

A faint, light blue world map is visible in the background of the slide, centered behind the text.

# INTERNET ACADEMY

Institute of Web Design & Software Services

## Spring Boot 3

インターネット・アカデミー

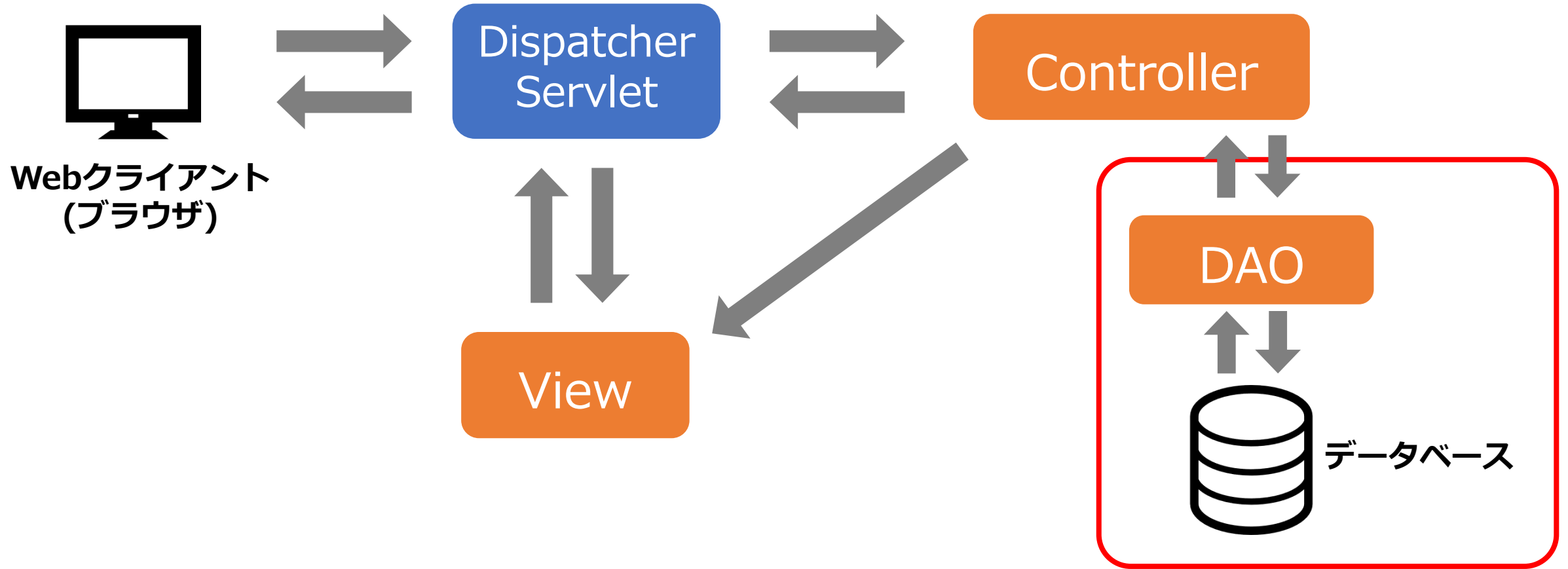
# Spring Boot 3 目次

- Spring Bootでのデータベース処理
- H2データベース
- 応用編 1 （データベースへの登録）

A faint, light blue world map is visible in the background of the slide, centered behind the title text.

