



# Webアプリケーション開発演習

前期の復習 その3

**今日はMVC・DBについて、思い出しましょう！**



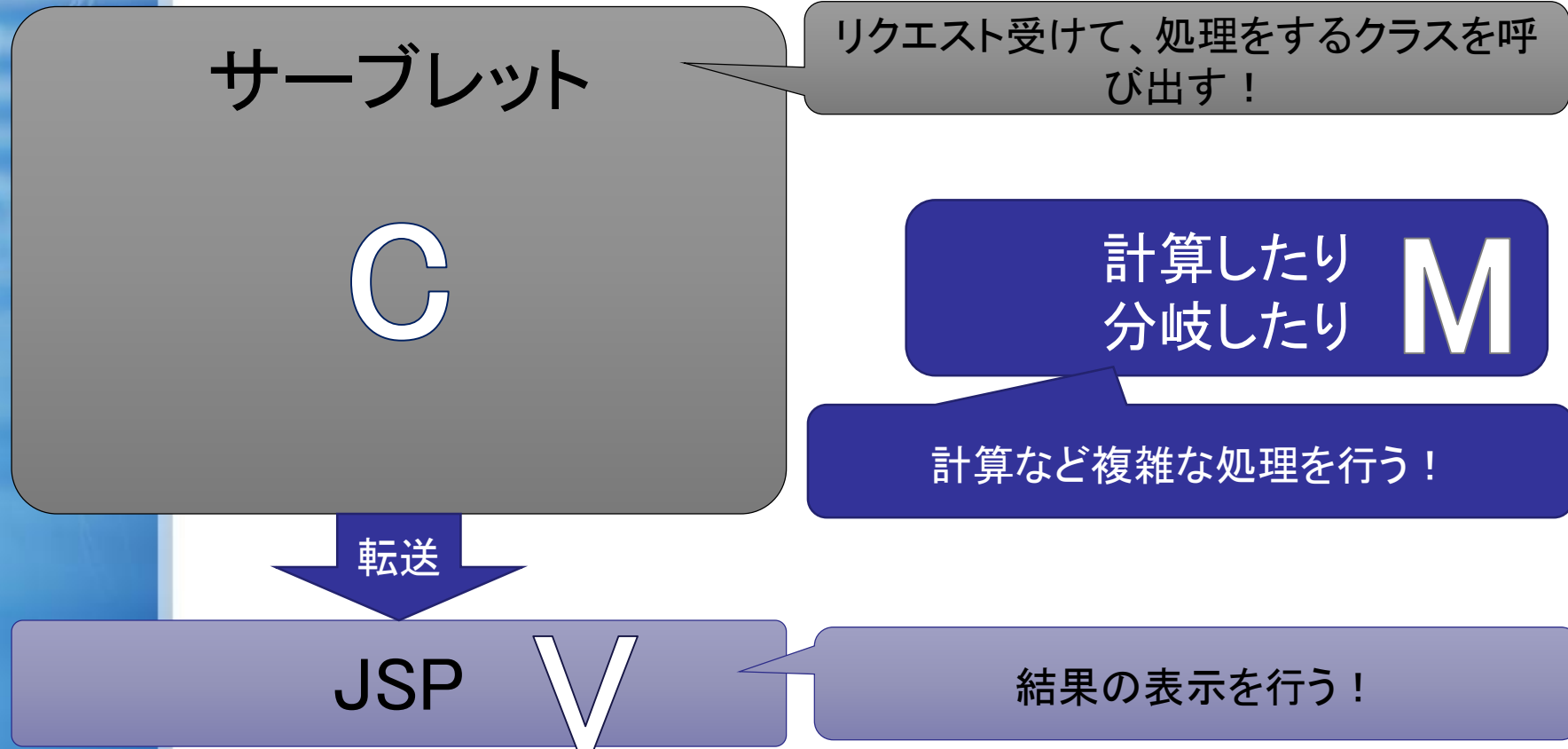
## 目次

- **MVCモデルとは？**
- **JavaBeansって何だった？**
- **DBの扱い方！**
- **演習**

## 目次

- **MVCモデルとは？**
- **JavaBeansって何だった？**
- **DBの扱い方！**
- **演習**

## 今までのやり方の問題点



# MVCモデルよく例にでるのが、受付業務

登録したいらしいよ。  
これが登録情報！

