14

FINAL CHAPTER

次のステップと学習パス

おめでとう — ここがスタート地点。これから何を、どの順で学ぶか

Happy Coding — 作り続けることで、上達する

AGENDA

この章で学ぶこと

01 学んだことの振り返り
本書で習得した6つの技術領域を棚卸し

02 さらに学ぶべきこと
TypeScript・DRF・Docker・K8s・Terraform ほか

03 次のプロジェクトアイデア
ゲーム・実用ツール・SaaSの6案

04 コミュニティ参加
GitHub公開・ブログ執筆・カンファレンス・OSS

05 キャリアパス
Webエンジニア・フリーランス・起業・AIエンジニア

PART 1

学んだことの振り返り

ゼロからデプロイまで ― 6つの技術領域

RECAP 1 / 3

環境構築 と Django の基礎

「自力で作って、データを動かす」までの土台

環境構築

学んだこと

- Python・Git・エディタのインストール
- Gemini API（Google AI Studio）
- ターミナル/CLIの基礎
- PATH設定・権限・ポート競合の解決

→ 開発環境を自力でセットアップできる

Djangoの基礎

学んだこと

- django-admin startproject
- MTV アーキテクチャ
- モデル設計とマイグレーション
- URL/View/Template + CSRF対策

→ DB駆動のWebアプリを作れる

RECAP 2 / 3

LLM API活用 と フロントエンド

AIをアプリに組み込む / 動的なUIを構築する

LLM API活用

学んだこと

- Gemini APIの統合
- プロンプトエンジニアリング
- エラーハンドリングとリトライ
- 環境変数で機密情報管理

→ LLMを自分のアプリに統合できる

フロントエンド

学んだこと

- htmx — JavaScriptなしでAjax
- Tailwind CSS でスタイリング
- レスポンシブデザインの基礎
- サーバー中心の動的UI構築

→ モダンなUIを構築できる

RECAP 3 / 3

AWS EC2デプロイ と 運用

本番環境で「動かし続ける」スキル

AWS EC2 デプロイ

学んだこと

- AWSアカウント / EC2 / SSH
- Gunicorn + Nginx 構成
- Systemd でプロセス管理
- Let's Encrypt で HTTPS化
- Route 53 で DNS設定

→ 本番環境にデプロイできる

運用とトラブル対応

学んだこと

- journalctl / Nginx ログ確認
- 502 / 500 / メモリ不足の対処
- htop / df / iftop で監視
- cron + S3 でバックアップ
- SSH / Fail2ban / UFW でセキュリティ

→ 本番のトラブルに対処できる

PART 2

さらに学ぶべきこと

本書の知識を「土台」に積み上げる技術スタック

LEARN — FRONTEND

フロントエンド強化 — TypeScript + UIフレームワーク

htmxの「サーバー中心」から、リッチな状態管理へ

TypeScript

JavaScriptに型を追加

01 バグの早期発見

コンパイル時に型エラーを検出

02 IDE補完が強化

自動補完・リファクタリングが快適

03 大規模で必須

React/Vue/Angularでも標準

React vs Vue.js

状況に応じて選ぶ — どちらも本書の延長で十分習得可能

R React

Meta製・コンポーネントベース

向いている人

求人数最多 — 就職/転職を狙うなら

V Vue.js

学習コスト低・日本語情報豊富

向いている人

個人開発・学習優先 / htmxの延長として

LEARN — BACKEND

バックエンド強化 — API・非同期・リアルタイム

DRF / Celery / Channels で「もう一歩」先へ

1

Django REST Framework

Django でRESTful APIを簡単構築

なぜ学ぶべきか

シリアライズ・認証・ページネーション
フロントとバックの分離、モバイル連携

2

Celery (非同期タスク)

重い処理をバックグラウンドで

なぜ学ぶべきか

メール送信・画像処理・定期実行
スケーラブルなアプリの必須要素

3

Django Channels (WebSocket)

リアルタイム通信を実装

なぜ学ぶべきか

チャット・ライブ通知・マルチプレイヤー
双方向通信が必要な機能で活躍

LEARN — INFRA

インフラ強化 — 自動化・コンテナ・IaC

手動デプロイから「コードで管理する」インフラへ

デプロイの自動化

GitHub Actions (CI/CD)

PRに紐づき、テスト→自動デプロイ

Ansible (構成管理)

サーバーのあるべき状態をコード化

rsync / Git pull

小規模なら手軽な選択肢でも十分

Docker (コンテナ)

「環境構築の悩み」から解放される

Kubernetes (K8s)

自動スケール・自動復旧の標準

Terraform (IaC)