# Spring Bootでのデータベース処理

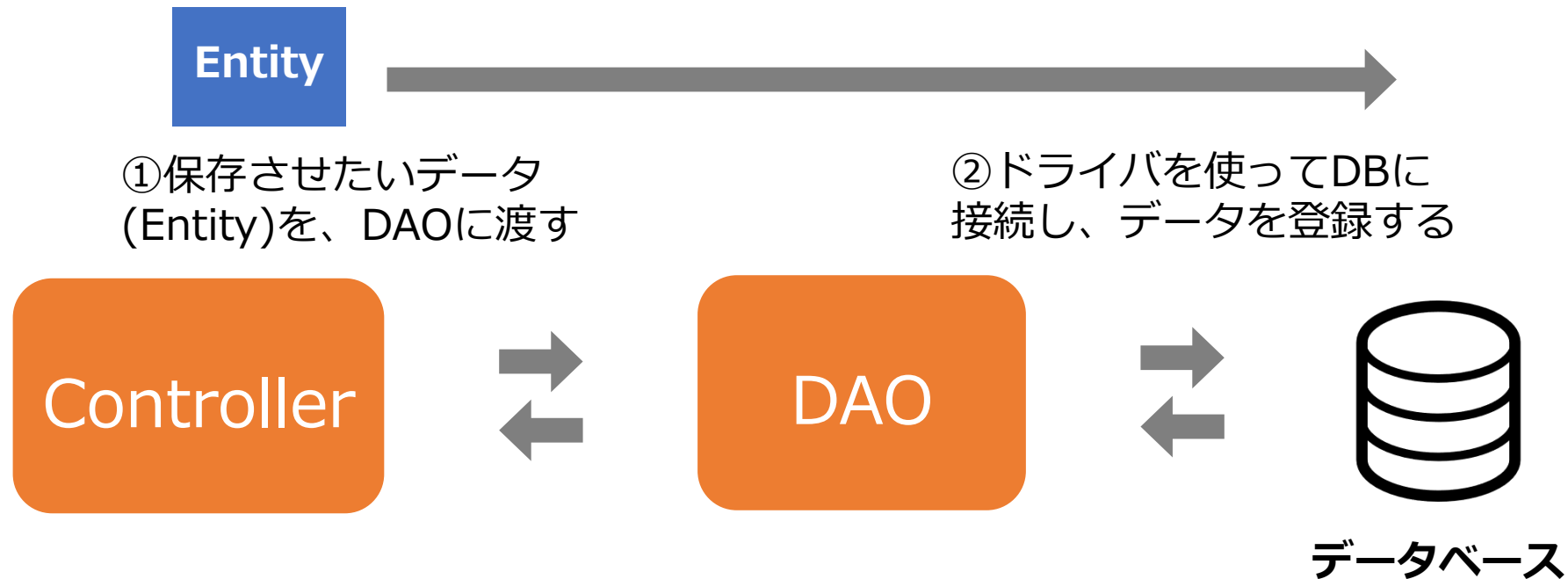
# Spring Bootにおけるデータベース処理



# DAOパターン

Data Access Objectの略で、データベース操作におけるデザインパターンの1つ。DAOとは、Entityを受け取ってデータベースに保存したり、検索するためにSQLを発行するオブジェクト。

## ・データ挿入時(INSERT)



# DAOパターン

## ・データ検索時(SELECT)

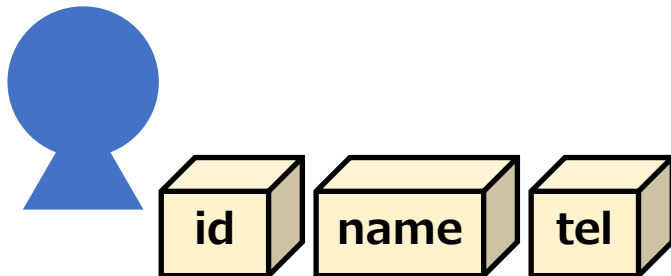


# O/Rマッピング

オブジェクト(Object)とデータベース(Relational Database)を対応させる仕組み

オブジェクト  
(エンティティ)

```
public class User{  
    private int id;  
    private Sting name;  
    private String tel;  
}
```



リレーショナル  
データベース

列名	型
id	INT
name	TEXT
tel	TEXT







## H2データベース

Java上で動く、オープンソースのインメモリデータベース。  
JDBCなどを用いてアクセスできる。SpringBootでは、予めインストールされている。src/main/resource直下にある、 **schema.sql**と**data.sql**を読み込んで初期化する。

### インメモリデータベース

アプリを起動するたびに初期化される。初期化時には、schema.sqlとdata.sqlが自動で実行される。練習やテストなどで、簡易的なデータベースを作りたいときに使われる。

# H2データベース

## データベース接続設定

### application.properties

```
spring.datasource.driverClassName: org.h2.Driver  
spring.datasource.url: jdbc:h2:mem:test  
spring.datasource.username: sa  
spring.datasource.password:  
spring.h2.console.enabled: true
```

