Application
Architecture
Bootcamp

4-Week Intensive

短期集中アプリケーションアーキテクチャ習得講座

1.バリデーション
2.Clean Architecture
3.トランザクション　& 排他制御
4.認証・認可とセキュリティ

生成AIの時代,超アジャイルに短期決戦するための　Bootcamp!

生成AIを活用して開発するのがあたりまえになりつつあります。

我々自身も、Claude Codeを利用して大幅に開発効率を向上させており、

人力で開発した場合の見積もりの1/3～1/6程度の時間に圧縮することに成功しています。

そこで大いに役立ったのが、これまで培ってきた、アプリケーションアーキテクチャに関する理解です。

この短期集中講座は、中規模～大規模アプリケーションに十分に適用可能な

アプリケーションアーキテクチャの基礎を身につけていただくものです。

ぜひ、本講座を通して、”アプリケーションアーキテクチャの型”を自分のものにして、

生成AI時代の超アジャイルな開発スタイルに向きあえるようになってください。

対象となる方

システム開発プロジェクトへに従事した経験：1年以上

SIer/事業会社のDX部門・情報システム部門　に所属する正社員の方

年齢層：20代 – 30代前半を想定

特にこんな方にオススメ！

開発プロジェクトに従事するも、実際のコーディングなどは委託先企業に委ねる事が多く、

もどかしい想いをいただいている方。

講座の進め方と価格

対面講義 ＋ 解説スライドに書かれた課題への取り組み ＋ Slackでの質疑応答を行います。

対面講義：1日1回、1時間程度、1 on 1方式

講義期間：4週間を想定
※ただし、1単位ごとで切り出し可能。標準講座の場合は3ヶ月間程度の完了猶予を設ける事も可能です。

Slackによる質疑応答：質問回数などの制限はありません。講義期間中はいつでも構いません。
（深夜や休日であっても可能な限り迅速にお答えします）

具体的な講義メニューについては、後続のスライドをご確認ください。

標準価格：32万円（税抜き）

1.バリデーション
2.Clean Architecture
3.トランザクション ＆ 排他制御
4.認証・認可とセキュリティ

オプションorお試し価格：8万円（税抜き）

左記に示す4つの講座単位のうち、
1単位ごとの販売に対応できます。
ただし、バラ売り価格の場合、
受講期限が平日5日以内での完了となります。

開始までのプロセス

参加企業さまの本講座購入責任者の方と弊社とでお打ち合わせ

参加企業様の教育目標や課題感と、弊社が提供できることについて打ち合わせさせていただきます。

受講される方との顔合わせと受講スケジュールの調整

「今忙しいんだけど、数カ月後にちょうど案件の切れ目・・」などのご要望にも柔軟に対応いたします。

受講メニューの確定

受講メニューの確定までは無料で、キャンセルも全く問題ありません。

受講開始

講師プロフィール

株式会社 Smart World Architect　紺野 祐介

1998年よりシステム開発に従事。

当時まだ目新しかった、Javaによる3階層クライアント/サーバ方式のフレームワークによる
SAP対抗馬の国産ERP開発に参画。フレームワーク、企業システムの基本などをみっちりと学ぶ。

以後フリーランスに転向しいくつかのプロジェクトに従事の後、2010年より自分の専門領域を”ITアーキテクト”
に絞り、コンサル系の中小企業に所属。

業務委託の立場でありながら、大手SIにて当時で20億円規模の大型開発にチーフアーキテクトとして参画。

100名超のオフショア開発において、クライアント（当時はjQuery）、サーバ（当時はStrtus）

フルスタックの開発技法、標準化などのすべての領域を網羅。

2018年末に株式会社SmartWorldArchitectを起業。

起業後は、自社サービスやSaaS組織のチームビルディングや開発プロセス改善に携わる。

また、それらのノウハウを2020年春より開発会社や大手SI向けに教育コンテンツとして展開。

開発会社向けの講義実施回数は5年間、通算250回を超えた。

2025年秋より、より講義方式を洗練し、短期集中型講座AABを開始。

講義メニュー：バリデーション

GOAL

プロジェクト性質と要求に沿ったバリデーション方式の選択と提案ができるようになる

クエスト：

・バリデーションのないアプリケーションで、どんなことがおきるか想像できますか？
・バリデーションの仕組みを作ってください、と言われたら作れそうですか？
・Bean Validationフレームワークの利用イメージがつきますか？
・メソッドチェーンで書かれた独自バリデーションフレームワークに触れて、Bean Validationとの違いを評価してください。