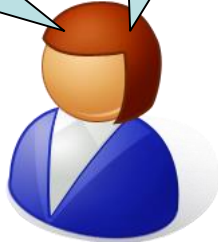
登録係さんよろしく  
お願いします！

アイアイサー！  
登録しまっせー！

じゃあ必要事項  
を書け！

了解！登録係に  
渡すね！

登録してください



受付



割振り



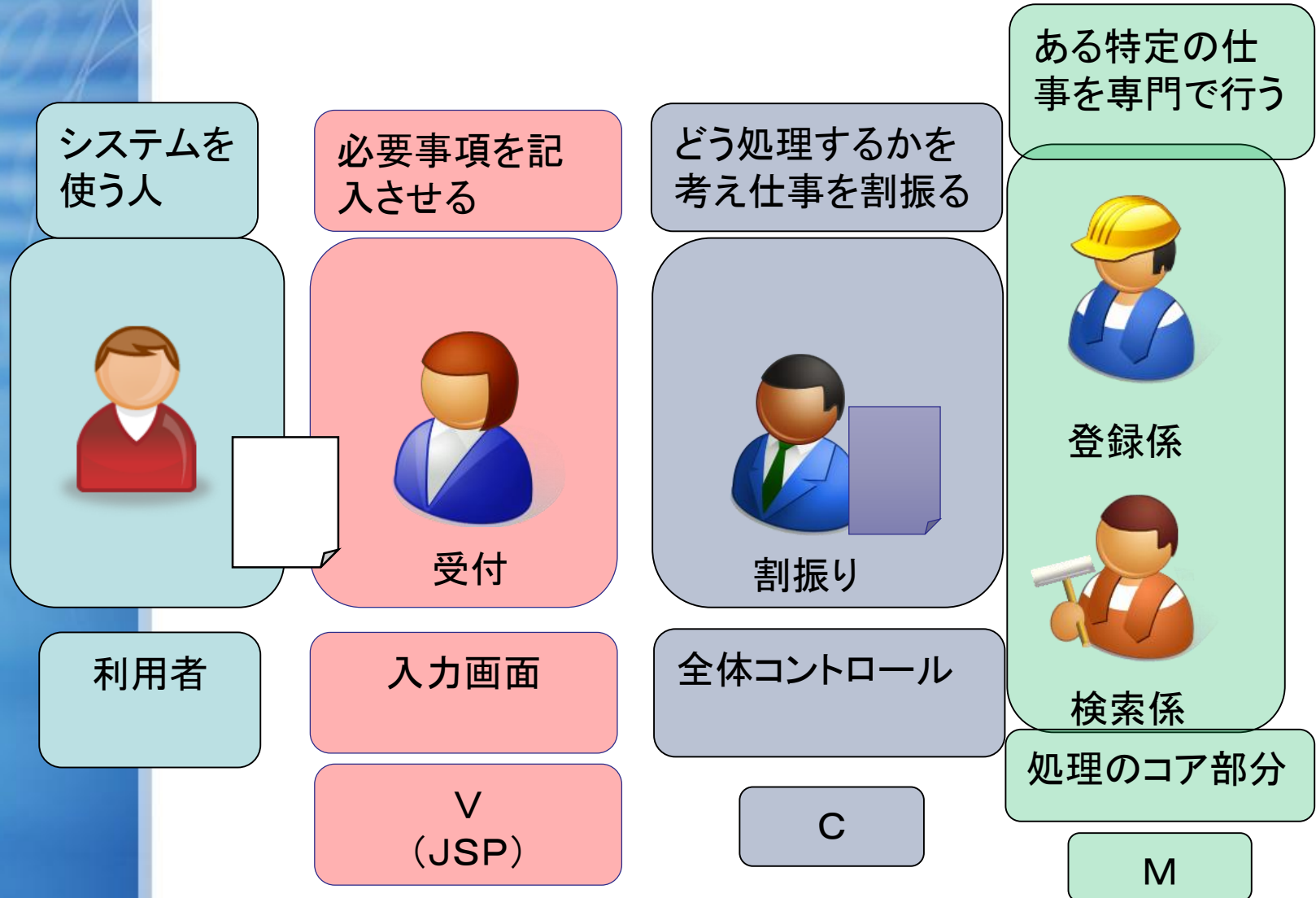
登録係



検索係

## MVCモデルのコツ

# 登場人物の役割に注目





## MVCモデルのコツ

# じゃあ、なにがメリットなん？

イメチェンするでー！

あ、そう仕事は  
ちゃんとしてね

ん？なんか  
あったん？



受付

V  
(JSP)