H2データベースのドライバー。プログラムからデータベースにアクセスする時に使われるソフト

データベースの設置場所を指す。「jdbc:h2:」までは固定。H2データベースがインストールされた場所のmem内に、testというデータベースを作成している

ユーザー名とパスワードは、デフォルトの管理者として利用

ブラウザでデータベースの確認をするH2consoleの利用を許可

# H2データベース

## データベース初期化

**schema.sql** ※ファイル名注意！

```
CREATE TABLE sample(  
  id INT NOT NULL AUTO_INCREMENT,  
  name VARCHAR(100) NOT NULL,  
  PRIMARY KEY(id)  
);
```

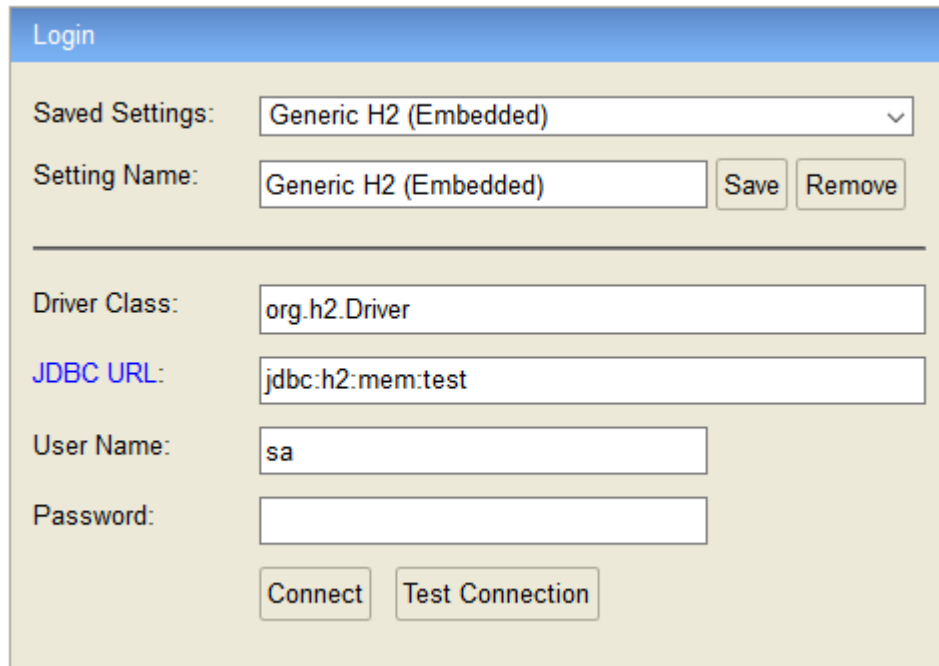
**data.sql**

```
INSERT INTO sample (name) VALUES ('taro');
```

# H2データベースのコンソール

## データベースの確認

1. 3つのファイル(application.properties、schema.sql、data.sql)を作成
2. アプリケーションを実行する
3. ブラウザで「localhost:8080/h2-console」にアクセス



The screenshot shows the H2 database console login interface. It has a blue header with the word "Login". Below the header, there are several input fields and buttons. The "Saved Settings:" dropdown menu is set to "Generic H2 (Embedded)". The "Setting Name:" text box also contains "Generic H2 (Embedded)", with "Save" and "Remove" buttons to its right. A horizontal line separates this section from the connection details. The "Driver Class:" text box contains "org.h2.Driver". The "JDBC URL:" text box contains "jdbc:h2:mem:test". The "User Name:" text box contains "sa". The "Password:" text box is empty. At the bottom, there are "Connect" and "Test Connection" buttons.

Login	
Saved Settings:	Generic H2 (Embedded) ▼
Setting Name:	Generic H2 (Embedded) [Save] [Remove]
<hr/>	
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:mem:test
User Name:	sa
Password:	
[Connect] [Test Connection]	

# H2データベースのコンソール

Login

Saved Settings: Generic H2 (Embedded) ▾

Setting Name: Generic H2 (Embedded) Save Remove

---

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:test

User Name: sa

Password:

Connect Test Connection

jdbc:h2:mem:testに変更  
(コロンの付け忘れに注意)

ユーザー名とパスワードは変更なしでOK

# H2データベースのコンソール

The screenshot shows the H2 database console interface. At the top, there is a toolbar with icons for undo, redo, and other functions. Below the toolbar, on the left, is a tree view showing the database structure: 'jdbc:h2:mem:test' (selected), 'SAMPLE' (highlighted with a red box and labeled ①), 'INFORMATION\_SCHEMA', 'Sequences', and 'Users'. The main area on the right contains a 'SQL statement:' input field with the text 'SELECT \* FROM SAMPLE;'. Above this field are buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear'. The 'Run' button is highlighted with a red box. Below the input field, the results of the query are displayed in a table format, also highlighted with a red box. The table has two columns: 'ID' and 'NAME'. The first row contains the values '1' and 'taro'. Below the table, it says '(1 row, 3 ms)'. An 'Edit' button is located below the table.