・サーバサイド処理の全体像を思い描けますか？　/ バリデーションはどの位置づけでしょう？
・バリデーション内の三つのフェーズとは何でしょう？

・バリデーション処理はどのように肥大化するか想像できますか？
・その肥大化はどのクラス、レイヤに影響するでしょう？
・有効な肥大化対策にはどんなパターンがあるでしょう？

クエストに答えていくことによって
最終的にGOALに到達できるようにします。
また、クエストの回答内容によって
講義の内容を調整します。

講義メニュー：Clean Architecture

GOAL

レイヤードアーキテクチャとどう違うのか？をまずは体感する。
「変更のしやすさ」への手段がClean Architectureなのだ、という実感を得て、提案の武器にする

クエスト：

・自分でパッケージの役割を整理しながら、Layered ArchitectureとClean Architectureを比較してみましょう。
・サービス→サービス呼び出しの違和感、感じたことがありますか？
・「依存性逆転の法則」を自分の言葉で再定義してみましょう。上位モジュールと下位モジュールの意味とは？

・変更のしやすさ　がいかに大切かを実感してみましょう
・なにが、ビジネスルール　で、なにが　アプリケーション（ユースケース）か、実際に取り組んでみましょう

講義メニュー:トランザクション　& 排他制御

GOAL

トランザクションは◯◯い方が良い。この鉄則を叩き込む。&
楽観排他と悲観排他という表現に惑わされることなく、課題に適した排他制御方式を採用できるようになる

クエスト:

・トランザクションって何？って聞かれたら、どう答えますか？
・ACID特性のうち、トランザクション設計と強く関連してくるものってなんでしょう？
・トランザクションはできるだけ〇〇い方が良い。〇〇に当てはまる言葉とは？

・楽観排他、悲観排他、あまり”主観”にとらわれないほうが良いかも。
・排他制御はリスクコントロール。verNoによるupdate　と　select for updateを適切に使い分けましょう。
・updateによる他のトランザクション待ち時間を計算してみましょう。

・@Transactionalの制約と効果的な利用、暗黙のトランザクションを避ける

講義メニュー：認証・認可とセキュリティ

GOAL

認証認可の基本と、それにまつわるセキュリティの考え方を押さえる　。&
SpringSecurityフレームワークを理解することで自ら認証認可の方式設計が実施できるようになる

