サーブレット課題７解説プリント

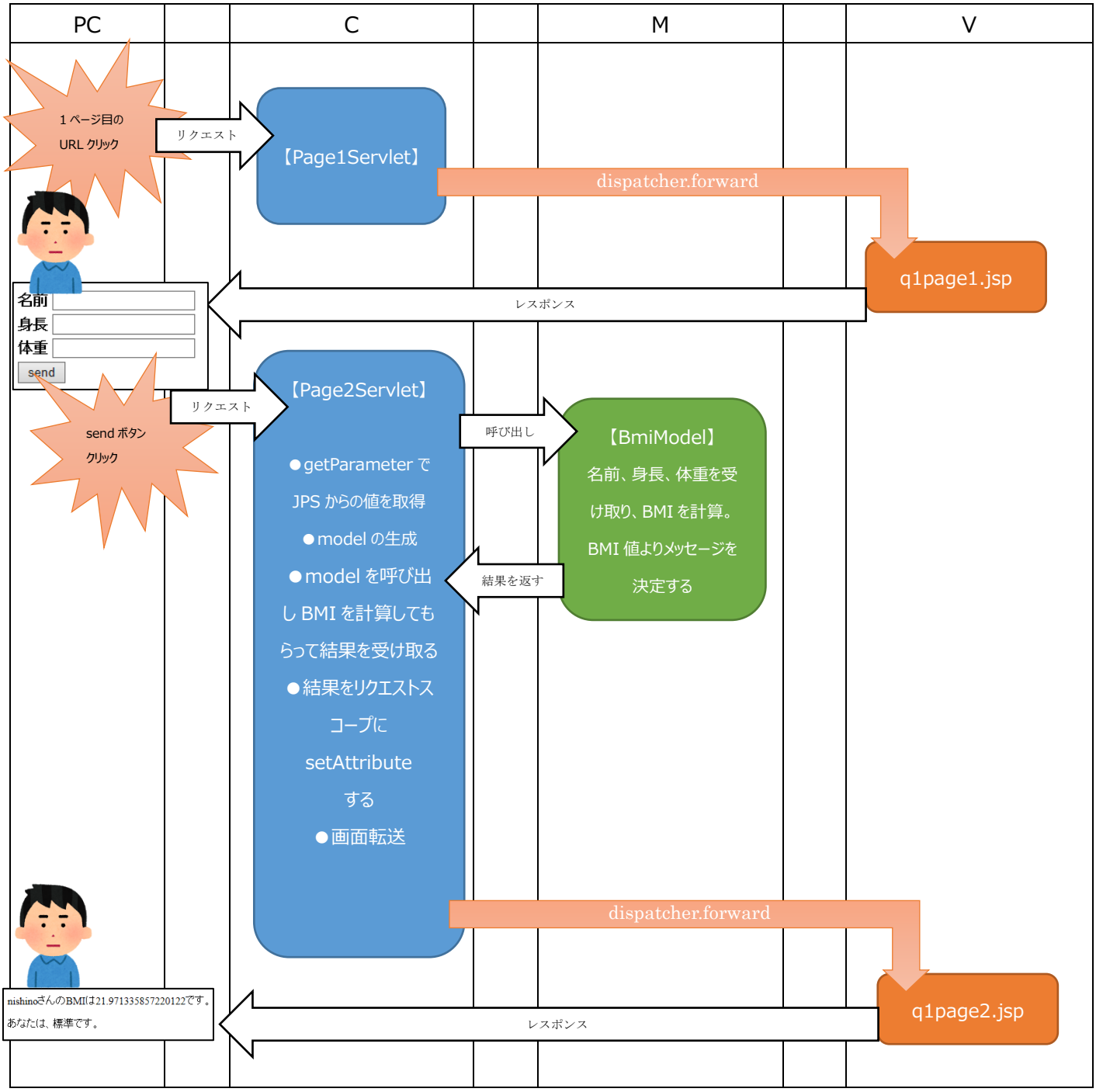
Step1．まずは、処理の流れを理解しよう！

MVC モデルでは、以下の役割分担となります。

M（モデル）	計算や複雑な処理を行う。 基本的には何も継承しないベタな Java
V（ビュー）	結果を表示する画面のこと。あまり複雑な Java の処理は書かない。JSP。
C（コントローラ）	処理の旗振りを行う。具体的にはリクエストを受け付けて、モデルを呼び出し、処理結果を取得して、次の画面を表示する。サーブレットのこと。

