2023年度版 OUTSYSTEMS 社内勉強会資料

HITスクール

研修内容目次

1. **イントロダクションとOutSystems概要**
2. **データベース設計の基礎とエンティティ**
3. **インデックスと参照整合性**
4. **ユニークキーとデーター意性**
5. **削除規則（Delete Rule）と参照制御**
6. **画面設計とデータ表示**
7. **アグリゲートとデータアクション**
8. **例外処理とエラーハンドリング**
9. **実践演習・小テスト**
10. **まとめ・次回予告**

1. イントロダクションとOutSystems概要

解説

- **OutSystemsとは**：ローコード開発プラットフォーム。Web・モバイルアプリを高速で開発できる。
- **研修の目的**：データベース設計、画面設計、データ表示、例外処理など、実践的な開発スキルを習得する[1] [2]。
- **Service Studioの基本操作**：モジュール・エンティティ・画面の作成方法を簡単に紹介。

ワンポイントアドバイス

> 教材ダウンロードは必ず行い、実際に手を動かしながら学ぶことが理解の近道です[1]。

2. データベース設計の基礎とエンティティ

解説

- **エンティティ（テーブル）**：データを格納するための基本単位。例：顧客、注文、製品。
- **アトリビュート（属性）**：エンティティの列。例：顧客ID、顧客名。
- **主キー（Primary Key）**：エンティティ内で一意となるID。通常、自動生成される[1]。

サンプル（図式）

```
+-----------------+
| 顧客エンティティ |
+-----------------+
| 顧客ID（PK）     |
| 顧客名           |
| 住所             |
+-----------------+
```

3. インデックスと参照整合性

解説

- **インデックス**：データ検索を高速化するためのデータ構造。例：ID、マイナンバー[1]。
- **参照整合性**：エンティティ間のリレーション（例：顧客と注文）を正しく管理する仕組み。
- **インデックスのメリット・デメリット**
  - メリット：検索が高速
  - デメリット：ストレージ消費、更新時のオーバーヘッド[1]

ワンポイントアドバイス

> インデックスは多すぎるとパフォーマンスが悪化します。必要な箇所だけに設定しましょう[1]。

4. ユニークキーとデータ一意性

解説

- **ユニークキー**：複数のアトリビュートを組み合わせて一意性を保証。例：注文IDと製品IDの組み合わせ[1]。
- **主キーとの違い**：主キーは1つ、ユニークキーは複数設定可能。
- **実例**：1つの注文に同じ製品を複数登録できないようにする[1]。

サンプル（図式）

```
+----------------------+
| 注文-製品エンティティ |
+----------------------+
| 注文ID（PK）          |
| 製品ID（PK）          |
| 数量                  |
+----------------------+
（※注文ID＋製品IDでユニークキー）
```

5. 削除規則（Delete Rule）と参照制御

解説

- **Delete Rule**：参照先エンティティ削除時の挙動を定義[1]。
  - **Protect**：参照先がある場合は削除不可
  - **Delete**：参照先も一緒に削除（カスケード削除）
  - **Ignore**：参照先は無視して削除（リレーションなし）
- **実務での注意点**：Delete Ruleは慎重に設定。パフォーマンスやデータ整合性に影響[1]。

ワンポイントアドバイス

> データ整合性を保つため、基本的にはProtectやDeleteを使い、Ignoreは特別な場合のみ使用しましょう[1]。

6. 画面設計とデータ表示

解説

- **画面設計**：ユーザーが操作・閲覧する画面の構成[2]。
- **データ表示方法**：アグリゲートやデータアクションで取得したデータを画面に表示[2]。
- **ウィジェット**：テーブル、リスト、カードなど、データ表示に使う部品[2]。

サンプル（図式）

```
[画面]
  └─ [テーブルウィジェット]
```

```
    └ データソース：アグリゲート（注文一覧）
```

7. アグリゲートとデータアクション

解説

- **アグリゲート**：データベースから直接データを取得する仕組み[2]。
- **データアクション**：サーバー側でカスタムロジックを実行し、データを加工・取得[2]。
- **表示タイミング**：画面初期化時に自動実行、取得データは即時表示[2]。

サンプル（コード例）

```
// （疑似コード）アグリゲートで注文一覧を取得
var orders = Order.GetAll().OrderBy(o => o.Priority);
```

8. 例外処理とエラーハンドリング

解説

- **例外処理**：エラー発生時にユーザーに分かりやすいメッセージを表示[1]。
- **エクセプションハンドラー**：サーバーアクションで例外をキャッチし、適切な処理を行う[1]。
- **実例**：同じ商品を重複登録した場合、エラーメッセージを表示[1]。

サンプル（コード例）

```
// （疑似コード）例外処理
try {
  // 注文登録処理
} catch (Exception ex) {
  // エラーメッセージ表示
}
```