クエスト：

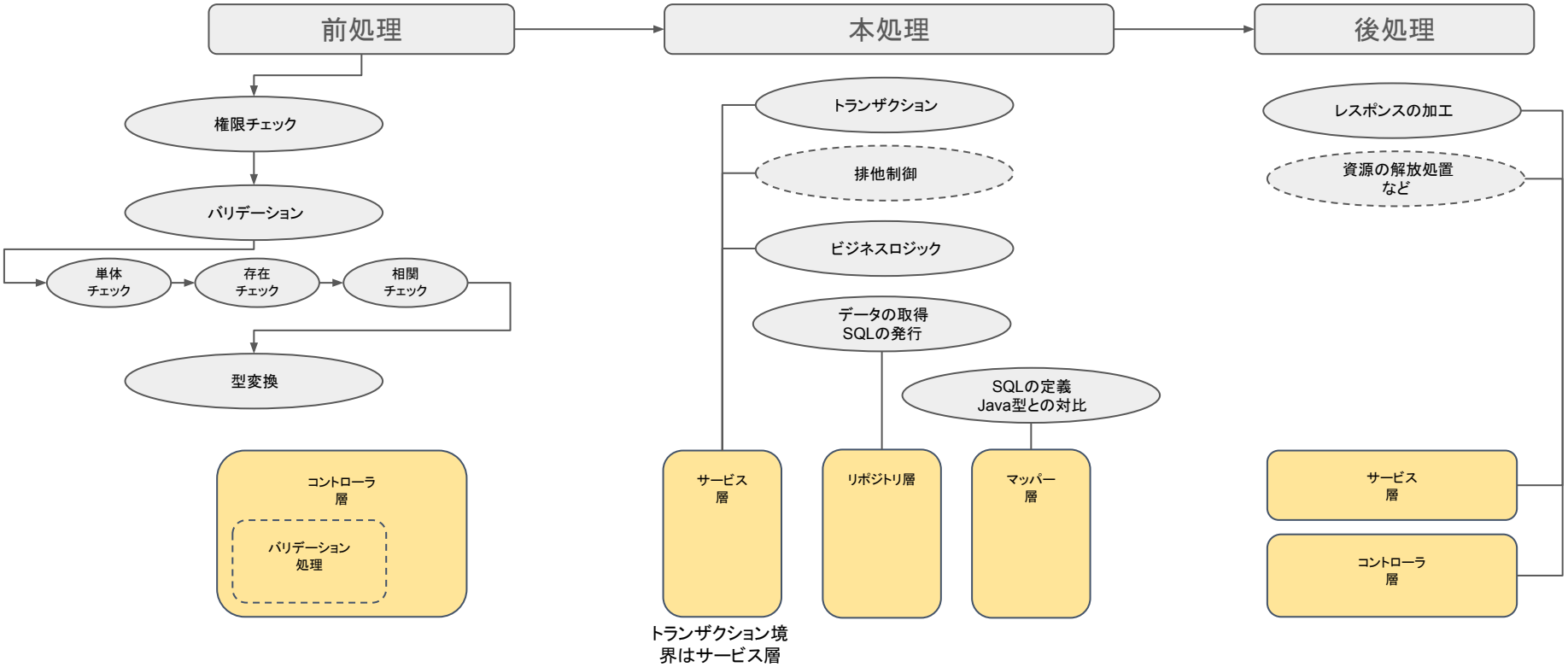
・認証の歴史を振り返る Basic認証からOAuthまで
・SessionとJWTとOAuth2とSAMLの比較
・脆弱性とセキュリティについてのおさらい
・認可の種類についてのおさらい URLベース / ロールベース / 属性ベース

・SpringSecurityの設定と実装を通した実践（jwtベース）
・SpringSecurityの設計思想に触れる
・SpringSecurityの機能を利用したロールベース認可の実現

・可能な限りアプリケーション独自のユーザーIDを発行せず、外部IdPを積極的に活用する

アプリケーションアーキテクチャの「型」

サーバサイドオンライン処理（API）開発において、その処理工程を、前処理・本処理・後処理の3ブロックにわけ、
すべてのAPIにおいて原則的に必要になる処理内容（楕円、点線は必要な場合のみ検討する）を以下の様にマッピングする



教材の構成

・Springboot+Reactによる簡単な注文システム一式のソースコード
講義はバックエンドのアプリケーションアーキテクチャがメインです。フロントエンドに対する解説コンテンツは現状ございません。

