Cline研修_受講ガイド

📖 はじめに

このガイドは、Clineを使用した実践的な研修プログラムの受講生向けマニュアルである。

前提条件:

- 研修講師により環境セットアップは完了済み
- Payara Server 6とHSQLDB Databaseが起動済み
- データソース（jdbc/HsqldbDS）が作成済み
- 初期データがセットアップ済み
- berry-books-1とberry-books-restがデプロイ済み

📁 プロジェクト構成

研修では以下のプロジェクトを使用する：

```
cline_training_work/
├── projects/
│   ├── java/
│   │   ├── berry-books-1/          # レッスン1〜5: Jakarta EE基礎【研修用】
│   │   │   ├── src/                # Javaソースコード
│   │   │   ├── spec/               # 詳細設計書、アーキテクチャガイドライン
│   │   │   └── prompts/            # レッスン1〜5のプロンプト
│   │   │       ├── lesson_1_minor_improvement/
│   │   │       ├── lesson_2_check_guideline/
│   │   │       ├── lesson_3_potential_bag/
│   │   │       ├── lesson_4_enhance/
│   │   │       └── lesson_5_unittest/
│   │   │
│   │   ├── berry-books-2/          # レッスン6: リファクタリング【研修用】
│   │   │   ├── src/                # Javaソースコード
│   │   │   ├── spec/               # アーキテクチャガイドライン
│   │   │   └── prompts/
│   │   │       └── lesson_6_refactoring/
│   │   │
│   │   ├── berry-books-rest/       # REST API【完成版】
│   │   │   ├── src/                # Javaソースコード
│   │   │   └── spec/               # OpenAPI仕様書
│   │   │
│   │   └── struts-to-jsf-person/   # レッスン8: FWマイグレーション【研修用】
│   │       ├── src/                # Javaソースコード（Struts 1.3）
│   │       ├── spec/               # 移行仕様書
│   │       └── prompts/
│   │           └── lesson_8_migration/
│   │
│   ├── react/
│   │   └── berry-books-frontend/   # レッスン7: React SPA【研修用】
│   │       ├── src/                # React + TypeScriptソースコード
│   │       └── prompts/
│   │           └── lesson_7_spec_driven/
│   │
│   └── python/
│       └── accounting_glue/        # レッスン9: ETLスクラッチ開発【研修用】
│           ├── src/                # Pythonソースコード
│           ├── spec/               # 要件定義書、基本設計書、詳細設計書
│           ├── ddl/                # データ定義（入出力仕様）
│           └── prompts/
│               └── lesson_9_etl_scratch/
```

プロジェクトの役割

berry-books-1	Jakarta EE 10 JSF + CDI + JPA	小規模改善、ガイドライン準拠、不具合検出、機能拡張、単体テスト
berry-books-2	Jakarta EE 10 JSF + CDI + JPA	大規模リファクタリング（Service/DAO分離）
struts-to-jsf-person	Struts 1.3 → Jakarta EE 10 JSF + CDI + JPA	レガシーシステムのモダナイゼーション
accounting_glue	Python / PySpark / Glue	ETL処理の実装（データ統合・変換）

🎓 学習手順

基本的な進め方

各レッスンは以下の流れで進める：

1. コンテキストの追加

Clineのチャットパネルに、対象プロジェクトのソースフォルダを追加する。

操作手順:

1. VS Codeのエクスプローラーで各プロジェクトのルートを選択
2. SHIFT + ドラッグ＆ドロップで、Clineチャットパネルにフォルダをドロップ
3. コンテキストとして追加されたことを確認

例:

- レッスン1〜5: projects/java/berry-books-1/ を追加
- レッスン6: projects/java/berry-books-2/ を追加
- レッスン7: projects/react/berry-books-frontend/ を追加
- レッスン8: projects/java/struts-to-jsf-person/ を追加
- レッスン9: projects/python/accounting_glue/ を追加

2. プロンプトの選択

各レッスンには3つのレベルのプロンプトが用意されている：

Simple	prompt_1_simple.txt	**最も抽象的** 要求のみ記述、実装詳細なし	生成AIの能力を試したいケース
Just	prompt_2_just.txt	**バランス型** 適度な情報量	標準的な進め方
Much	prompt_3_much.txt	**最も具体的** 詳細な指示、参照資料明示	確実に成果を出したいケース

