

## 第 1 章

## クローリング・スクレイピングとは何か

<b>1.1 本書が取り扱う領域</b>	2
1.1.1 クローリングとスクレイピング	3
1.1.2 クローリング・スクレイピングと Python	3
1.1.3 本書が対象とするプラットフォーム	4
1.1.4 本書の構成	4
<b>1.2 Wgetによるクローリング</b>	5
1.2.1 Wgetとは	5
1.2.2 Wgetの使い方	6
1.2.3 実際のサイトのクローリング	8
<b>1.3 Unixコマンドによるスクレイピング</b>	11
1.3.1 Unixコマンドの基礎知識	11
1.3.2 テキスト処理に役立つUnixコマンド	13
1.3.3 正規表現	15
<b>1.4 gihyo.jpのスクレイピング</b>	17
1.4.1 電子書籍の総数を取得する	17
column 正規表現における欲張り型のマッチ	21
1.4.2 書籍名の一覧を取得する	22
<b>1.5 まとめ</b>	24

## 第 2 章

## Pythonではじめるクローリング・スクレイピング

<b>2.1 Pythonを使うメリット</b>	26
2.1.1 言語自体の特性	26
2.1.2 強力なサードパーティライブラリの存在	27
2.1.3 スクレイピング後の処理との親和性	27
<b>2.2 Pythonのインストールと実行</b>	27
2.2.1 Python 2とPython 3	27
2.2.2 パッケージマネージャによるPython 3のインストール	28
2.2.3 仮想環境 (venv) の使用	28
2.2.4 インタラクティブシェルの使用	31
<b>2.3 Pythonの基礎知識</b>	32
2.3.1 スクリプトファイルの実行と構成	32
2.3.2 基本的なデータ構造	34
2.3.3 制御構造と関数・クラス定義	38
2.3.4 組み込み関数	41
2.3.5 モジュール	42
<b>2.4 Webページを取得する</b>	43
2.4.1 urllibによるWebページの取得	43
2.4.2 文字コードの扱い	44
<b>2.5 Webページからデータを抜き出す</b>	47
2.5.1 正規表現によるスクレイピング	47
column search() と match()	49
2.5.2 XML (RSS) のスクレイピング	50
<b>2.6 データを保存する</b>	52
2.6.1 CSV形式での保存	53
2.6.2 JSON形式での保存	56
2.6.3 データベース (SQLite 3) への保存	57
<b>2.7 Pythonによるスクレイピングの流れ</b>	59
<b>2.8 まとめ</b>	62

## 第3章

## 強力なライブラリの活用

<b>3.1</b>	<b>ライブラリのインストール</b>	64
3.1.1	pipによるインストール	64
<b>3.2</b>	<b>Webページを簡単に取得する</b>	65
<b>3.3</b>	<b>HTMLのスクレイピング</b>	67
3.3.1	XPathとCSSセレクター	68
3.3.2	lxmlによるスクレイピング	69
3.3.3	Beautiful Soupによるスクレイピング	72
	<b>column</b> pyqueryによるスクレイピング	75
<b>3.4</b>	<b>RSSのスクレイピング</b>	76
<b>3.5</b>	<b>データベースに保存する</b>	78
3.5.1	MySQLへのデータの保存	78
	<b>column</b> Python Database API 2.0	82
3.5.2	MongoDBへのデータの保存	83
<b>3.6</b>	<b>クローラーとURL</b>	87
3.6.1	URLの基礎知識	87
3.6.2	パーマリンクとリンク構造のパターン	89
3.6.3	再実行を考慮したデータの設計	91
<b>3.7</b>	<b>Pythonによるクローラーの作成</b>	92
3.7.1	一覧ページからパーマリンク一覧を抜き出す	93
3.7.2	詳細ページからスクレイピングする	95
3.7.3	詳細ページをクローリングする	98
3.7.4	スクレイピングしたデータを保存する	99
<b>3.8</b>	<b>まとめ</b>	102