・解説スライド
・Googleスライドによる共有　もしくはPPT形式のダウンロードで提供

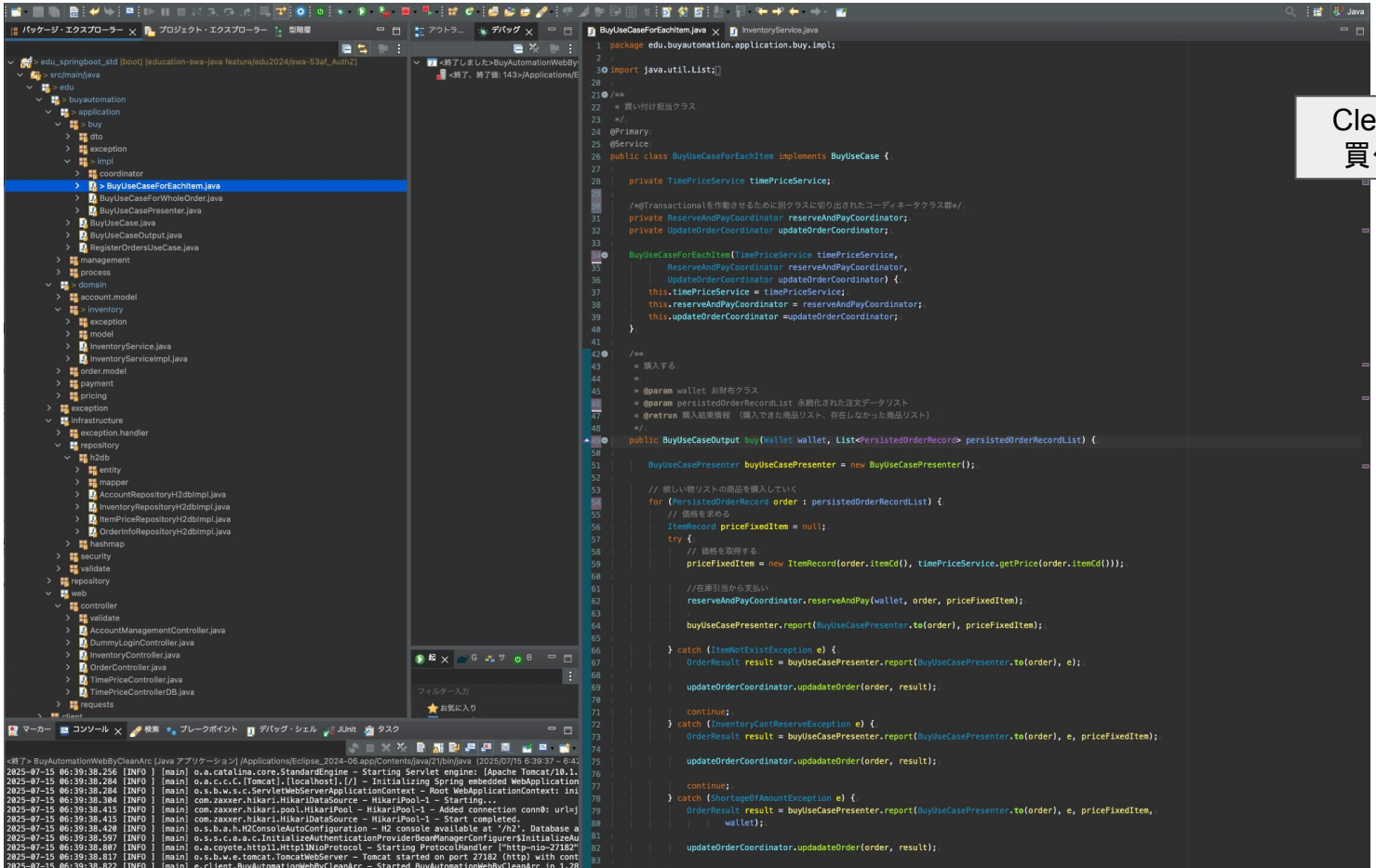
バックエンド

- フレームワーク: Spring Boot 3.3.1
- 言語: Java 21
- アーキテクチャ: 2パターン実装
  - Clean Architecture版
  - Layered Architecture版
- Web: Spring Web MVC
- セキュリティ: Spring Security + JWT (jjwt 0.12.5)
- ORM: MyBatis 3.5.19
- データベース: H2 Database（インメモリ）
- ユーティリティ: Lombok
- バリデーション: Bean Validation
- テスト: JUnit 5.10.2, Spring Boot Test
- ビルドツール: Gradle

