pi-15. カプセル化，MVCモデル，オブジェクトのマッピング

トピックス：カプセル化，MVC モデル，MVC モデルの応用，オブジェクトのマッピング

URL: https://www.kkaneko.jp/cc/pi/index.html

（**Java の基本**）

金子邦彦

今回の内容

- **カプセル化**

Java では、public, private の指定により，属性やメソッドへのアクセス制御を行う

- **MVC**

「モデル」、「ビュー」、「コントローラー」のこと

- **マッピング**

Java のオブジェクトを、データベースや DOM オブジェクトにマッピングできる

アウトライン

番号	項目
	復習
15-1	カプセル化
15-2	MVC モデル
15-3	Java での MVC モデル
15-4	MVC モデルの応用
15-5	オブジェクトのマッピング
15-6	Java でのオブジェクトのマッピング

各自、資料を読み返したり、課題に取り組んだりも行う

この授業では、**Java** を用いて基礎を学び、マスターする

Run Debug Stop Share Save { } Beautify

Main.java

```java
public class Main
{
    public static void main(String[] args) {
        int x = 100;
        int y = 200;
        System.out.printf("%d\n", x + y);
    }
}
```

input

```
300


...Program finished with exit code 0
Press ENTER to exit console.
```

Java などのプログラミング言語の体験，演習ができる**オンラインサービス**

GDB online

http://www.pythontutor.com/

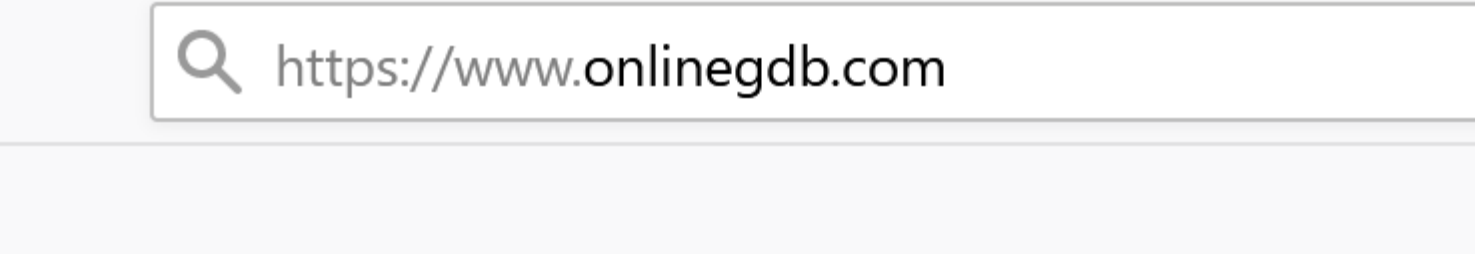
オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

