdateien und Gettext	557
10.3	Ein Plugin zur Übersetzung vorbereiten	557
10.3.1	Übersetzung von MU-Plugins	559
10.4	Ein Theme zur Übersetzung vorbereiten	560
10.5	Übersetzungsfunktionen	561
10.5.1	Zeichenketten übersetzen und zurückgeben	562
10.5.2	Mehrdeutige Zeichenketten übersetzen	563
10.5.3	Zeichenketten mit Pluralisierung übersetzen	563
10.5.4	Zeichenketten übersetzen und ausgeben	564
10.5.5	Zeichenketten übersetzen und escapen	565
10.6	JavaScript-Inhalte (und Blöcke) übersetzen	567
10.7	Eine Übersetzungsdatei erstellen	569
10.7.1	Software zum Übersetzen	569
10.7.2	Ein Plugin mit Poedit übersetzen	569
10.7.3	Ein Theme mit Poedit übersetzen	574
10.7.4	Einen Block übersetzen	574
11	Fehler finden und beheben: Das Debugging	581
11.1	Kompatibilität	582
11.1.1	Kompatibilität zu vergangenen WordPress-Versionen	582
11.1.2	Kompatibilität zu anderen Projekten	584
11.1.3	Kompatibilität zu PHP-Versionen	585
11.1.4	Auf dem Laufenden bleiben	588

11.2	Error-Logging in WordPress	588
11.2.1	Das Error-Logging aktivieren	589
11.2.2	Fehlermeldungen in eine Datei umleiten	590
11.2.3	Bedeutung von Fehlermeldungen	593
11.3	Debugging aktivieren und Skripte analysieren	597
11.3.1	Xdebug in der lokalen Entwicklungsumgebung aktivieren	597
11.3.2	Xdebug mit Local und PhpStorm verbinden	598
11.3.3	Xdebug mit Local und Visual Studio Code verbinden	601
11.4	Ein WordPress-Plugin zum Debuggen	605
11.4.1	Datenbank-Abfragen prüfen	606
11.4.2	Variablen loggen	606
11.4.3	Konditionale Abfragen	606
11.4.4	Weitere nützliche Helfer	607
11.5	JavaScript debuggen	607

TEIL III Ihr Projekt erweitern und individualisieren

12	Schöne URLs mit der Rewrite-API	613
12.1	Warum »schöne« URLs?	613
12.2	Was ist ein Permalink in WordPress?	614
12.2.1	Konfiguration der Permalinkstruktur	614
12.2.2	Permalink definieren	615
12.3	Wie WordPress schöne URLs auflöst	616
12.3.1	Apache-Rewrite-Regeln	617
12.3.2	Nginx-Rewrite-Regeln	619
12.3.3	WordPress-Rewrite-Regeln	620
12.4	Eigene Permalink-Strukturen anlegen	624
12.4.1	Entwickeln des regulären Ausdrucks	625
12.4.2	Eine Rewrite-Regel erstellen	627
12.4.3	Eine Abfrage-Variable ergänzen	630
12.4.4	Arbeiten mit eigenen Templates	631
12.4.5	Bereits vorhandene Templates nutzen	633
12.5	Ein alternativer Weg zu neuen Permalink-Strukturen	636

13 Mit eigenen Inhaltstypen arbeiten

641

13.1 Benutzerdefinierte Artikeltypen	641
13.1.1 Einen Artikeltyp registrieren	642
13.1.2 Rewrite-Regeln bei Custom Post Types	657
13.1.3 Helferfunktionen für Custom Post Types	659
13.2 Benutzerdefinierte Taxonomien	660
13.2.1 Eine Taxonomie registrieren	661
13.2.2 Helferfunktionen für Taxonomien	668
13.3 Das Kind braucht einen Namen: Terme für Taxonomien erstellen	670
13.3.1 Einen Term erstellen und aktualisieren	670
13.3.2 Terme löschen	672
13.3.3 Term-Verbindungen	673
13.3.4 Arbeiten mit Termen im Frontend	674
13.4 Zusätzliche Daten zu Inhalten speichern mit der Metadaten-API	674
13.4.1 Die Hauptfunktionen der API	675
13.4.2 Ungeschützte Metadaten	679
13.4.3 Geschützte Metadaten	685
13.5 Mit Inhaltstypen im Frontend arbeiten	686
13.5.1 Templating im Classic-Theme	686
13.5.2 Templating im Block-Theme	690
13.5.3 Templating im Plugin	695