9. 実践演習・小テスト

演習課題

1. **エンティティ作成**：顧客、注文、製品のエンティティを作成し、リレーションを設定。
2. **ユニークキー設定**：注文-製品エンティティにユニークキーを設定し、重複登録できないことを確認。
3. **削除規則の設定**：顧客削除時に注文も削除されるようにDelete Ruleを設定。
4. **画面設計**：注文一覧画面を作成し、テーブルで表示。
5. **例外処理**：重複登録時のエラー処理を実装。

小テスト（例）

1. インデックスのメリットとデメリットを説明せよ。
2. ユニークキーと主キーの違いを説明せよ。
3. Delete RuleのProtect、Delete、Ignoreの違いを説明せよ。
4. アグリゲートとデータアクションの違いを説明せよ。

10. まとめ・次回予告

まとめ

- **データベース設計**：エンティティ、インデックス、ユニークキー、Delete Ruleの基本を習得。
- **画面設計**：アグリゲート、データアクションを使ったデータ表示。
- **例外処理**：ユーザーに分かりやすいエラーハンドリング。

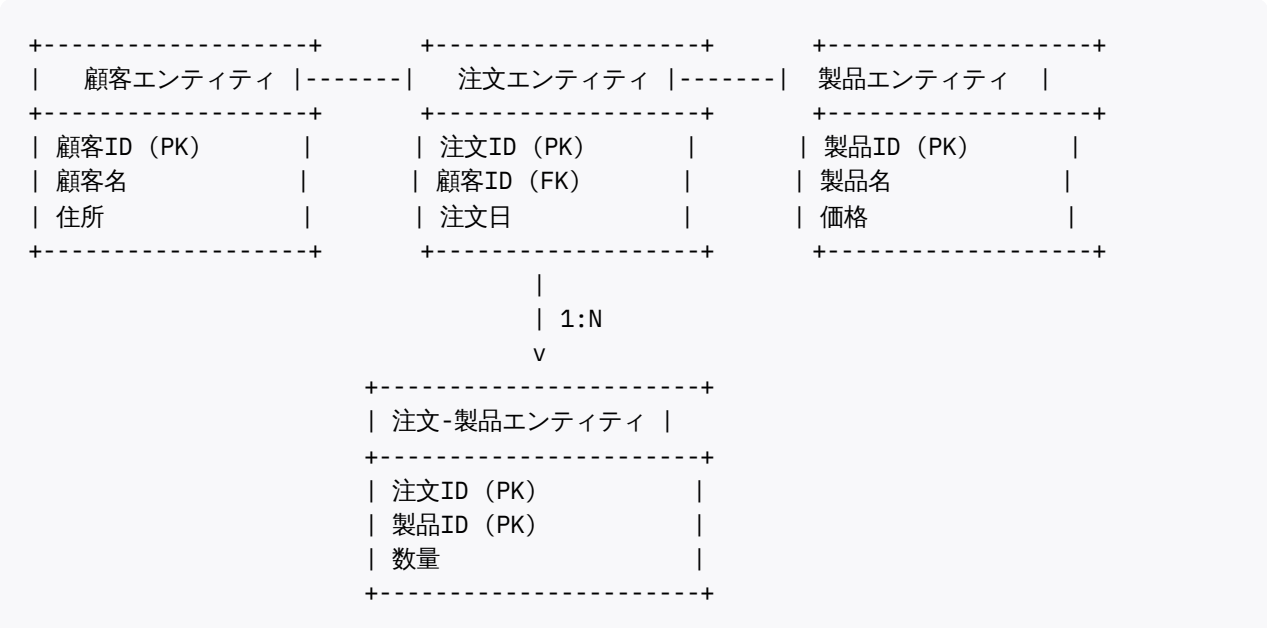
次回予告

- **外部システム連携**：REST API連携や外部データ取得の実践。
- **ワークフロー設計**：業務フローに沿った画面遷移の設計。

ワンポイントアドバイスまとめ

- **教材ダウンロードは必ず行い、実際に操作しながら学ぶ[1]。
- **インデックスは多すぎるとパフォーマンス低下。必要な箇所だけに[1]。
- **Delete Ruleはデータ整合性を意識して設定[1]。
- **例外処理はユーザーに分かりやすいメッセージを表示[1]。
- **アグリゲートとデータアクションの違いを理解し、適切に使い分ける[2]。

図式サンプル



サンプルコード（疑似コード）

```
// アグリゲートで注文一覧取得
var orders = Order.GetAll().OrderBy(o => o.Priority);

// 例外処理
try {
  // 注文登録処理
} catch (Exception ex) {
  // エラーメッセージ表示
}
```

この構成で、1日の研修（6時間）を効果的に進められます。各項目ごとに解説・サンプル・小テストを用意し、実践的なスキル習得を目指してください。

⁂

1. GMT20250521-015154_Recording_1920×1080.mp4.txt
2. GMT20250521-080412_Recording_1920×1080.mp4.txt