GDB online で Java を動かす手順

① ウェブブラウザを起動する

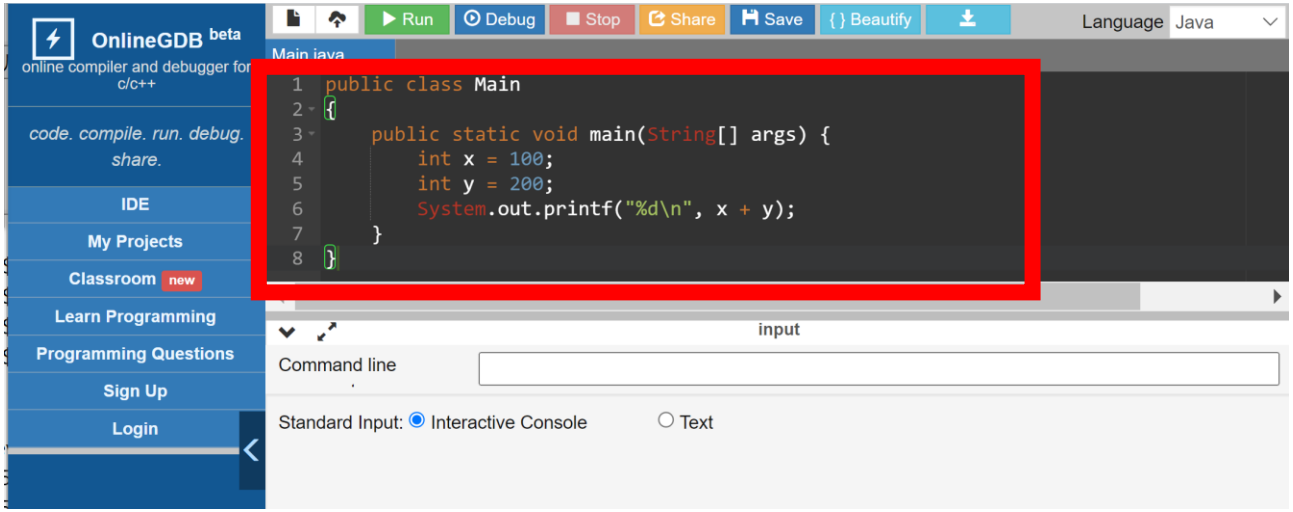
② 次の URL を開く

https://www.onlinegdb.com



③ 「Language」のところで，「Java」を選ぶ

④ ソースコードを入れる



⑤ 実行．実行結果を確認

「**Run**」をクリック．

15-1. カプセル化

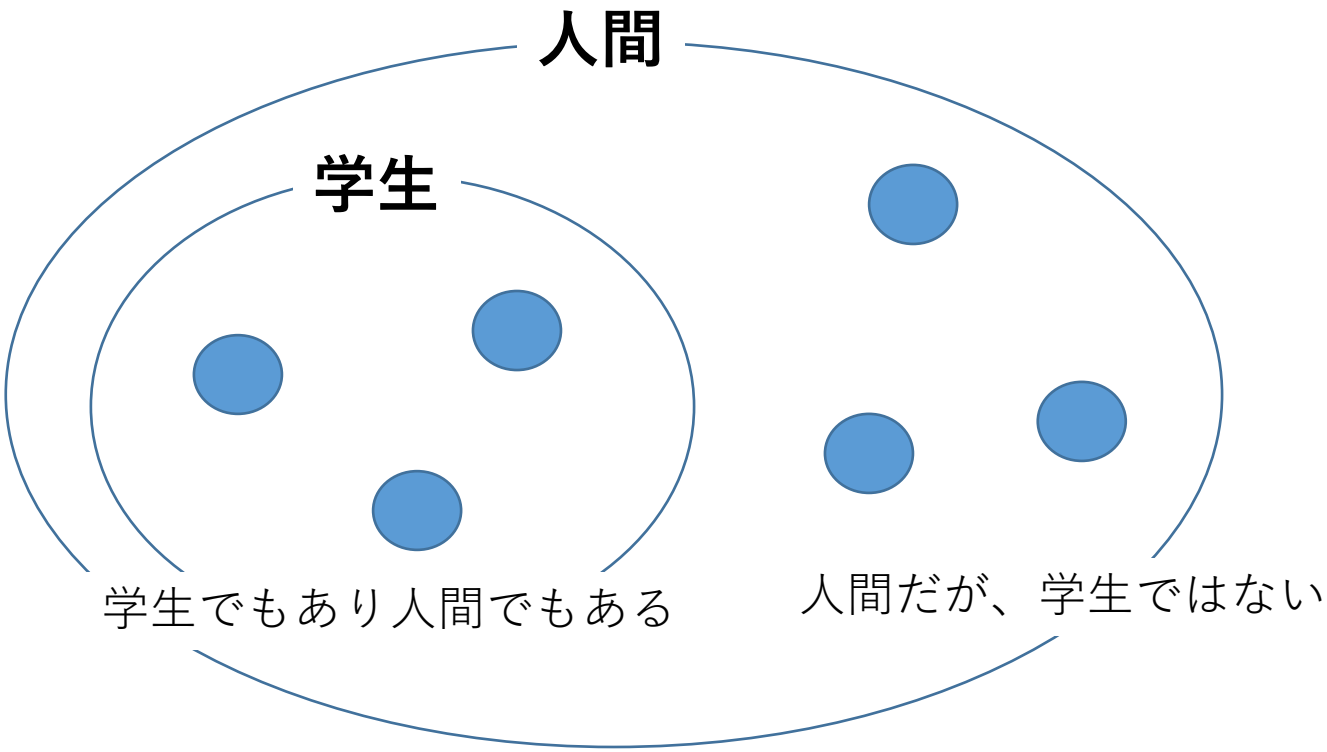
カプセル化とは

- **オブジェクト**は、**属性**と、**メソッド**を持つ

- 必要な**属性**と**メソッド**のみ、他のオブジェクトに**公開**する

クラス

- **クラス**は，**同じ種類のオブジェクトの集まり**と考えることができる

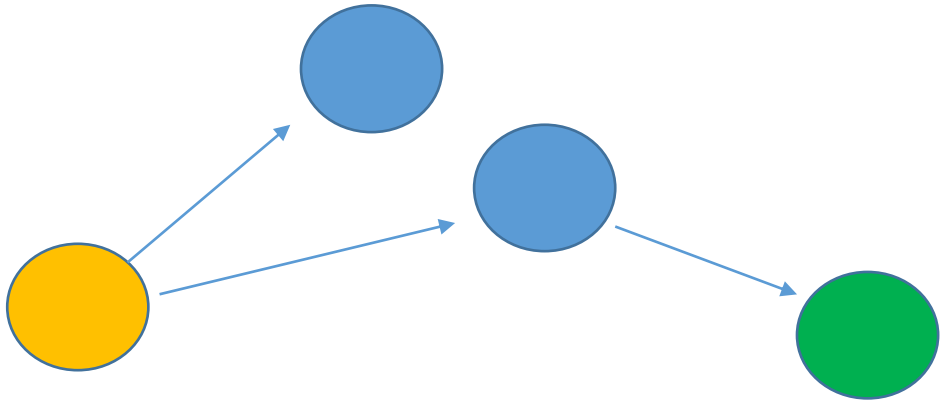


ソースコード