① 押す

② 下に「SELECT \* . . .」が表示されていることを確認して押す

③ テーブルの内容が表示される

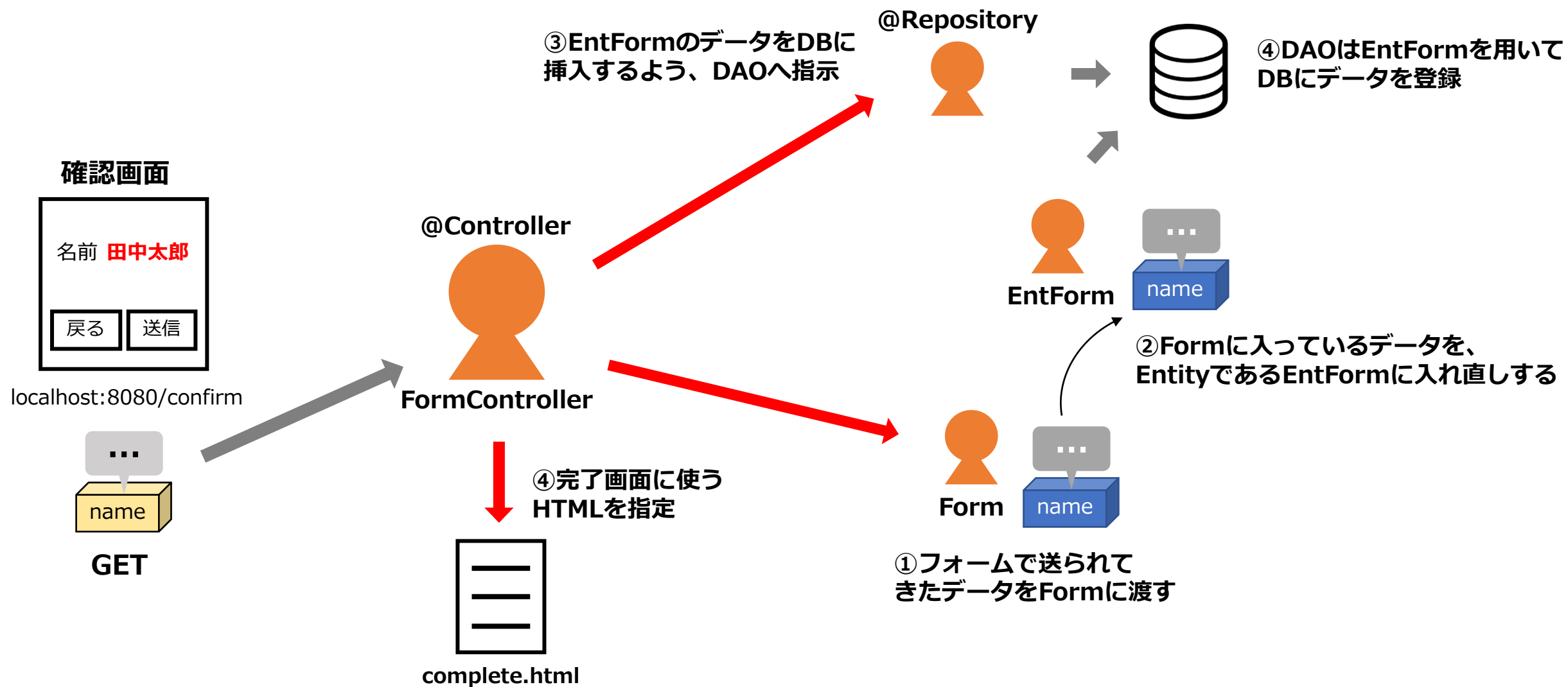
ID	NAME
1	taro

(1 row, 3 ms)

A faint, light blue world map is visible in the background of the slide, centered behind the main text.