## 第4章

## 実用のためのメソッド

<b>4.1</b>	<b>クローラーの分類</b>	104
4.1.1	状態を持つかどうかによる分類	104
4.1.2	JavaScriptを解釈するかどうかによる分類	106
4.1.3	不特定多数のサイトを対象とするかどうかによる分類	107
<b>4.2</b>	<b>クローラー作成にあたっての注意</b>	107
4.2.1	著作権と利用規約	107
4.2.2	robots.txtによるクローラーへの指示	109
4.2.3	XMLサイトマップ	111
4.2.4	クローリング先の負荷	113
4.2.5	連絡先の明示	114
4.2.6	ステータスコードとエラー処理	114
<b>4.3</b>	<b>繰り返しの実行を前提とした設計</b>	118
4.3.1	更新されたデータだけを取得する	118
	<b>column</b> プロキシサーバーでのキャッシュ	120
<b>4.4</b>	<b>クローリング先の変化に対応する</b>	121
4.4.1	変化を検知する	121
4.4.2	変化を通知する	123
<b>4.5</b>	<b>まとめ</b>	124

## 第5章

## クローリング・スクレイピングの実践とデータの活用

<b>5.1</b>	<b>データセットの取得と活用</b>	126
5.1.1	Wikipediaのデータセットのダウンロード	126
5.1.2	自然言語処理技術を用いた頻出単語の抽出	130
<b>5.2</b>	<b>APIによるデータの収集と活用</b>	135
5.2.1	Twitterからのデータの収集	136

5.2.2	Amazonの商品情報の収集	143
5.2.3	YouTubeからの動画情報の収集	145
<b>5.3</b>	<b>時系列データの収集と活用</b>	<b>153</b>
5.3.1	為替などの時系列データの収集	153
5.3.2	CSV/Excelファイルの読み込み	157
5.3.3	グラフによる可視化	166
	<b>column</b> pandasとmatplotlib	172
	<b>column</b> 科学技術計算やデータ分析のための便利なツール: IPython・Jupyter・Anaconda	172
<b>5.4</b>	<b>オープンデータの収集と活用</b>	<b>173</b>
5.4.1	オープンデータとは	173
5.4.2	PDFからのデータの抽出	174
5.4.3	Linked Open Dataからのデータの収集	179
	<b>column</b> オープンデータとシビックテック	185
<b>5.5</b>	<b>Webページの自動操作</b>	<b>186</b>
5.5.1	自動操作の実現方法	186
5.5.2	Amazon.co.jpの注文履歴を取得する	188
<b>5.6</b>	<b>JavaScriptを使ったページのスクレイピング</b>	<b>191</b>
5.6.1	JavaScriptを使ったページへの対応方法	191
5.6.2	noteのおすすめコンテンツを取得する	194
5.6.3	RSSフィードを生成する	202
<b>5.7</b>	<b>取得したデータの活用</b>	<b>205</b>
5.7.1	地図による可視化	205
	<b>column</b> JSONに対してクエリを実行するjqコマンド	206
5.7.2	BigQueryによる解析	215
<b>5.8</b>	<b>まとめ</b>	<b>223</b>

## 第6章

# フレームワーク Scrapy

<b>6.1</b>	<b>Scrapyの概要</b>	<b>226</b>
6.1.1	Scrapyのインストール	227
6.1.2	Spiderの実行	227
<b>6.2</b>	<b>Spiderの作成と実行</b>	<b>229</b>
6.2.1	Scrapyプロジェクトの開始	230
6.2.2	Itemの作成	231
6.2.3	Spiderの作成	232
6.2.4	Scrapy Shellによるインタラクティブなスクレイピング	236
	<b>column</b> ScrapyのスクレイピングAPIの特徴	242
6.2.5	作成したSpiderの実行	243
	<b>column</b> FTPサーバーやAmazon S3などにデータを保存する	247
<b>6.3</b>	<b>実践的なクローリング</b>	<b>248</b>
6.3.1	クローリングでリンクをたどる	248
6.3.2	XMLサイトマップを使ったクローリング	250
<b>6.4</b>	<b>抜き出したデータの処理</b>	<b>254</b>
6.4.1	Item Pipelineの概要	254
6.4.2	データの検証	255
6.4.3	MongoDBへのデータの保存	256
6.4.4	MySQLへのデータの保存	258
<b>6.5</b>	<b>Scrapyの設定</b>	<b>260</b>
6.5.1	設定の方法	260
6.5.2	クローリング先に迷惑をかけないための設定項目	261
6.5.3	並行処理に関する設定項目	262
6.5.4	HTTPリクエストに関する設定項目	263
6.5.5	HTTPキャッシュの設定項目	264
6.5.6	エラー処理に関する設定	265
6.5.7	プロキシを使用する	266
<b>6.6</b>	<b>Scrapyの拡張</b>	<b>267</b>
6.6.1	ダウンロード処理を拡張する	267
6.6.2	Spiderの挙動を拡張する	269
	<b>column</b> ScrapyでJavaScriptを使ったページに対応する: Splash	270