※このように役割分担を明確にすることで、メンテナンス性（仕様変更に強い）プログラムが作成できる

サーブレット課題７の流れは以下の通り



Page2Servlet.java と BmiModel について詳しく解説

●Page2Servlet.java

```java
@WebServlet("/q1page2")
public class Page2Servlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        ////////////////////////////
        //パラメータの取得
        String name = request.getParameter("name");
        String weight = request.getParameter("weight");
        String tall = request.getParameter("tall");

        ////////////////////////////
        //文字列→数値変換
        BmiBeans bmiBeans = new BmiBeans();
        try{
            bmiBeans.setName(name);
            bmiBeans.setWeight(Integer.parseInt(weight));
            bmiBeans.setTall(Integer.parseInt(tall));
        }catch(Exception e){
            //エラー発生
            request.setAttribute("error", "数値を入力してください");
            RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/jsp/q1page1.jsp");
            dispatcher.forward(request, response);
            return;
        }

        ////////////////////////////
        //モデル呼び出し
        BmiModel bmiModel = new BmiModel();
        bmiBeans = bmiModel.calculate(bmiBeans);

        ////////////////////////////
        //リクエストスコープにセット
        request.setAttribute("bmiBeans", bmiBeans);

        ////////////////////////////
        //転送
        RequestDispatcher dispatcher = request.getRequestDispatcher("WEB-INF/jsp/q1page2.jsp");
        dispatcher.forward(request, response);
    }
}
```

getParameter で値を取得

BmiBeans のインスタンスを作る

BmiBeans に setXXX(セッター) を使って値をセットする

parseInt があるので変換できない場合を考慮して try-cach しています

モデルのインスタンスを作成して計算するメソッドを呼び出しています

計算結果をリクエストスコープにセットしています

2 ページ目の JSP に処理を転送しています

●BmiModel.java

```java
/**
 * BMIを計算する
 *
 * @param bmiBeans
 * @return
 */
public BmiBeans calculate(BmiBeans bmiBeans){

    //BMIを計算
    double wk = bmiBeans.getTall()/100.0;
    double bmi = (double)bmiBeans.getWeight()/(double)(wk*wk);

    //メッセージを取得
    String message = "";

    if( bmi < 16.0 ){
        message = "痩せすぎ";
    }else if( bmi < 17.0 ){
        message = "痩せ";
    }else if( bmi < 18.5 ){
        message = "痩せぎみ";
    }else if( bmi < 25.0 ){
        message = "標準";
    }else if( bmi < 30.0 ){
        message = "過体重";
    }else if( bmi < 35.0 ){
        message = "肥満（1度）";
    }else if( bmi < 40.0 ){
        message = "肥満（2度）";
    }else{
        message = "肥満（3度）";
    }

    //ビーンにセット
    bmiBeans.setBmi(bmi);
    bmiBeans.setMessage(message);

    return bmiBeans;
}
```