🎓 学習手順

3. Clineの実行モード選択

Clineには2つの実行モードがある：

A. Planモード（推奨）

1. プロンプトを投入
2. ClineがPlanモードで実行計画を提示
3. 計画を確認し、問題なければActモードで実行を指示
4. コード生成・修正が実行される

利点:

- 生成AIの理解度を事前確認できる
- 誤った方向性を早期に修正できる
- 学習効果が高い

B. Actモード（直接実行）

1. プロンプトを投入
2. Clineがいきなりコード生成・修正を実行

利点:

- 素早く結果が得られる
- シンプルな作業に適している

4. 成果物の確認とテスト

Clineが作成したコードを確認し、動作をテストする。

確認ポイント:

- ビルドエラーがないか
- 期待通りの動作をするか
- コード品質（可読性、保守性）

デプロイとテスト:

```
# ビルド
./gradlew :projects:java:berry-books-1:war

# デプロイ
./gradlew :projects:java:berry-books-1:deploy

# ブラウザでアクセス
http://localhost:8080/berry-books-1
```

各レッスンの進め方

レッスン1: 小規模改善【berry-books-1】

目的: 既存コードの品質改善（定数管理、メッセージ管理、共通化）

プロンプト格納先:

```
projects/java/berry-books-1/prompts/lesson_1_minor_improvement/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

01

コンテキスト追加

projects/java/berry-books-1/src/ をClineチャットパネルに追加する

02

プロンプト選択

lesson_1_minor_improvement/ から適切なプロンプトファイルを選択する

03

実行モード選択

Planモード（推奨）またはActモードでClineを実行する

04

成果物確認

定数管理、メッセージ管理、共通化が適切に実装されているか確認する

レッスン2: ガイドライン準拠チェック【berry-books-1】

目的: アーキテクチャガイドラインへの準拠状況をAIに診断させる

プロンプト格納先:

```
projects/java/berry-books-1/prompts/lesson_2_check_guideline/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

診断対象項目

- アーキテクチャ構造の妥当性
- コーディング規約の遵守状況
- セキュリティベストプラクティス
- パフォーマンス最適化の余地
- エラーハンドリングの適切性

期待される成果

- ガイドライン違反箇所の特定
- 改善提案の具体的な内容
- 優先度付けされた修正リスト
- ベストプラクティスへの適合度評価

レッスン3: 潜在的不具合検出【berry-books-1】

目的: 生成AIによる解析で潜在的な不具合を検出

プロンプト格納先:

```
projects/java/berry-books-1/prompts/lesson_3_potential_bag/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

Null参照の危険性

NullPointerExceptionが発生する可能性のある箇所を特定する

リソースリーク

データベース接続やファイルハンドルの適切なクローズ処理を確認する

並行処理の問題

スレッドセーフでない実装や競合状態の可能性を検出する

セキュリティ脆弱性

SQLインジェクションやXSSなどの脆弱性を洗い出す

レッスン4: 機能拡張【berry-books-1】

目的: 詳細設計書からの機能拡張（在庫チェック機能の追加）

プロンプト格納先:

```
projects/java/berry-books-1/prompts/lesson_4_enhance/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

実装する機能

書籍の在庫管理機能を既存システムに追加する。

設計書の理解

詳細設計書から要件を正確に把握する

既存コードの分析

影響範囲を特定し、修正箇所を洗い出す

実装とテスト

機能を実装し、動作確認を行う

レッスン5: 単体テスト作成【berry-books-1】

目的: JUnitテストコードの自動生成

プロンプト格納先:

```
projects/java/berry-books-1/prompts/lesson_5_unittest/
 ├── prompt_1_simple.txt
 ├── prompt_2_just.txt
 ├── prompt_3_much.txt
 └── prompt_4_bad.txt（失敗例）
```

テスト作成の観点

1. **正常系テスト**: 期待通りの入力で正しく動作するか
2. **異常系テスト**: 不正な入力やエラー条件での挙動
3. **境界値テスト**: 境界条件での動作確認
4. **カバレッジ**: コードの実行経路を網羅的にテスト