<b>6.7 クローリングによるデータの収集と活用</b>	271
6.7.1 レストラン情報の収集	271
6.7.2 不特定多数のWebサイトのクローリング	275
6.7.3 Elasticsearchによる全文検索	281
<b>6.8 画像の収集と活用</b>	291
6.8.1 Flickrからの画像の収集	292
6.8.2 OpenCVによる顔画像の抽出	297
column UbuntuでのOpenCV 3のビルド	303
<b>6.9 まとめ</b>	304

## 第7章

### クローラーの継続的な運用・管理

<b>7.1 クローラーをサーバーで動かす</b>	306
7.1.1 仮想サーバーの立ち上げ	307
column Windowsから公開鍵認証を使ってSSH接続する	315
7.1.2 サーバーへのデプロイ	316
column AWS利用におけるセキュリティの注意点	316
<b>7.2 クローラーの定期的な実行</b>	319
7.2.1 Cronの設定	319
column Windowsでサーバーにファイルを転送する	320
7.2.2 エラーの通知	322
<b>7.3 クローリングとスクレイピングの分離</b>	325
7.3.1 メッセージキューRQの使い方	326
column Redisのデータの永続化に関する設定	327
7.3.2 メッセージキューによる連携	330
column ScrapyからRQにジョブを投入する	334
7.3.3 メッセージキューの運用	336
<b>7.4 クローリングの高速化・非同期化</b>	338
7.4.1 マルチスレッド化・マルチプロセス化	338
7.4.2 非同期I/Oを使った効率的なクローリング	342
column Python 3.4でasyncioを使う	346
column 複数のマシンによる分散クローリング	348

<b>7.5 クラウドを活用する</b>	349
7.5.1 クラウドを使うメリット	349
7.5.2 AWSのSDKを使う	350
7.5.3 クラウドストレージを使う	351
<b>7.6 まとめ</b>	355
column 外部サービスを活用したスクレイピング	356

## Appendix

### Vagrantによる開発環境の構築

<b>A.1 VirtualBoxとVagrant</b>	360
A.1.1 VirtualBoxとは	360
A.1.2 Vagrantとは	361
<b>A.2 CPUの仮想化支援機能を有効にする</b>	361
A.2.1 Windows 10の場合	361
A.2.2 Windows 7の場合	362
A.2.3 ファームウェアの設定で仮想化支援機能を有効にする	363
<b>A.3 VirtualBoxのインストール</b>	363
<b>A.4 Vagrantのインストール</b>	364
<b>A.5 仮想マシンを起動する</b>	365
<b>A.6 ゲストOSにSSH接続する</b>	368
A.6.1 Tera Termのインストール	369
A.6.2 Tera TermでゲストOSにSSH接続する	369
<b>A.7 Linuxの基本操作</b>	371
A.7.1 ソフトウェアをインストールする	373
<b>A.8 Vagrantで仮想マシンを操作するコマンド</b>	373
A.8.1 仮想マシンを起動する (vagrant up)	374
A.8.2 仮想マシンを終了・再起動する (vagrant halt/reload)	374
A.8.3 仮想マシンを削除する (vagrant destroy)	374
A.8.4 仮想マシンの状態を表示する (vagrant status)	374
A.8.5 仮想マシンにSSH接続する (vagrant ssh)	375

<b>A.8.6</b> 仮想マシンをエクスポートする (vagrant package) .....	375
おわりに .....	376
参考文献 .....	377
索引 .....	378