14 Externe Daten nachladen mit der HTTP-API

697

14.1 Was ist eine HTTP-Abfrage?	698
14.1.1 Anfrage senden	699
14.1.2 Antwort erhalten	701
14.2 HTTP-Abfragen mit WordPress	702
14.2.1 HTTP-Abfragen mit prozedurelem Code	702
14.2.2 HTTP-Abfragen mit objektorientiertem Code	708
14.2.3 Performance und Sicherheit	711
14.2.4 Dateidownloads	716

15.1 Was bedeutet »REST«?	719
15.2 Abgrenzung zu HTTP	720
15.3 Schlüsselkonzepte der REST-API	721
15.3.1 Routen und Endpunkte	722
15.3.2 Schema	724
15.3.3 Request und Response	725
15.3.4 Controller-Klassen	725
15.4 Mit Endpunkten arbeiten	725
15.4.1 REST-API-Tools	726
15.4.2 API-Discovery	727
15.4.3 Extension-Discovery	728
15.4.4 Authentifizierung	729
15.4.5 Arbeiten mit JSONP	733
15.4.6 Anfrage-Methode überschreiben	734
15.4.7 Richten Daten zurückliefern	735
15.4.8 Anfragen einsparen	736
15.4.9 Rückgabe einschränken	737
15.4.10 Paganierung	738
15.4.11 Sortierung	739
15.4.12 Links folgen	740
15.4.13 Datenübermittlung	742
15.4.14 Interne Abfragen	743
15.5 Die REST-API mit eigenen Routen erweitern	744
15.5.1 Eine Route registrieren	744
15.5.2 Endpunkte registrieren	747
15.5.3 Endpunkt-Parameter	748
15.5.4 Pfadvariablen	760
15.5.5 Die Antwort-Callback-Funktion	762
15.5.6 Das Antwort-Schema	765
15.6 Controller-Klassen nutzen	766
15.7 Mit Metadaten und REST-Feldern arbeiten	768
15.7.1 Optionen für die Seitenleiste	768
15.7.2 Meta-Felder registrieren	776
15.7.3 REST-Felder registrieren	781

16 Benutzer- und Rechtemanagement

785

16.1 Arbeiten mit Benutzern	786
16.1.1 Ein Hook in den Login-Screen	787
16.1.2 Die API vorbereiten	791
16.1.3 Der Nutzer-Login	792
16.1.4 Einen Benutzer aus der Datenbank lesen	795
16.1.5 Benutzer erstellen, aktualisieren, löschen	796
16.2 Rollenverteilung: Rollen und Benutzer-Befugnisse (Capabilities)	801
16.2.1 Befugnisse (Capabilities)	802
16.2.2 Rollen	807
16.2.3 Komplexere Befugnisse	812
16.2.4 Eigene Befugnisse erstellen und prüfen	816
16.2.5 Rechte-Checks	817
16.3 Daten zu Benutzern speichern: Die Metadata-API	819

17 Zugriff auf die Datenbank

825

17.1 Eigene Datenbank-Verbindungen	825
17.1.1 Eine Datenbank-Verbindung aufbauen	825
17.1.2 Eine Datenbank auswählen	826
17.1.3 Verbindung schließen oder neu aufbauen	827
17.2 Benutzerdefinierte Tabellen	827
17.2.1 Lieber nachfragen: Eine Datenbank-Tabelle installieren	827
17.2.2 Eine einzelne Variable lesen	832
17.2.3 Daten schreiben	834
17.2.4 Daten lesen	841
17.2.5 Daten löschen	845
17.2.6 Individuelle Abfragen	847
17.3 Die wichtigsten Datenbank-Klassen in WordPress	849
17.3.1 Mit WP_Query arbeiten	849
17.3.2 Date-Query	855
17.3.3 Tax- und Term-Query	857
17.3.4 Meta-Query	860
17.3.5 User-Query	864
17.3.6 Comment-Query	867
17.3.7 Site-(und Network-)Query	871