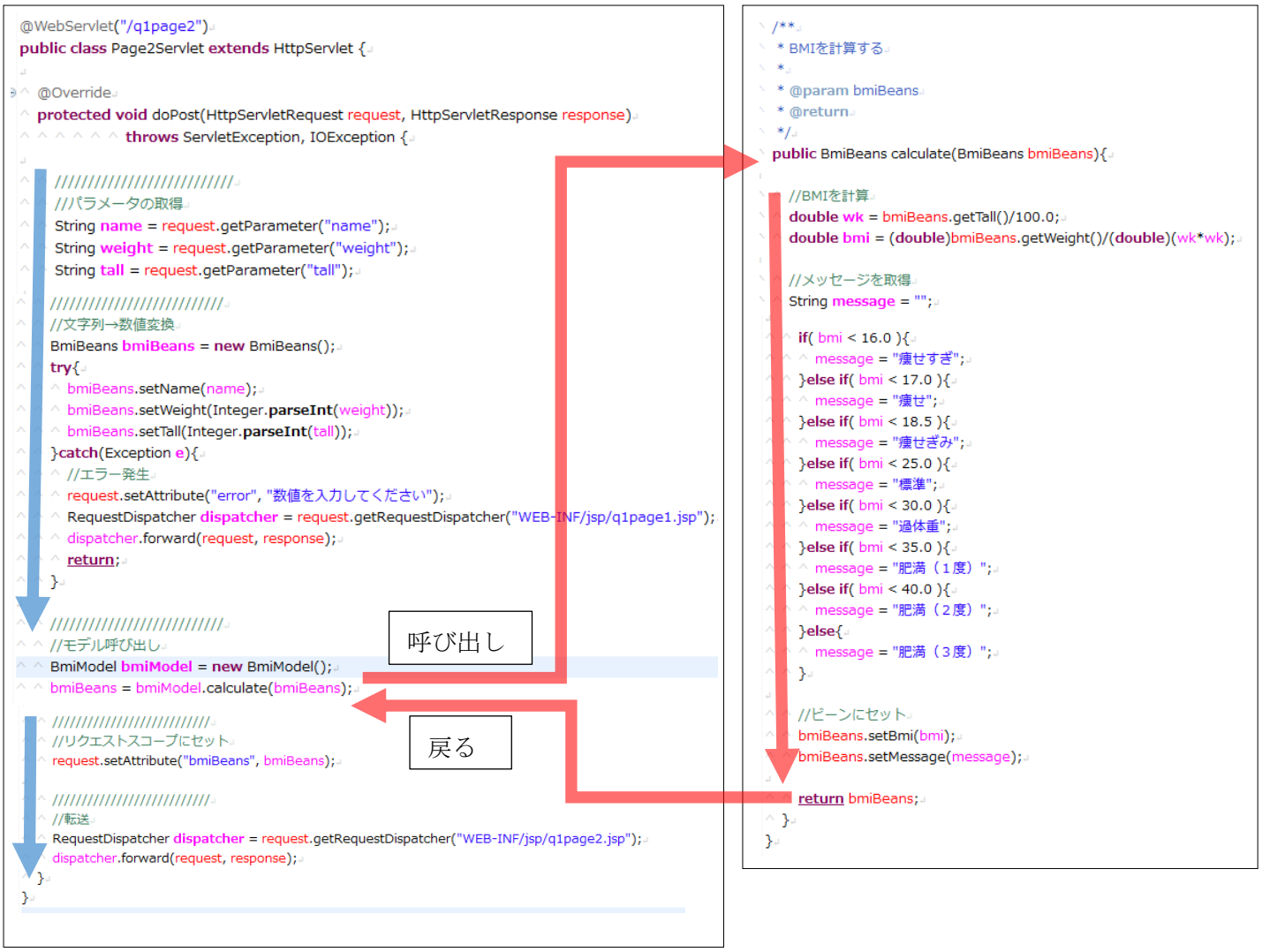
getXXX（ゲッター）で値を取り出し BMI を計算しています

BMI の値からメッセージを決定します

ビーンズに BMI 値とメッセージをセットします

ビーンズの値を返します

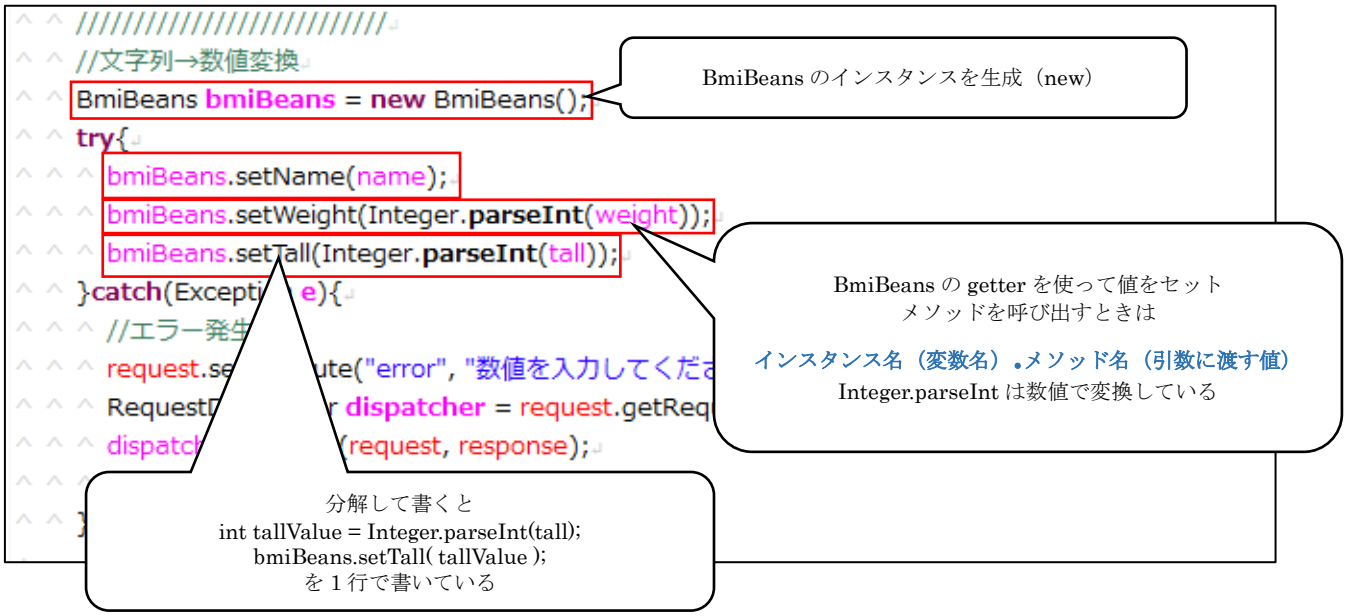
Step2.プログラムの流れを抑えよう。