フロントエンド

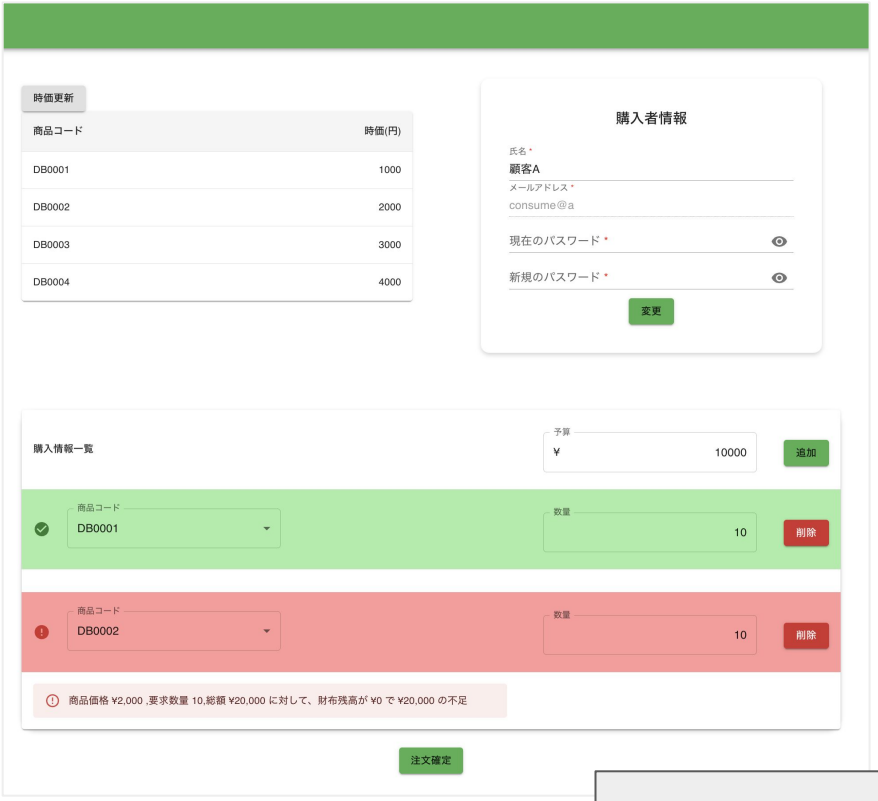
- フレームワーク: React 19.1.0
- 言語: TypeScript 5.8.3
- ビルドツール: Vite 6.0.1
- UIライブラリ: Material-UI (MUI) v7.1.0
- 状態管理:
  - Jotai 2.12.5（グローバル状態）
  - React Query 5.80.6（サーバー状態）
- フォーム: React Hook Form 7.53.0
- ルーティング: React Router v7.5.3
- HTTP通信: Axios 1.9.0
- 開発ツール: ESLint, TypeScript ESLint

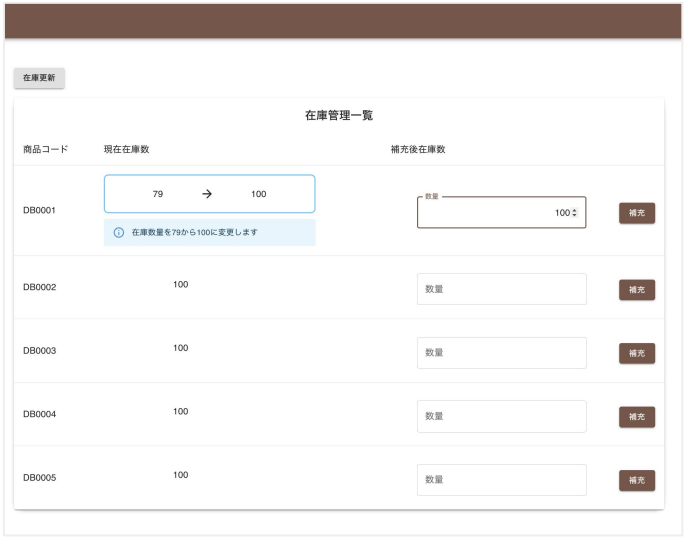
教材サンプル　1/2



Clean Architectureに沿った
買付処理ロジックの例です

教材サンプル　2/2





Reactで書かれたフロントエンドです