AWSリソース全体をコードで管理

クラウド選択肢

主要クラウド

AWS	サービス数最多・エコシステム充実
Azure	企業向け・Windows統合に強い
GCP	ML/データ分析・料金が分かりやすい

安価・手軽系（個人開発向け）

- DigitalOcean — シンプル・ドキュ充実
- さくらのクラウド — 国内/日本語サポ
- Vultr / ConoHa — 低価格・初心者向け

選定の観点: 運用難易度 / 総コスト / デプロイのしやすさ / 将来性

LEARN — DATA / OBSERVABILITY

データベース と 監視・可観測性

PostgreSQL + Redis、 Prometheus + Grafana

DB

PostgreSQL

オープンソースの高機能RDB

なぜ学ぶべきか

SQLiteより高性能・マルチユーザー対応
JSON・全文検索・地理情報など高度機能

KV

Redis

インメモリKVS

なぜ学ぶべきか

キャッシュ・セッション・メッセージキュー
パフォーマンス改善・Celeryブローカー

OBS

Prometheus + Grafana

メトリクス収集+可視化

なぜ学ぶべきか

システム健全性をダッシュボード化
障害の早期発見・SLO監視

LEARNING ROADMAP

推奨学習順序 — 4フェーズ・15ヶ月

フロント → バック → インフラ → 高度な技術

PHASE 1

フロントエンド強化

3 ヶ月

学習内容

TypeScript → React or Vue.js

PHASE 2

バックエンド強化

3 ヶ月

学習内容

DRF → Celery → PostgreSQL

PHASE 3

インフラ強化

3 ヶ月

学習内容

Docker → GitHub Actions → Ansible

PHASE 4

高度な技術

6 ヶ月

学習内容

Kubernetes → Terraform → Prometheus + Grafana

完璧主義にならない — 各フェーズは「使えるレベル」で次へ進む

PART 3

次のプロジェクトアイデア

ゲーム・実用ツール・SaaS — 6つの題材

PROJECT IDEAS

実践で磨く 6 つのアイデア

難易度別 / 推奨フェーズ別に並べた次のプロジェクト

ゲーム PHASE 3+

マルチプレイヤーRPG

★★★★☆

技術スタック

Channels + Redis + PostgreSQL

ゲーム PHASE 3-4

AIダンジョンマスター

★★★★☆

技術スタック

Gemini + DALL-E + React

実用 PHASE 2-3

AI議事録ツール

★★★☆☆

技術スタック

Whisper + Gemini + Celery

実用 PHASE 2

コード解説ボット

★★★☆☆

技術スタック

GitHub API + Gemini + DRF

SaaS PHASE 4+

ノーコードLLMチャットボット

★★★★★

技術スタック

DRF + React + Stripe + K8s

SaaS PHASE 3-4

AIコンテンツ生成プラットフォーム

★★★★☆

技術スタック

DRF + Gemini + DALL-E + S3

PART 4

コミュニティと キャリアパス

学んだことを共有し、進む方向を描く

COMMUNITY

コミュニティ参加 — 4つの動き方

学んだことを共有し、他の開発者と交流する

01 GitHubで公開

- README + ライセンス整備
- リポジトリを Public へ
- デモURL も明記

02 ブログ執筆

- Qiita / Zenn / Dev.to / Medium
- 「躓きと解決」が読まれる
- 導入→技術選定→結果→学び

03 カンファレンス登壇

- PyCon JP / DjangoCongress
- 5分の LT から始める
- ローカル勉強会も入口◎

04 OSSへの貢献

- まず誤字修正・翻訳から
- 「good first issue」を探す
- 新機能提案 → PR

CAREER PATHS

代表的な4つのキャリア

年収は東京の目安。地域・規模で大きく変わる

Webエンジニア

Webアプリ開発・運用

求められるスキル

本書 + React/Vue + Docker + PostgreSQL

年収/特徴(目安)

- 未経験〜2年: 300〜450万
- 3〜5年: 500〜700万
- 5年以上: 700〜1,000万+

フリーランス

案件受注・リモート主流

求められるスキル

技術力 + コミュ力 + 自己管理

年収/特徴(目安)

- 月単価 60〜100万
- （年収 720〜1,200万）
- 案件の取り方が肝

スタートアップ起業

自分のプロダクトを作る

求められるスキル

技術力 + ビジネススキル

年収/特徴(目安)

- 収入は不安定
- 失敗確率は高い
- 成功時のリターンは大

AIエンジニア

LLM・MLモデルの開発/運用

求められるスキル

Python + ML + 統計/数学

年収/特徴(目安)

- 年収 650〜1,350万+
- TensorFlow / PyTorch
- データ・モデル両方

※地域・企業規模・スキルレベルで大きく異なります

FINAL WORDS

Happy Coding.

本書を最後まで読んでくださり、ありがとうございました

作り続ける

アイデアを思いついたら、すぐに作り始める

諦めない

エラーが出ても、LLMに相談しながら解決する

発信する

GitHubで公開し、ブログで学びを言語化する

交流する

コミュニティに参加し、他の開発者と繋がる

プログラミングは、自分のアイデアを形にする魔法

あなたが作るアプリが、誰かの役に立ちますように。