割振り

C



登録係



検索係

M

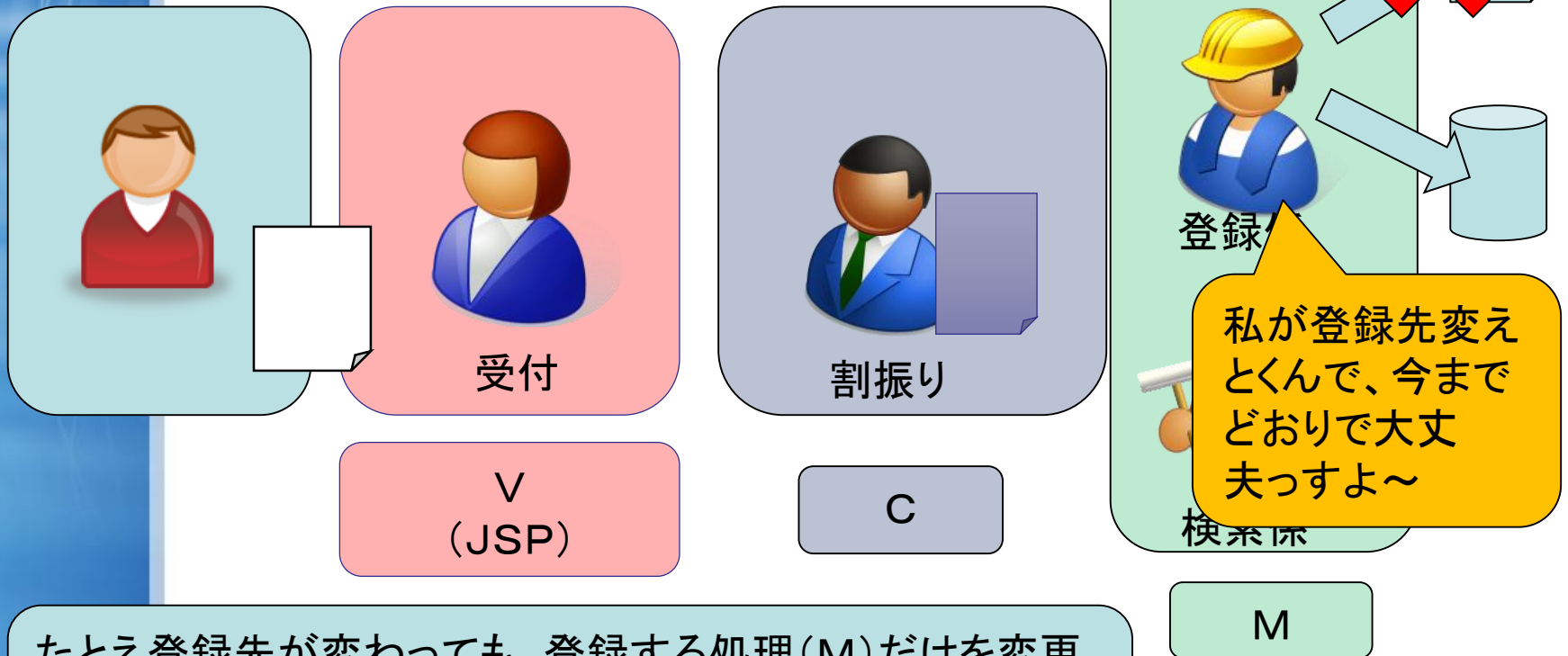
渡ってくる情報が一緒なら、CとMは**何の影響も受けない**！



## MVCモデルのコツ

# じゃあ、なにがメリットなん？

登録先が紙からDBに変わった！



たとえば登録先が変わっても、登録する処理(M)だけを変更すればOK！ VもCも影響は受けない！

## 役割はわかったけど、実際どうソースを組めば良い？

役割	分ける基準
M	システムのコアとなる処理(一般的にそれをビジネスロジックと言う) 例) ログイン処理、ユーザー登録処理、履歴登録処理
V	画面 JSP
C	リクエストやセッションから情報を取得する処理 リクエストやセッションに情報を設定する処理 JSPに処理を転送する処理 入力データの変換(文字列→数値変換、日付変換など) 入力データのチェック処理(文字数チェックなど)

## 役割はわかったけど、実際どうソースを組めば良い？

ざっくりした言い方をすると

Vで入力したデータを  
Cでチェック、加工や準備をして  
Mで処理し、Vで結果を表示する

## MVCモデルのコツ

では、処理をどうわける？

### 登録処理

処理内容	M? V? C?
登録情報を入力する画面	V
登録情報に入力ミスがないか、整合性が取れているかのチェック	C
登録情報をデータベースに保存する	M
登録情報に入力ミスがある場合、エラー画面に遷移させる	C
登録に成功した場合、登録完了画面に処理を転送する	C
登録に失敗した場合、エラー画面に処理を転送する	C