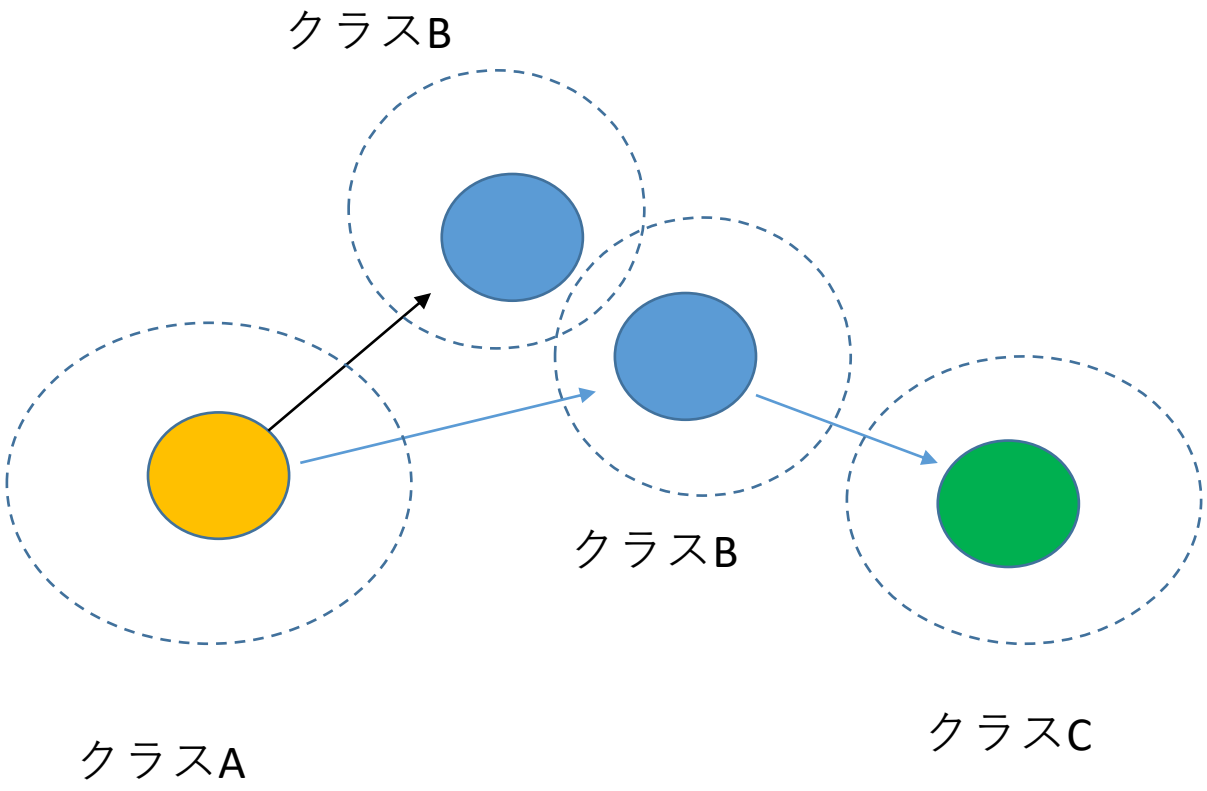
・クラス定義

・オブジェクト生成など

プログラムの起動



オブジェクトが生成され、互いに連携しながら動作する



すべての**オブジェクト**は**カプセル化**されている。

- 必要な**属性**と**メソッド**のみ、他のオブジェクトに**公開**する
- **<u>何を公開し、何を公開しないか</u>**は、クラス単位で**<u>指定可能</u>**

Java でのカプセル化

- 公開　　**public**
- 非公開　　**private**

ain.java

```java
import java.util.*;

class Circle {
    double x;
    double y;
    double r;
    String color;
    public Circle(double x, double y, double r, String color) {
        this.x = x;
        this.y = y;
        this.r = r;
        this.color = color;
    }
    public void printout() {
        System.out.printf("%f %f %f %s\n", this.x, this.y, this.r, this.color);
    }
}
public class Main {
    public static void main(String[] args) throws Exception {
        Circle x = new Circle(2, 4, 3, "green");
        Circle y = new Circle(8, 10, 1, "blue");
        x.printout();
        y.printout();
    }
}
```