テストの品質基準

- テストケースの独立性
- アサーションの明確性
- テストデータの適切性
- モックの適切な使用

prompt_4_bad.txtについて

このプロンプトは、プロダクションコードに不具合があるにも関わらず、テストコードがその影響を受けてしまい、不具合を検出できない、というバッドプラクティスの例。

レッスン6: 大規模リファクタリング【berry-books-2】

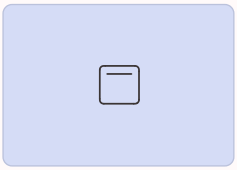
目的: アーキテクチャの再設計（Service/DAO分離）

プロンプト格納先:

```
projects/java/berry-books-2/prompts/lesson_6_refactoring/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

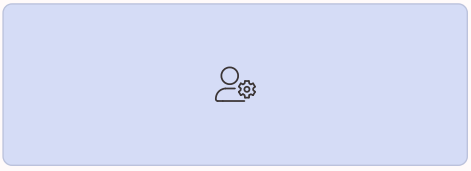
リファクタリングの方針

既存ではビジネスロジック層においてデータアクセスを行っているが、それを3層からなる**レイヤードアーキテクチャ**に再構成する。具体的には以下の層に分離する：



プレゼンテーション層

UI（JSF）



ビジネスロジック層

Service（CDI Bean）

データアクセス層

DAO（JPA）

レッスン7: スペック駆動開発【berry-books-frontend】

目的: OpenAPI仕様書からのReact SPA実装

プロンプト格納先:

```
projects/react/berry-books-frontend/prompts/lesson_7_spec_driven/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

概要

Berry Books オンライン書店の管理者画面。React + TypeScriptによるSPA（Single Page Application）として実装。`berry-books-rest` APIと連携して顧客管理機能を提供。

採用技術

- **フロントエンド**: React 18 + TypeScript 5
- **ビルドツール**: Vite 5
- **バックエンドAPI**: `berry-books-rest`

主な機能

- 顧客一覧表示（テーブル形式）
- 顧客情報の編集（ダイアログ形式）

システム構成

```
[Webブラウザ] ⇄ [Vite Dev Server] ⇄ [Payara Server]
              http://localhost:3000  (berry-books-rest.war)
                └── React SPA          └── REST API
```