## 目次

- MVCモデルとは？
- **JavaBeansって何だった？**
- DBの扱い方！
- 演習

JavaBeansって何だっけ？

**JavaBeansざっくりいうと...**

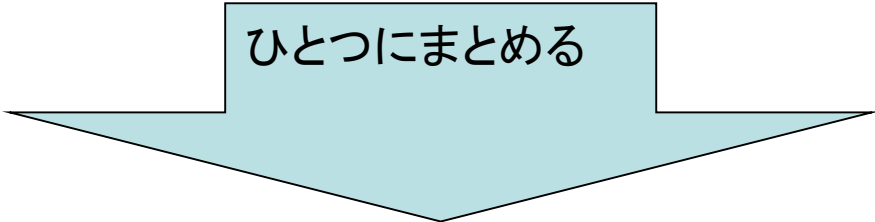
**情報を保持する為のクラス**



JavaBeansって何だっけ？

**例えば・・・**

**登録画面には入力項目がたくさんあります  
ひとつひとつを別々の変数にすると  
登録項目をセッションに保存する時は  
項目数分setAttributeするので大変です**



ひとつにまとめる

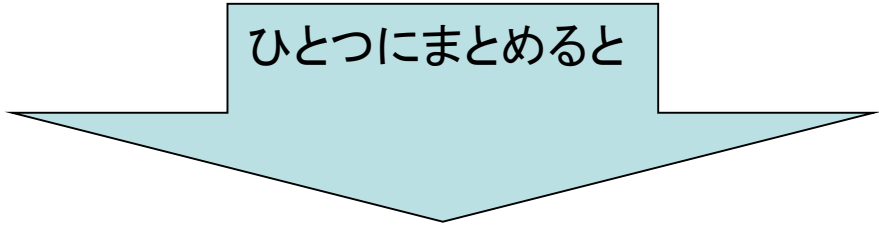
**まとめたクラスがJavaBeansです！**



JavaBeansって何だっけ？

**ひとつにまとめて、何がうれしいと？**

ひとつにまとめると

A large, light blue arrow pointing downwards, with a black outline. The text 'ひとつにまとめると' is centered inside the top rectangular part of the arrow.

**バラバラだった情報が一つのクラス  
にまとまることで、「何の為の情報か？」が  
解りやすくなる！**

JavaBeansって何だっけ？

**例えば・・・**  
**これ、何の情報かわかる？ ↓**

**String myScore;**  
**String enemyScore;**  
**String myFoulNum;**  
**String enemyFoulNum;**

何かのスポーツの試合情報？？サッカーかなあ？？

## JavaBeansって何だっけ？

```
public class Kabaddi implements Serializable{  
    private String myScore;  
    private String enemyScore;  
    private String myFoulNum;  
    private String enemyFoulNum;  
    public String getMyScore(){ return myScore;}  
    public void setMyScore(String myScore){  
        this.myScore = myScore;  
    }  
    ....  
}
```

カバディのスコアを表す情報でした～

JavaBeansって何だっけ？

**データ保持クラスであるJavaBeansには決まりがあります。**

- **情報は全てprivateな変数であること**
- **情報にアクセスする為のgetter、setterを持つこと**
- **Serializableをimplementsすること**

