## フレームワーク応用実習

# 06. 複数テーブルの連携

株式会社ジードライブ

# 今回学ぶこと

- MyBatisを利用し、以下のような関係をもつテーブル同士の連携方法を学習する
  - １対１
  - １対 多
  - 多 対１
  - 多 対 多

# 1対1 の例

- 以下のような1対1の関連のあるテーブルを連携させる
  - 各テーブルの id が紐づいている

usersテーブル (親)

| id | name |
|---|---|
| 1 | 山田太郎 |
| 2 | 木村次郎 |
| 3 | 本田花子 |

user_detailsテーブル (子)

| id | email | age |
|---|---|---|
| 1 | taro@yamada.com | 33 |
| 2 | jiro@kimura.net | 29 |
| 3 | hanako@honda.co.jp | 35 |

# 1対1 の例

- テーブルに対応するドメインクラスでは、親に対し、連携に必要なプロパティを設ける

User.java (親)

```
public class User {
  private Integer id;
  private String name;
  private UserDetail userDetail; // 連携させるためのプロパティ
  … アクセッサなど …
}
```

UserDetail.java (子)

```
public class UserDetail {
  private Integer id;
  private String email;
  private Integer age;
  … アクセッサなど …
}
```

# 1対1 の例

- association要素を使い、resultMapを連携する

usersテーブルとUser.javaのマッピングファイル

```
<resultMap id="joinedResult" type="com.ex.domain.User">
    <id property="id" column="id" />
    <result property="name" column="name" />
    <association property="userDetail" resultMap="userDetailResult" />
</resultMap>

<resultMap id="userDetailResult" type="com.ex.domain.UserDetail">
    <id property="id" column="id" />
    <result property="email" column="email" />
    <result property="age" column="age" />
</resultMap>

<select id="getUsers" resultMap="joinedResult">
    SELECT * FROM users  JOIN user_details  ON users.id = user_details.id
</select>
```

usersテーブル

user_detailsテーブル

JOIN句を含むSQL

# 1対1 の例

- resultMapを分けずに、association要素内にテーブルとドメインクラスの対応を記述することも可能

**前ページと同等のマッピングファイル**

```
<resultMap id="joinedResult" type="com.ex.domain.User">
    <id property="id" column="id" />
    <result property="name" column="name" />
    <association property="userDetail" javaType="com.ex.domain.UserDetail">
        <id property="id" column="id" />
        <result property="email" column="email" />
        <result property="age" column="age" />
    </association>
</resultMap>

<select id="getUsers" resultMap="joinedResult">
    SELECT * FROM users  JOIN user_details  ON users.id = user_details.id
</select>
```

# 1対1 の例

- association要素を使わない連携も可能

**前ページと同等のマッピングファイル**

```
<resultMap id="joinedResult" type="com.ex.domain.User">
    <id property="id" column="id" />
    <result property="name" column="name" />
    <result property="userDetail.email" column="email" />
    <result property="userDetail.age" column="age" />
</resultMap>

<select id="getUsers" resultMap="joinedResult">
    SELECT * FROM users  JOIN user_details  ON users.id = user_details.id
</select>
```

# 1対多 の例

- 以下のような1対多の関連のあるテーブルを連携させる
  - authorsテーブルの id と booksテーブル author_id が紐づく

authorsテーブル (親)

| id | name |
|---|---|
| 1 | アガサ・クリスティ |
| 2 | コナン・ドイル |

booksテーブル (子)

| id | title | story | author_id |
|---|---|---|---|
| 1 | 緋色の研究 | ホームズとワトスンが出会う... | 2 |
| 2 | スタイルズ荘の怪事件 | イギリスに帰還したヘイスティングス... | 1 |
| 3 | バスカヴィル家の犬 | 大富豪のバスカヴィル家には魔犬の... | 2 |
| 4 | ゴルフ場殺人事件 | ある富豪からの依頼を受け、ポワロは... | 1 |
| 5 | 牧師館の殺人 | ロンドン郊外の小さな村で事件が... | 1 |

# 1対多 の例

- ドメインクラスでは、親に対し、連携するクラスのListをプロパティとして記述する

Author.java (親)

```
public class Author {
  private Integer id;
  private String name;
  private List<Book> books; // 対応するクラスのリスト
  … アクセッサなど …
}
```

Book.java (子)

```
public class Book {
  private Integer id;
  private String title;
  private String story;
  private Integer authorId;
  … アクセッサなど …
}
```

# 1対多 の例

- 1対多の連携では、associationの代わりにcollection要素で対応することが可能
  - 基本的な用法はassociationと同じ
  - ofType属性でListで扱う型を指定する