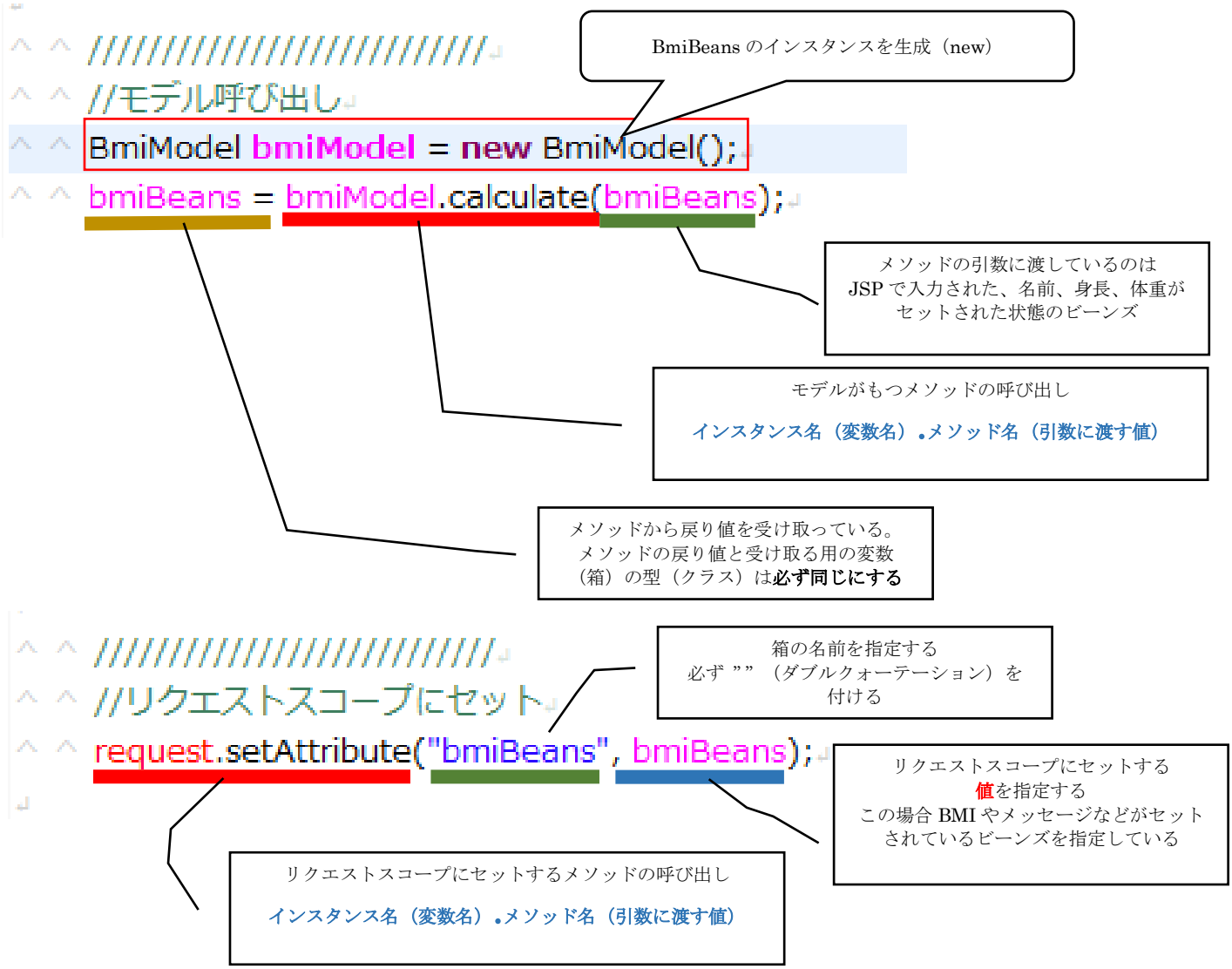
Step3.プログラムの細かいところを理解しよう

処理をピックアップ

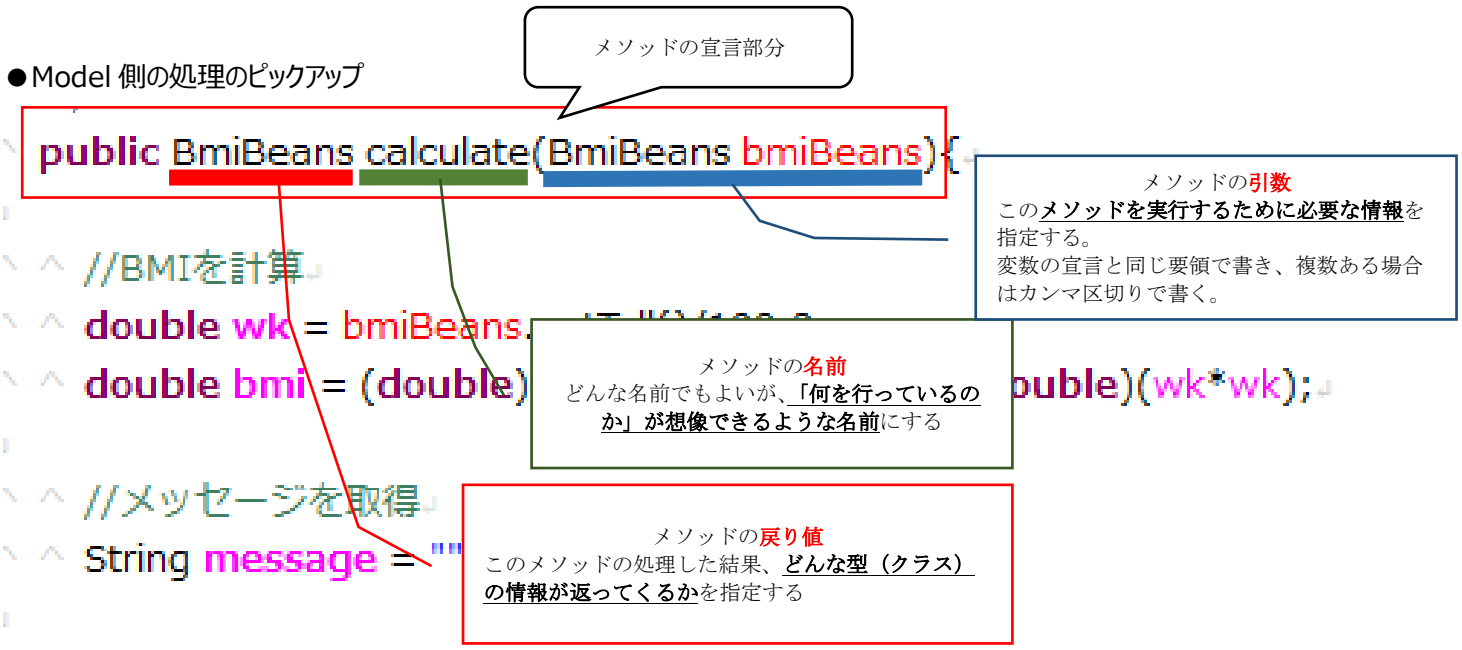
●ビーンズに値をセットする部分



●Model の呼び出し



●Model 側の処理のピックアップ



【ポイント１】
今回のモデルの場合、「BMI の計算と表示するメッセージを決定したい」という目的がある。
そのために、BMI を扱うための BmiModel クラスを作り、BMI を計算するための calculate メソッドを作った。

【ポイント２】
BMI を計算するための calculate メソッドを作るときに、引数を考える。
計算に必要な情報は「体重と身長」なので、その２つを引数に取るメソッドを作っても OK だが、今回はせっかく、体重と身長がセットされているビーンズが存在するのでビーンズを引数として渡すようにした。

【ポイント３】
BMI を計算するための calculate メソッドを作るときに、戻り値を考える。
このメソッドで返したい値は「BMI の計算値」と「BMI に沿ったメッセージ」の２つ。
戻り値では１つの値しか返せないので、ビーンズにどちらの値もセットして、ビーンズを返すことで複数の値を返すことを実現している