## 目次

- MVCモデルとは？
- JavaBeansって何だった？
- **DBの扱い方！**
- 演習

DBアクセスの書き方！

**重要!**

**Webアプリケーションから  
データベースにアクセスするときには  
必要なものがあります！**

**JDBCドライバ**  
**と言います。**



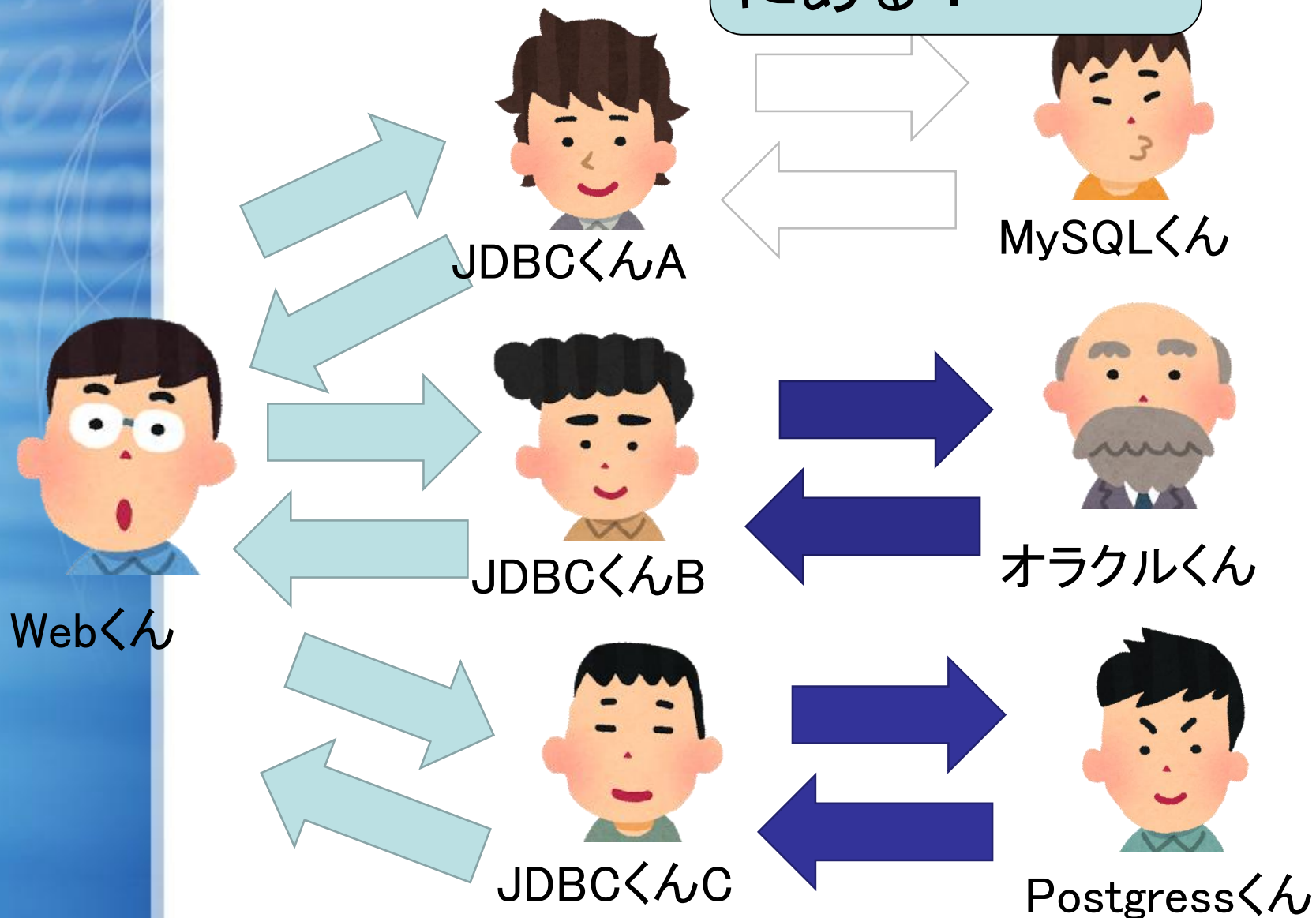
DBアクセスの書き方！

**実はJDBCドライバーは  
DBの種類ごとに存在します！**



DBアクセスの書き方！

JDBCはDB毎  
にある！



## DBアクセスの書き方！

```
public class DBModel {  
    public List<String> getMailList(){  
        List<String> list = new ArrayList<String>();  
  
        Connection con = null;  
        PreparedStatement stmt = null;  
        ResultSet rs = null;  
  
        try{  
            //////////////////////////////////////  
            //DBの接続  
            Class.forName("com.mysql.jdbc.Driver");  
  
            con = DriverManager.getConnection(  
                "jdbc:mysql://localhost:3306/webtestdb","root","password");  
  
            //////////////////////////////////////  
            //SELECT文の発行  
            stmt = con.prepareStatement("SELECT * FROM user_tbl");  
  
            rs = stmt.executeQuery();  
  
            //////////////////////////////////////  
            //DBから値を取得  
            while( rs.next() ){  
                String value = rs.getString("name");  
                list.add(value);  
            }  
  
        }catch(Exception e){  
            System.out.print(e);  
        }  
  
        return list;  
    }  
}
```

## DBアクセスの書き方！

### DBから情報を取得する時の処理手順

1. DBに**接続**する
2. SQL文を**組み立て**る
3. SQLを**発行**する
4. 結果を**取得**する

## DBアクセスの書き方！

# 1. DBに接続する

```
Connection con = null;
PreparedStatement stmt = null;
ResultSet rs = null;

try{
    //////////////////////////////////////
    //DBの接続
    Class.forName("com.mysql.jdbc.Driver");

    con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/webtestdb","root","password");
}
```

JDBCドライバのロード

接続して接続オブジェクト  
(Connectionオブジェクト)を取得する

DBアクセスの書き方！

## DB接続文字列

ユーザ名

パスワード

```
con = DriverManager.getConnection(  
    "jdbc:mysql://localhost:3306/webtestdb", "root", "password");
```

"jdbc:mysql://localhost:3306/webtestdb"