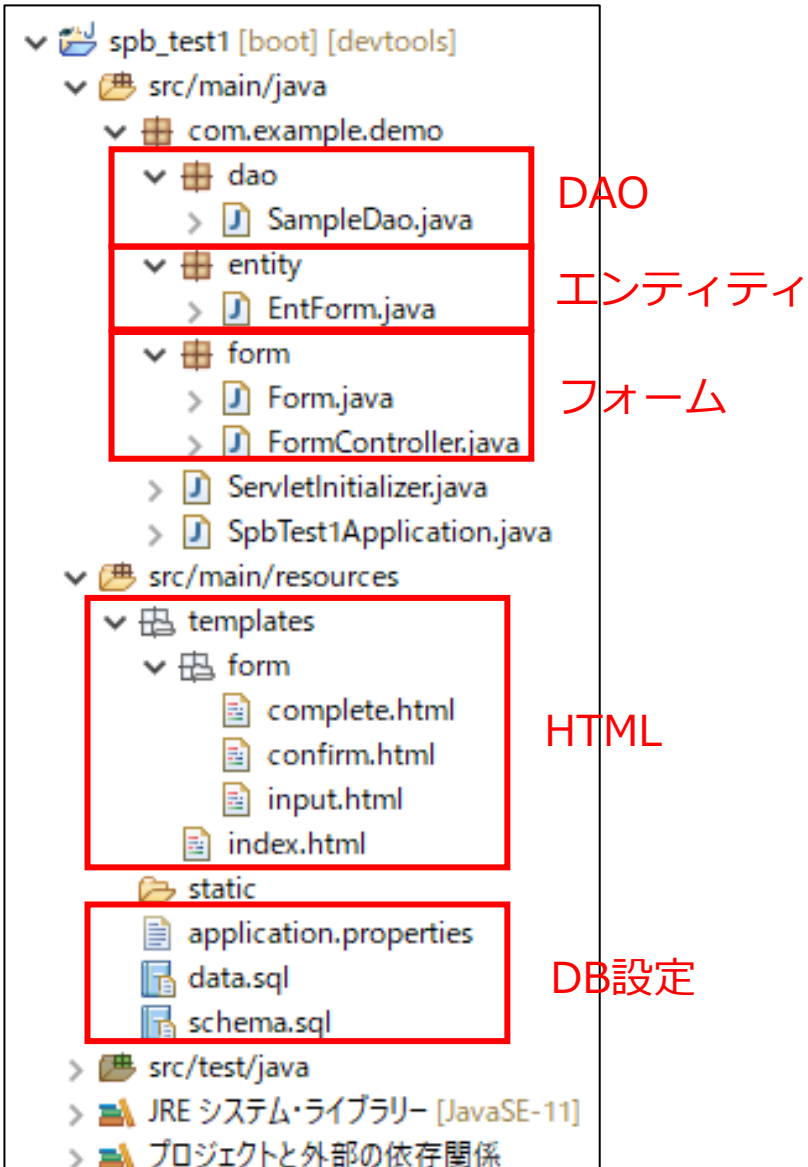
# データベース処理をするコード

# FormControllerの動きのイメージ





# 作業：フォームのデータをDBへ登録



## 作業の流れ

- ↓ トップページ表示  
FormController.java、index.html
- ↓ 入力画面表示  
FormController.java、form.html
- ↓ 確認画面表示  
FormController.java、confirm.html、Form.java
- ↓ データベース設定  
application.properties、schema.sql、data.sql
- ↓ エンティティ作成  
EntForm.java
- ↓ DAO作成  
SampleDao.java
- ↓ Controllerと完了画面の用意  
SampleController.java、complete.html