Step4.MVC モデルについて再度考えてみよう

モデル（M）とは・・・メインの処理を書くところ。今回で言うと「BMI の計算」という**一番おいしい（難しい）部分**を担当する

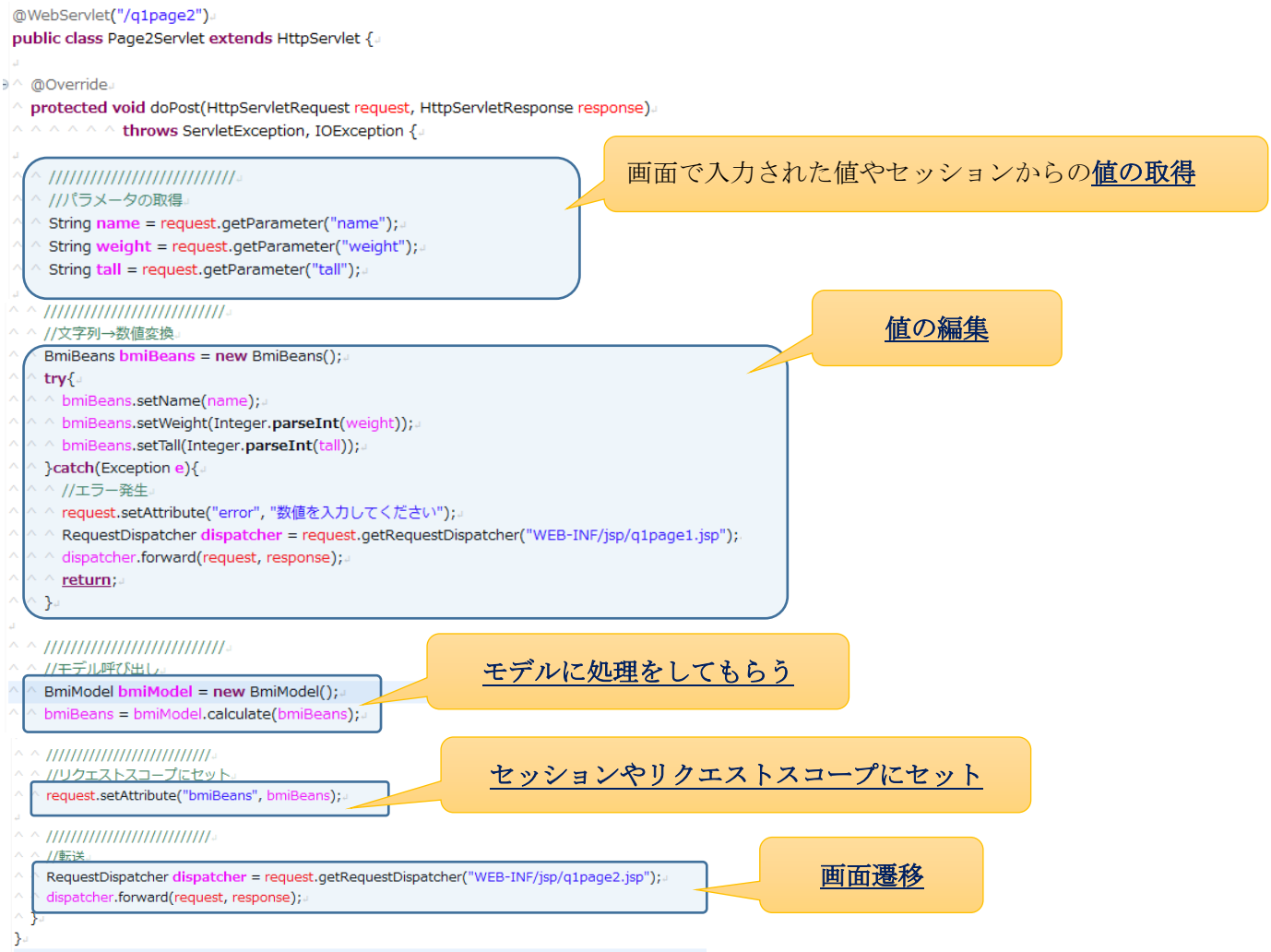
ビュー（V）とは・・・モデルが計算した**結果を表示する**担当。もしくは処理に**必要な情報を入力させる**担当。

コントローラ（C）とは・・・モデルとビューの橋渡し役。モデル（メインの処理）に情報を渡すためにビューからの情報を取得したり（getParameter）、加工したり（文字列→数値変換）して、モデルに渡す。
さらにモデルに処理してもらった情報をビューに渡すために情報をセット（setAttribute）する。
また、次にどの画面を表示すればよいかを決定して、JSP に処理を転送（dispatcher.forward）する。

コントローラの処理は以下のような処理になるパターンが多いです。

1．画面で入力された値やセッションからの値の取得
2．値の編集（文字列→数値変換したり、ビーンズに格納したり）
3．取得した値を渡してモデルに処理をしてもらう（モデルのメソッド呼び出し）
4．モデルの処理結果を戻り値で受け取って、その値をセッションやリクエストスコープにセットする
5．画面遷移する