ドライバ名

使用する  
DB種類

接続ホスト:ポート番号  
localhostは同じPC内のDB  
ポート番号はMySQLでは3306

接続する  
DB名

DBアクセスの書き方！

## 2. SQL文を組み立てる

## 3. SQLを発行する

SQLを組み立てる

SQL文を接続オブジェクトに  
セットして、Statementオブジェクトを取得する

```
//SELECT文の発行  
stmt = con.prepareStatement("SELECT * FROM user_tbl");  
rs = stmt.executeQuery();
```

SQLを発行する

executeQueryを実行してSQLを実行する  
execute(実行)Query(SQL文)の意味。  
実行結果はResultSetオブジェクトに返される

## DBアクセスの書き方！

### 4. 結果を取得する

nextメソッドは、結果があればtrueを返し  
無ければfalseを返す

```
//////////  
//DBから値を取得  
while( rs.next() ){  
    String value = rs.getString("name");  
    list.add(value);  
}
```

getStringで値を取得する





DBアクセスの書き方！

# ResultSetオブジェクトのイメージ

## ResultSetオブジェクト

- 1回目のnextで指す行 ➡
- 2回目のnextで指す行 ➡
- 3回目のnextで指す行 ➡
- 4回目のnextで指す行は無いのでfalseを返す

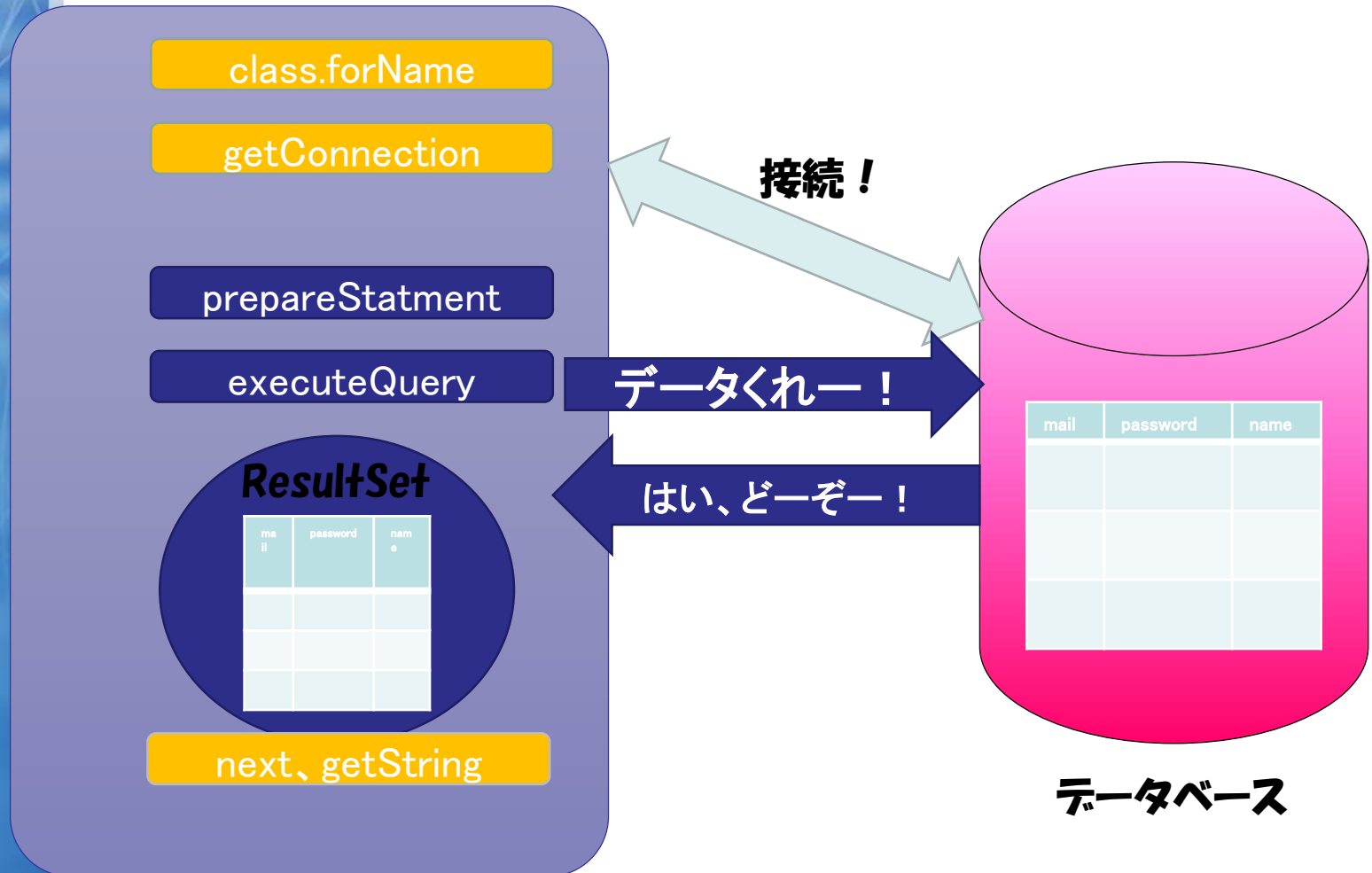
mail	password	name
himeno@asojuku.ac.jp	himeno	姫先生
nishino@asojuku.ac.jp	nishino	西野先生
kitajima@asojuku.ac.jp	kitajima	北島先生