**authorsテーブルとAuthor.javaのマッピングファイル**

```
<resultMap id="joinedResult" type="com.ex.domain.Author">
    <id property="id" column="author_id" />
    <result property="name" column="name" />
    <collection property="books" ofType="com.ex.domain.Book">
      <id property="id" column="id" />
      <result property="title" column="title" />
      <result property="story" column="story" />
    </collection>
</resultMap>
<select id="getAuthors" resultMap="joinedResult">
    SELECT * FROM authors JOIN books ON authors.id = books.author_id
</select>
```

booksテーブルと
Bookクラスのマッピング

# 多対1 の例

- 以下のような多対1の関連のあるテーブルを連携させる
  - 一つの著書に対し、複数の著者(共著)はないという前提

booksテーブル (親)

| id | title | story | author_id |
|---|---|---|---|
| 1 | 緋色の研究 | ホームズとワトスンが出会う... | 2 |
| 2 | スタイルズ荘の怪事件 | イギリスに帰還したヘイスティングス... | 1 |
| 3 | バスカヴィル家の犬 | 大富豪のバスカヴィル家には魔犬の... | 2 |
| 4 | ゴルフ場殺人事件 | ある富豪からの依頼を受け、ポワロは... | 1 |
| 5 | 牧師館の殺人 | ロンドン郊外の小さな村で事件が... | 1 |

authorsテーブル (子)

| id | name |
|---|---|
| 1 | アガサ・クリスティ |
| 2 | コナン・ドイル |

# 多対1 の例

- ドメインクラスでは、親に対し、連携のためのプロパティを記述する

**Book.java (親)**

```
public class Book {
  private Integer id;
  private String title;
  private String story;
  private Author author;  // private Integer authorId  は不要
  … アクセッサなど …
}
```

**Author.java (子)**

```
public class Author {
  private Integer id;
  private String name;
  … アクセッサなど …
}
```

# 多対1 の例

- association要素を使い、連携させる
  - javaType属性でプロパティの型を指定する

booksテーブルとBook.javaのマッピングファイル

```
<resultMap id="joinedResult" type="com.ex.domain.Book">
    <id property="id" column="id" />
    <result property="title" column="title" />
    <result property="story" column="story" />
    <association property="author"
                 javaType="com.ex.domain.Author">
        <id property="id" column="author_id" />
        <result property="name" column="name" />
    </association>
</resultMap>
<select id="getBooks" resultMap="joinedResult">
    SELECT *, books.id FROM books
    JOIN authors ON books.author_id = authors.id
</select>
```

authorsテーブルとAuthorクラスのマッピング

# 多対多 の例

- 以下のような多対多の関連のあるテーブルを連携させる
  - 生徒(student)は複数のコース(course)に所属でき、
    各コースは複数人の生徒を所属させることができる

studentsテーブル (親)

| id | name | age |
|---|---|---|
| 1 | 山田太郎 | 22 |
| 2 | 木村次郎 | 24 |
| 3 | 本田花子 | 19 |

student_courseテーブル (中間)

| student_id | course_id |
|---|---|
| 1 | 2 |
| 1 | 3 |
| 2 | 1 |
| 3 | 1 |
| 3 | 3 |

coursesテーブル (子)

| id | title |
|---|---|
| 1 | 英会話 |
| 2 | プログラミング |
| 3 | 油絵 |

多対多の関連では、中間のテーブルが必要になる

# 多対多 の例

studentsテーブルを
親とした場合

- 1対多と同様のドメインクラスを作成する
  - 中間テーブルに対応するドメインクラスは不要

**Student.java (親)**

```
public class Student {
  private Integer id;
  private String name;
  private Integer age;
  private List<Course> courses; // 対応コースをリストで保持
  … アクセッサなど …
}
```

**Course.java (子)**

```
public class Course {
  private Integer id;
  private String title;
  … アクセッサなど …
}
```

# 多対多 の例

studentsテーブルを
親とした場合

- SELECT文では、JOIN句を２つ記述する
  - まず中間テーブルと関連付け、次に子テーブルと関連付ける

**studentsテーブルとStudent.javaのマッピングファイル**

```
<resultMap id="joinedResult" type="com.ex.domain.Student">
    <id property="id" column="id" />
    <result property="name" column="name" />
    <result property="age" column="age" />
    <collection property="courses" ofType="com.ex.domain.Course">
      <result property="title" column="title" />
    </collection>
</resultMap>

<select id="getStudents" resultMap="joinedResult">
    SELECT * FROM students
    JOIN student_course ON students.id = student_course.student_id
    JOIN courses ON student_course.course_id = courses.id
</select>
```