今回の場合・・・・



ビーンズとは・・・**複数の情報をひとまとめにしたクラス**。**ただのクラス**。怖くない。

ひとまとめにして扱うと分かりやすいというだけの意味で存在する

通常、ビーンズは以下の特徴がある。

1．Serializable を implements している
2．変数が全て private
3．getter（ゲッター）、setter（セッター）を持つ

教科書にはいろいろ書いてあるが、この３つは絶対覚えておこう、

【脱線話】

・Java のコーディング規約について

Java に限らず、プログラミング言語には「コーディング規約」というものがあります。

会社や現場、プロジェクトごとに細かい部分が違っていたりしますが、標準的なコーディング規約や↓です。

https://future-architect.github.io/coding-standards/documents/forJava/

様々な規約がありますが、以下は最低限覚えておきましょう。

項目	説明	例
クラス名について	クラス名は**大文字で始め**、文節を大文字にする	○ LoginServlet × login_servlet
変数名について	変数名は**小文字で始め**文節を大文字にする ループカウンタ以外での 1 文字変数は避ける	○ bombPostion × BombPosition
メソッド名について	メソッド名は小文字で始め、文節を大文字にする 動詞＋名詞 の順で名前を付ける（動詞だけでも OK）	○ getName × GetName × nameGet
	動詞はプロジェクト内で同じ意味のものは同じになるようにする。 例えば、「検索する」メソッドがプロジェクト内で「find」や「search」が混在しないようにする	○ 検索処理のメソッドで find で統一されている × 検索処理のメソッドが find、search 混在する
	Boolean、boolean を返すメソッドは is で始める	○ public boolean isCorrectParam() × public boolean getParamCorrect()

ネストについて	{ ごとにネストを 1 段下げる。
	{ } 内は同じネストにする
		○
		if(value == 0){
		value = 1;
		System.out.println(“val=”+value);
		}
		
		×
		if(value == 0){
		value = 1;
		System.out.println(“val=”+value);
		}

ここから先は私（西野）の主観も入りますが、「見やすいプログラム」を書くために守ってほしいことを書きます。
1 段レベルアップしたプログラムを組みたい人は、是非取り組んでください。

マジックナンバーを避ける	マジックナンバーとは、プログラム中にベタに書かれた数字のこと。プログラムを作った人には、その数字の意味がわかるが、ほかの人にはわからない。
	また、仕様変更があったときに同じ意味を持つ数値の個所の変更を効率よく行えるメリットもある
		×
		for(int i =0; i<9; i++){
		・・・
		}
		
		○
		private final int MAX_INING = 9;
		for(int i =0; i< MAX_INING; i++){
		・・・
		}

ネストが 4 段以上になる場合は、メソッドに分けることを検討する	for や if が連続するとネストが深くなる（字下げの改装が深くなる）ことがあります。
	３段まではいいですが、４段以上になる場合はプログラムが見にくくなることが多いので、なるべく避けましょう
		×
		try{
		for(int i=0; i<9;i++){
		if(i == 0){
		for(int j=0; j<10;j++){
		System.out.print(“”);
		}
		}
		}
		catch(Exception e){
		・・・・
		}
		上記の場合、ネストが深すぎて読みづらいので
		if 文以降をメソッドにできないかなど、ネストを深くしないで済む方法を考える

メソッドが長いときは別メソッドに分ける	1 メソッドのコーディング量が長いとき、具体的にはメソッドの始まりから終わりまでを見るのにスクロールが必要なくらい（だいたい 100 行くらい） 長いときは、メソッドを分けることを検討する	
インスタンス名とメソッド名のコンボで何をしているかが分かるような名前付けを心がける	「究極の良いプログラム」は「コメントが無くても何をしているかわかるプログラム」だと思います。 そのためにも、クラス名、メソッド名は重要です メソッドを呼び出すときは インスタンス名.メソッド名 になるので、インスタンス名とメソッド名を続けてみると何をしたいのかが分かるようにするとよい	例１） MemberModel member = new MemberModel(); member.register(memberDto); こう書くと member を register していると直感的に わかるので「メンバーの登録をしている」というのがコメントが無くて

	です。
	難しい条件の if 文も条件部分をメソッド化すると意外とすっきりしたりします
		もわかる。
		
		例２）
		String movieName = req.getParameter(“name”);
		MovieModel movie = new MovieModel();
		movieList = movie.searchBy(movieName);
		
		こう書くと、コメントが無くても映画の名前をもとに映画の情報を検索してそうだなというのが直感的にわかる
		
		例３）
		public void getStudentInfo(int garade){
		  if(isTargetGrade(grade)){
		  ....
		  }
		}
		
		private boolean isTargetGrade(int grade){
		  return (0<grade && grade<3 && grade==1);
		}
		
		こう書くと単純に条件を and でつないで書くより、対象の学年かどうかを判定しているのだなと直感的にわかる