getString("name")  
で  
取得するのはココ

getString("name")で  
取得するのはこの列！

# DBアクセスの書き方！

## 接続の全体イメージ



Webアプリケーション

## DBアクセスの書き方！

### DBから情報を取得する時の処理手順

1. DBに**接続**する
2. SQL文を**組み立て**る
3. SQLを**発行**する
4. 結果を**取得**する

## DBアクセスの書き方！

# 検索はどうする？

Nishino@asojuku.ac.jp

検索



DBアクセスの書き方！

**まず、SQLはどうなる？**

**user\_tblからNishino@asojuku.ac.jp  
を検索するSQLは？**

**SELECT \* FROM user\_tbl WHERE  
mail= 'Nishino@asojuku.ac.jp'**

メアド

検索

検索したい項目は  
WHERE句に指定すればOKだね！

DBアクセスの書き方！

```
SELECT * FROM user_tbl WHERE  
mail= 'Nishino@asojuku.ac.jp'
```

**このSQLをプログラムで書くとどうなる？**



## DBアクセスの書き方！

ポイント1  
SQLの中で変数を埋め込みたい  
ところに「？」を書く

```
////////////////////////////////////  
//SELECT文の発行  
stmt = con.prepareStatement("SELECT * FROM user_tbl WHERE mail=?");  
  
//検索文字列をセットする  
stmt.setString(1, searchString);  
rs = stmt.executeQuery();
```

ポイント2  
setStringを使って「？」に埋め込  
みたい値を指定する





## DBアクセスの書き方！

### 【注意その1】

setStringはセットするDBの型によってメソッドが異なる！

```
//検索文字列をセットする↓  
stmt.setString(1, searchString);↓
```

### 主なセットメソッド

DBの型	セットメソッド
varchar	setString
int	setInt
TimeStamp	setTimeStamp

他にも色々あるよ

## DBアクセスの書き方！

### 【注意その2】

複数の検索値がある場合は、その数分「？」が増える  
また、「？」の数とsetStringなどの数は同じになる！

```
//////////////////////////////////////////  
//SELECT文の発行  
stmt = con.prepareStatement("SELECT * FROM user_tbl WHERE mail=? AND name=?");  
  
//検索文字列をセットする  
stmt.setString(1, searchString);  
stmt.setString(2, name);  
rs = stmt.executeQuery();
```

値のセットメソッドの仕様は  
setString(何番目の？が対象か，埋め込みたい値)  
です。左から数えて何番目の？かを指定します。

DBアクセスの書き方！

なるほど、..  
検索条件は「？」で書けばいいのね？  
でも……

全体のプログラムの流れって  
どうだっけ？？？？



## DBアクセスの書き方！

# 処理の流れを考える

メアド

検索

1. メールアドレスを入力する
2. メールアドレスをサーブレットに送信する
3. サーブレットがメールアドレスを取得する
4. DBに接続しメールアドレスを検索する
5. 検索結果を受け取り画面遷移する
6. 結果を画面に表示する