# H2データベース

## データベース接続設定

application.properties

spring.datasource.driverClassName: org.h2.Driver

spring.datasource.url: jdbc:h2:mem:test

spring.datasource.username: sa

spring.datasource.password:

spring.h2.console.enabled: true

H2データベースのドライバー。プログラムからデータベースにアクセスする時に使われるソフト

データベースの設置場所を指す。「jdbc:h2:」までは固定。H2データベースがインストールされた場所のmem内に、testというデータベースを作成している

ユーザー名とパスワードは、デフォルトの管理者として利用

ブラウザでデータベースの確認をするH2consoleの利用を許可

# H2データベース

## データベース初期化

schema.sql

```
CREATE TABLE sample(  
  id INT NOT NULL AUTO_INCREMENT,  
  name VARCHAR(100) NOT NULL,  
  PRIMARY KEY(id)  
);
```

data.sql

```
INSERT INTO sample (name) VALUES ('taro');
```

# データベース処理(エンティティ)

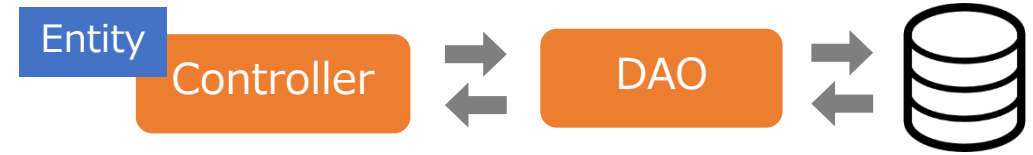
EntForm.java(エンティティ)

```
public class EntForm{
    private int id;
    private String name;

    public EntForm() {};
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```



# データベース処理(DAO)



SampleDao.java(データベース書き込み)

```
package com.example.demo.dao;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import com.example.demo.entity.EntForm;
```

**@Repository** **DAOの印**

```
public class SampleDao{
```

```
    private final JdbcTemplate db;
```

```
    @Autowired
```

```
    public SampleDao(JdbcTemplate db) {
```

```
        this.db = db;
```

```
    }
```

```
    public void insertDb(EntForm entform) {
```

```
        db.update("INSERT INTO sample (name) VALUES(?)",entform.getName() );
```

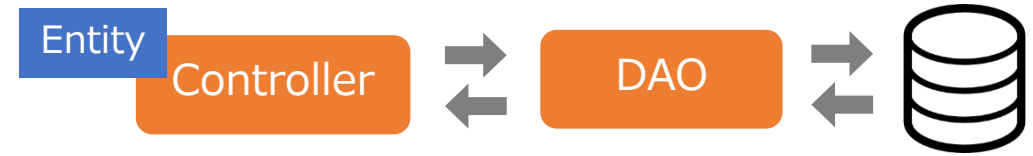
```
    }
```

```
}
```

**DB書き込み用  
オブジェクトの用意**

プリペアドステートメント  
…SQLインジェクションへの対策。  
「?」をプレースホルダーと呼ぶ

# データベース処理(Controller)



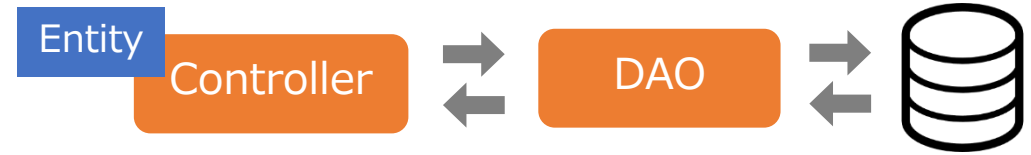
FormController.javaに追記

```
:
@Controller
public class FormController {
    :
    :
    private final SampleDao sampledao;
    @Autowired
    public FormController(SampleDao sampledao) {
        this.sampledao = sampledao;
    }

    @RequestMapping ("/complete")
    public String complete(Form form, Model model){
        EntForm entform = new EntForm();
        entform.setName(form.getName());

        sampledao.insertDb(entform);
        return "form/complete";
    }
}
```

# データベース処理(Controller)



FormController.javaに追記

```
:
@Controller
public class FormController {
    :
    :
    private final SampleDao sampledao;
    @Autowired
    public FormController(SampleDao sampledao) {
        this.sampledao = sampledao;
    }

    @RequestMapping ("/complete")
    public String complete(Form form, Model model){
        EntForm entform = new EntForm();
        entform.setName(form.getName());
        sampledao.insertDb(entform);

        return "form/complete";
    }
}
```

DAOのオブジェクトを用意

localhost:8080/completeでアクセスされたとき実行

確認画面から引き継いだデータ(form)を受け取って、  
EntForm型のオブジェクト(entform)にセット。  
最後に、DAO(sampledao)にDB登録させる

完了画面の表示