1対多と
同様

親テーブルと中間テーブルをJOIN

中間テーブルと子テーブルをJOIN

# テーブル連携時の注意点

- 以下のように連携するテーブルのカラム名が重複する場合、親テーブルのデータを取得できないので注意する

booksテーブル (親)

| id | name | story | author_id |
|---|---|---|---|
| 1 | 緋色の研究 | ホームズとワトスンが出会う... | 2 |
| 2 | スタイルズ荘の怪事件 | イギリスに帰還したヘイスティングス... | 1 |
| 3 | バスカヴィル家の犬 | 大富豪のバスカヴィル家には魔犬の... | 2 |
| 4 | ゴルフ場殺人事件 | ある富豪からの依頼を受け、ポワロは... | 1 |
| 5 | 牧師館の殺人 | ロンドン郊外の小さな村で事件が... | 1 |

authorsテーブル (子)

| id | name |
|---|---|
| 1 | アガサ・クリスティ |
| 2 | コナン・ドイル |

# テーブル連携時の注意点

- この問題はテーブルのカラム名をASで変更することで回避可能

booksテーブルとBook.javaのマッピングファイル

```
<resultMap id="joinedResult" type="com.ex.domain.Book">
    <id property="id" column="id" />
    <result property="name" column="name" />
    <result property="story" column="story" />
    <association property="author" column="author_id
                         javaType="com.ex.domain.Author">
        <id property="id" column="author_id" />
        <result property="name" column="author_name" />
    </association>
</resultMap>
<select id="getBooks" resultMap="joinedResult">
    SELECT books.id, books.name, books.story,
           authors.id AS author_id, authors.name AS author_name
    FROM books JOIN authors ON books.author_id = authors.id
</select>
```

**ASで変更したものを column属性として設定する**

# autoMapping

- resultMap, association, collection内では、すべてのカラムについて対応関係を記す必要はない
  - 基本的には autoMapping="true" と記述して、対応関係の記述は省略することが可能
  - ただし、1対多の関係を使う場合は、<id>タグについては省略しない方がよい
    ⇒ TooManyResultsExceptionが発生する可能性が生じるため
  - ASの利用により、カラム名がドメインクラスのフィールド名と不一致になる場合は、手動で対応関係を記す必要がある

# autoMappingの記述例

- 以降のページでは、以下のようなbooksテーブルとBookオブジェクトの対応関係を記述する例を取り上げる

booksテーブル

| id | title | author |
|---|---|---|
| 1 | 蜘蛛の糸 | 芥川龍之介 |
| 2 | 坊ちゃん | 夏目漱石 |

Book.java

```
@Data
public class Book {
    private Integer id;
    private String title;
    private String author;
}
```

# autoMappingの記述例

例：booksテーブルとBook.javaのマッピング

```
<resultMap id="Map" type="com.example.ex.domain.Book">
    <id property="id" column="id" />
    <result property="title" column="title" />
    <result property="author" column="author" />
</resultMap>

<select id="selectAll" resultMap="map">
   SELECT id, title, author FROM books
</select>
```

propertyとcolumnが一致しているので自動マッピングが可能

autoMappingの適用

```
<resultMap id="Map" type="com.example.ex.domain.Book"
                    autoMapping="true">
</resultMap>
```

# autoMappingの記述例

例：booksテーブルとBook.javaのマッピング

```
<resultMap id="Map" type="com.example.ex.domain.Book">
    <id property="id" column="id" />
    <result property="title" column="title" />
    <result property="author" column="name" />
</resultMap>

<select id="selectAll" resultMap="map">
   SELECT id, title, author AS name FROM books
</select>
```

自動マッピング可能

propertyとcolumnが不一致しているので自動マッピングができない

autoMappingの適用

```
<resultMap id="Map" type="com.example.ex.domain.Book"
                    autoMapping="true">
    <result property="author" column="name" />
</resultMap>
```

ASを使用したので、手動で対応する

# 要素の記述順

- resultMap内は、id要素、result要素、association要素、collection要素の順に記述する必要がある

```
<resultMap …>
    <id … />
    <result … />
    <association …>
        …
    </association>
    <collection …>
        …
    </collection>
</resultMap>
```

① id
② result
③ association
④ collection の順に記述する

# 自動生成されたIDの取得

- データ追加時に自動生成されたIDを取得する場合、keyProperty属性を設定する

マッピングファイル

```
<insert id="insert" parameterType="com.ex.domain.Student"
  useGeneratedKeys="true" keyProperty="id">
  INSERT INTO students (name, age, address)
  VALUES
  (#{name}, #{age}, #{address})
</insert>
```

Studentオブジェクトのidフィールドに自動生成されたIDが入るようになる

Mapperを利用するJavaのコード

```
studentMapper.insert(student); // これが実行された時点で、IDが入る
System.out.println(student.getId()); // IDが表示される
```

# 練習

- 練習06-1
- 練習06-2
- 練習06-3
- 練習06-4
- 練習06-5
- 練習06-6
- 練習06-7