## それぞれの処理はMVCのどこに書けばよい？

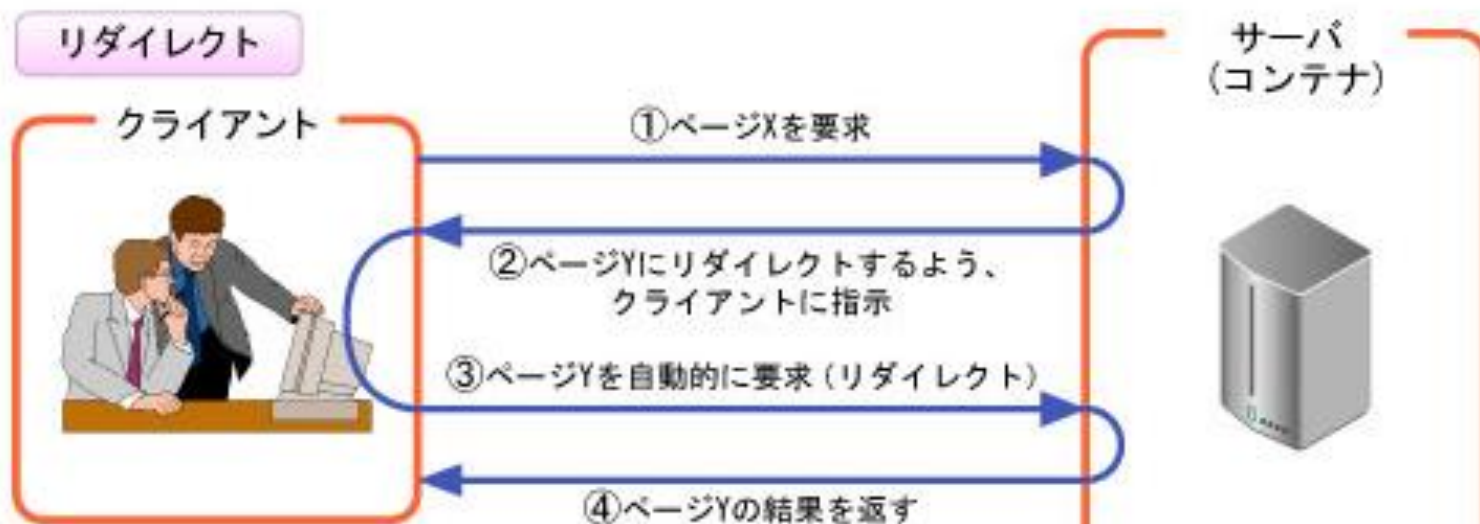
## DBアクセスの書き方！

MVCは、Vで入力したデータをCでチェック、加工や準備をしてMで処理し、Vで結果を表示する

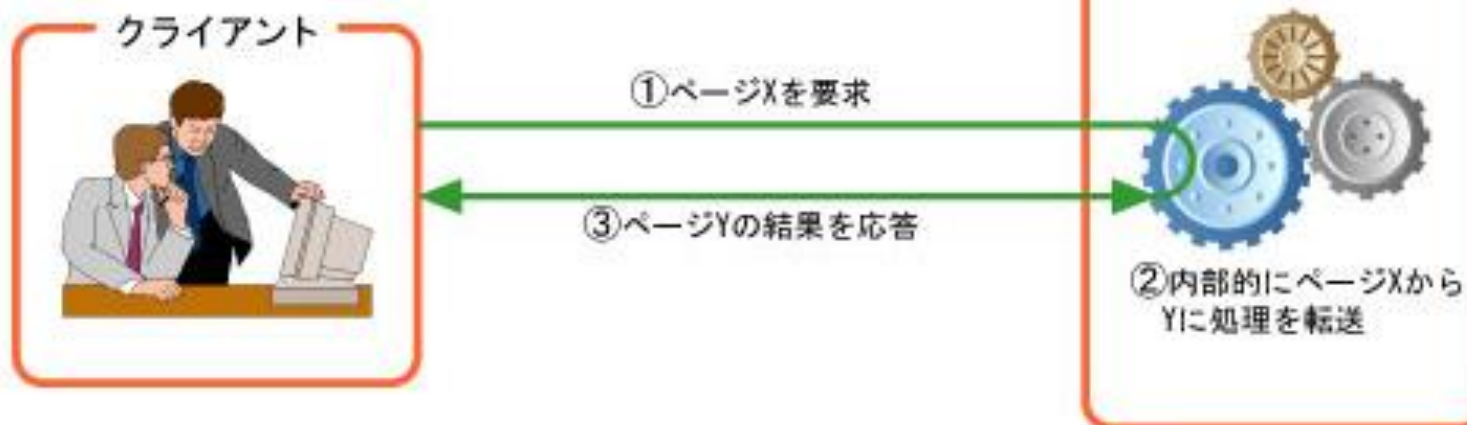
No.	処理内容	MVC
1	メールアドレスを入力する	V
2	メールアドレスをサーバーレットに送信する	V
3	サーバーレットがメールアドレスを取得する	C
4	DBに接続しメールアドレスを検索する	M
5	検索結果を受け取り画面遷移する	C
6	結果を画面に表示する	V

# リダイレクト？

## リダイレクト



## フォワード





リダイレクト？

**もうひとつ**

**リダイレクトだと、URLが変わります。  
フォワードは変わりません。**

リダイレクト？

## 使い分けはどうする？

方法	用途
forward (フォワード)	サーバー内の転送(JSPを表示するなど)
redirect (リダイレクト)	サーバー外への画面遷移やURLを変更させたいとき



## 演習

# 今日の内容を含んだ演習ですっ！

おねえ座りを成敗！





Webのしくみを思い出せ

**<https://github.com/nishino-naoyuki/2019Web/課題サーバレット復習3.pdf>**

**課題はメールでnishino@asojuku.ac.jpに送信**