レッスン8: FWマイグレーション【struts-to-jsf-person】

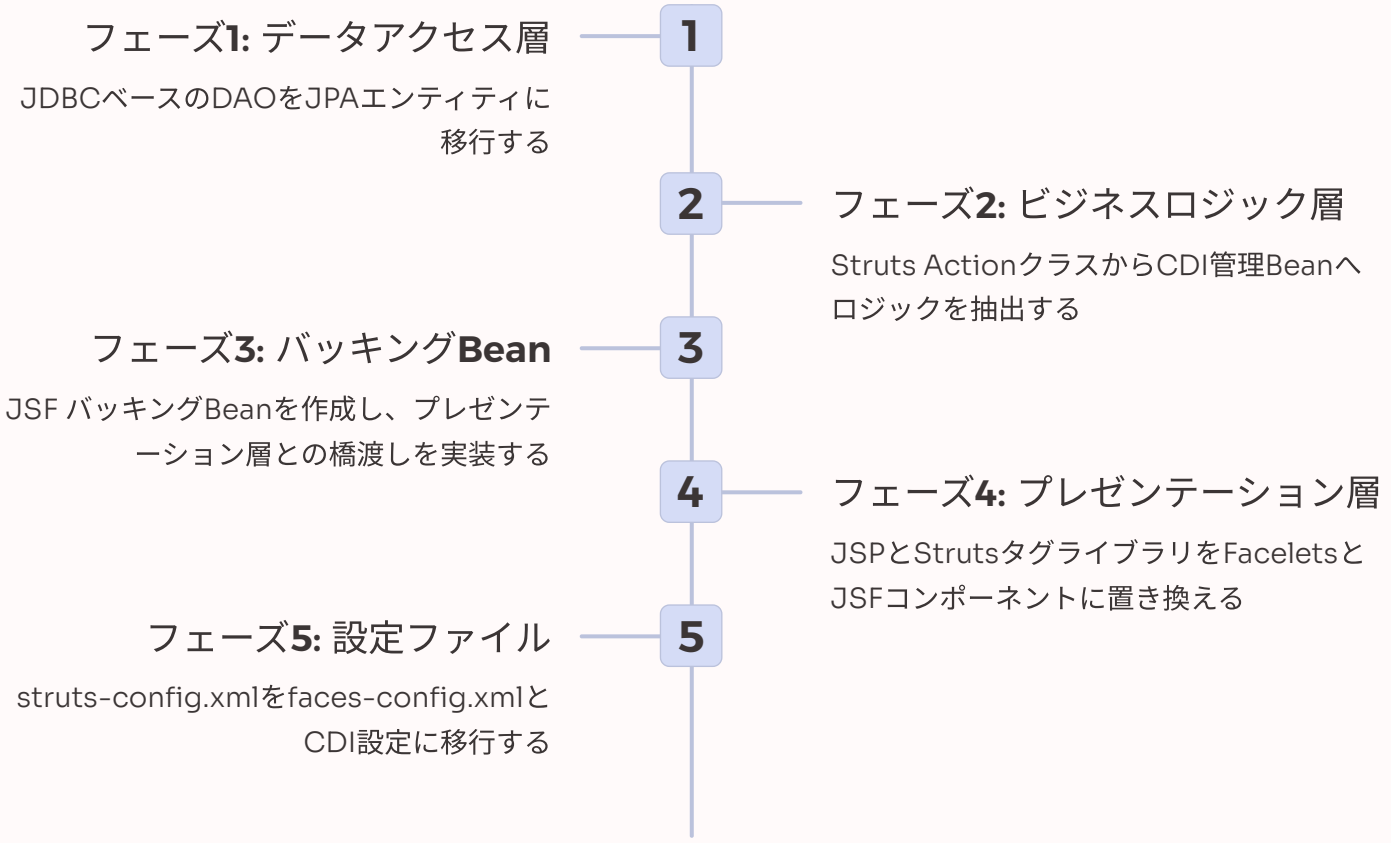
目的: レガシーシステム（Struts 1.3）のモダナイゼーション

プロンプト格納先:

```
projects/java/struts-to-jsf-person/prompts/lesson_8_migration/
├── prompt_1_simple.txt
├── prompt_2_just.txt
├── prompt_3_much_1-jpa.txt（データアクセス層）
├── prompt_3_much_2-cdi.txt（ビジネスロジック層）
├── prompt_3_much_3-jsf.txt（バッキングBean）
├── prompt_3_much_4-facelets.txt（プレゼンテーション層）
└── prompt_3_much_5-config.txt（設定ファイル）
```

マイグレーション戦略

Struts 1.3からJakarta EE 10への移行は、**段階的なアプローチ**で実施する。prompt_3_muchシリーズは、各層ごとに分割されたプロンプトを提供しており、以下の順序で移行を進める：



レッスン9: ETLスクラッチ開発【accounting_glue】

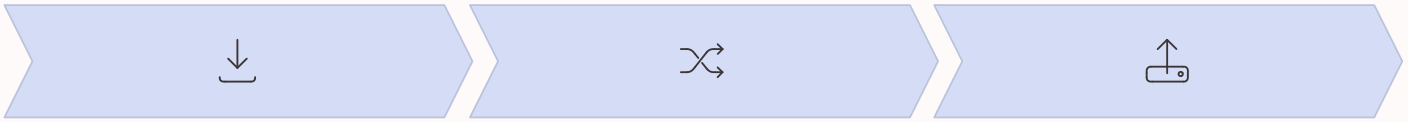
目的: Python標準/PySpark/Glueによるデータ統合ETL処理のスクラッチ開発

プロンプト格納先:

```
projects/python/accounting_glue/prompts/lesson_9_etl_scratch/
├── prompt_1_simple.txt
├── prompt_2_just.txt
└── prompt_3_much.txt
```

ETL処理の概要

複数の会計システムからデータを抽出し、統合データウェアハウスに格納するETLパイプラインを実装する。



Extract（抽出）

複数のソースシステムからデータを取得する

Transform（変換）

データクレンジング、正規化、集計処理を実行する

Load（格納）

統合データを目的のデータストアに書き込む

実装要件

技術要件

- Python標準/PySpark/Glueを使用
- 大量データの効率的な処理
- エラーハンドリングとログ出力
- データ品質チェック機能

設計書の活用

- 要件定義書: ビジネス要件の理解
- 基本設計書: システム構成の把握
- 詳細設計書: 処理ロジックの実装
- DDL: 入出力データ構造の確認