18 Dateien auf dem Laufwerk manipulieren

18.1 Was ist die Filesystem-API?	873
18.2 Mögliche Einsatzzwecke	873
18.3 Lokale Daten manipulieren	875
18.3.1 Avatar-Auswahl erweitern	876
18.3.2 Upload-Formular	878
18.3.3 Datei hochladen und verarbeiten	881
18.3.4 Avatar hochladen und anzeigen	888
18.3.5 Weitere Dateisystem-Funktionen	892
18.4 Entfernte Daten manipulieren	893

19 Zeitgesteuerte Aufgaben (Cronjobs)

19.1 Was ist ein Cronjob?	897
19.1.1 Pseudo-Cronjobs	897
19.1.2 Echte Cronjobs	898
19.2 Planung von Cron-Ereignissen	900
19.2.1 Einen Cronjob registrieren	900
19.2.2 Eine Callback-Funktion registrieren	902
19.2.3 Einen Cronjob löschen	903
19.2.4 Prüfen, ob ein Event bereits besteht	904
19.3 Eigene Intervalle erstellen	905
19.4 Einen Cronjob neu planen	906
19.5 Ein einzelnes Event planen	907

20 WordPress auf der Kommandozeile: Die WP-CLI

20.1 Was ist die WP-CLI und wofür braucht man sie?	911
20.2 WP-CLI installieren und konfigurieren	913
20.2.1 Die Bevorzugte installationsweise	913
20.2.2 Alternative Installationsmöglichkeiten	914
20.3 WP-CLI-Befehle auf der Kommandozeile nutzen	915
20.3.1 Ein Kommando finden und ausführen	915
20.3.2 Kommandos kombinieren	917

20.3.3	Globale Parameter	918
20.3.4	Konfigurationsdateien	919
20.4	Eigene CLI-Befehle programmieren	920
20.4.1	WP-CLI via Composer installieren	921
20.4.2	Plugin-Scaffolding	921
20.4.3	Code bei der WP-CLI-Initialisierung ausführen	922
20.4.4	CLI-Kommandos erstellen	922
20.4.5	Initiale FTP-Prüfungen	928
20.4.6	Verbindung herstellen	929
20.4.7	Medien abrufen	930
20.4.8	Dateien hochladen	932
20.4.9	Den Befehl ausführen	933
20.4.10	Weitere Informationen ausgeben	933
20.4.11	Weiterführende WP-CLI-Anwendungsmöglichkeiten	935

21 Mehrere Websites in einer Installation: Die Multisite-Umgebung

937

21.1	Eine Multisite-Installation durchführen	939
21.1.1	Multisite von Anfang an	940
21.1.2	Auf Multisite umsteigen	941
21.1.3	Die Netzwerkverwaltung	943
21.1.4	Die Theme-Verwaltung	944
21.1.5	Die Plugin-Verwaltung	945
21.1.6	Die Benutzerverwaltung	945
21.2	Datenbanken	945
21.3	Multisite-Funktionen	946
21.3.1	Auf Multisite prüfen	946
21.3.2	Websites abrufen	947
21.3.3	Aktuelle Site-ID	948
21.3.4	Site Switching	950
21.3.5	Netzwerke miteinbeziehen	951
21.3.6	Funktionen zur Verwaltung von Netzwerken und Websites	953
21.3.7	Netzwerkweite Abfragen	957
21.4	Benutzer in Multisite-Umgebungen verwalten	957
21.4.1	Benutzerprüfungen	958
21.4.2	Benutzer zur Website hinzufügen	958

22 Inhalte dynamisch einfügen mit der Shortcode-API

961

22.1 Was sind Shortcodes?	961
22.2 Eigene Shortcodes registrieren	964
22.2.1 Zusätzlicher Inhalt	966
22.2.2 Arbeiten mit Parametern	968
22.3 Shortcode-Funktionen	969

23 Alles gesund? Die Site-Health-API

971

23.1 Der Website-Status	971
23.1.1 Eigene Status-Tests	972
23.2 Website-Bericht	975
23.2.1 Eigene Felder im Website-Bericht	975

Schlusswort	979
Index	981