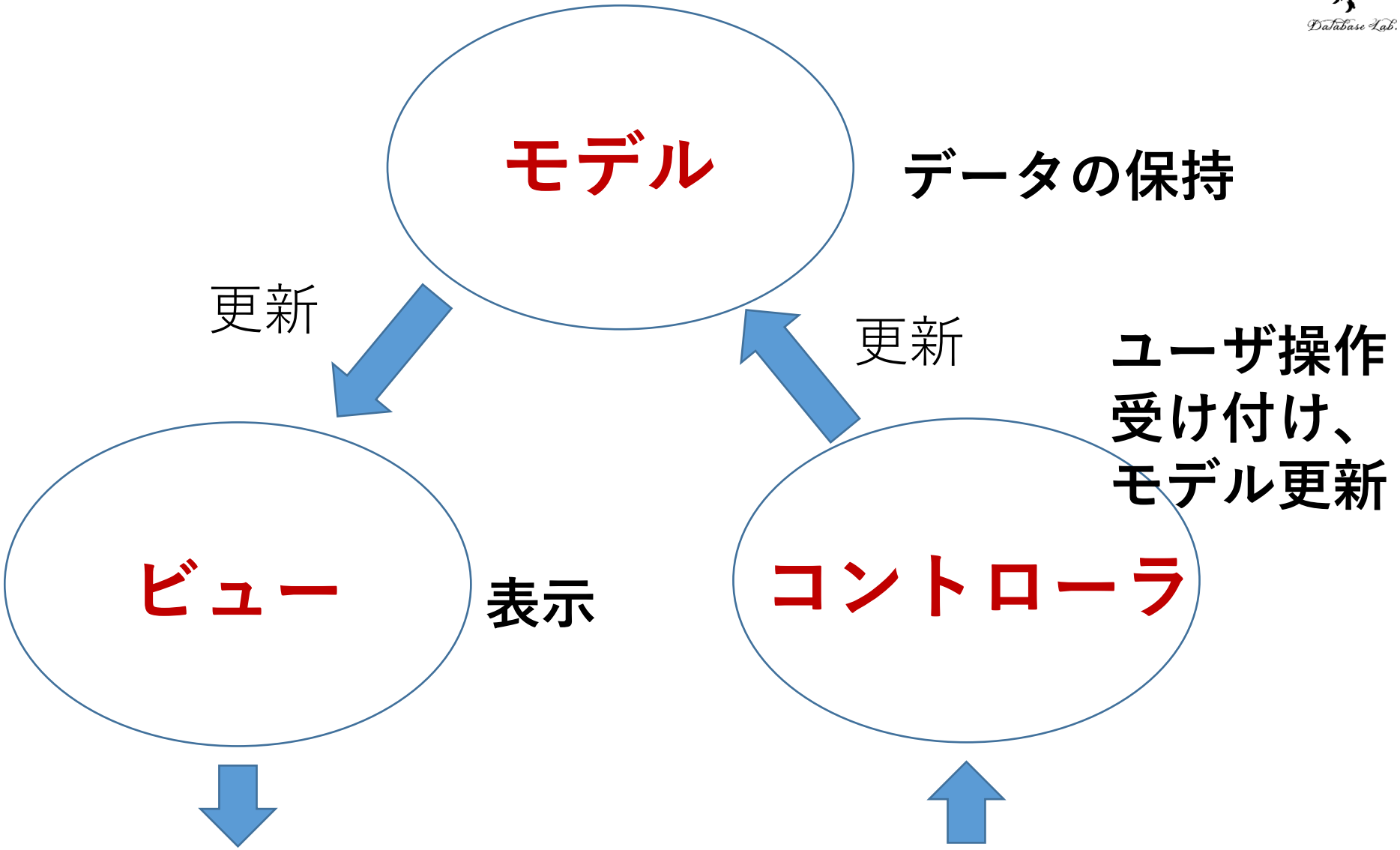
まとめ

- **カプセル化**

Java では、public, private の指定により，属性やメソッドへのアクセス制御を行う

15-2. MVC モデル

モデルとビューとコントローラ



モデルとビューとコントローラ

- **オブジェクト指向**では、
  アプリ（Webアプリなど）は、
  さまざまな**オブジェクトの集まり**

- **モデル、ビュー、コントローラーに分けて、プログラムを設計、製作、テストすることで、プログラムを見通し良く作成可能**　（私の見解）

MVC の例

名簿を作るとき

モデルの例

- **Person** クラス：　オブジェクトは 1 人の人間
- **Meibo** クラス：　オブジェクトは名簿全体

ビューの例

表やフォーム形式で、名簿データを表示

コントローラーの例

フォーム記入内容をもとに、モデルを更新

15-3. Java での MVC モデル

今から行うこと

- モデル部分： Person クラス，Meibo クラス
- コントローラー部分：　Main クラス

※ ビューは考えないことにする

演習

資料：**22 ～ 29**

【トピックス】

- **MVC モデル**

① プログラム （**モデル部分**）

```
1  import java.util.HashMap;
2  import java.util.Iterator;
3  import java.util.ArrayList;
4  class Person {
5    String name;
6    String address;
7    public Person(String name, String address) {
8      this.name = name;
9      this.address = address;
10   }
11   public void printout() {
12     System.out.printf("%s %s\n", this.name, this.address);
13   }
14 }
```

プログラムの続き（**モデル部分**）

```
15 class Meibo {
16   HashMap<Integer, Person> m;
17   public Meibo(HashMap<Integer, Person> m) {
18     this.m = m;
19   }
20   public void add(int id, String name, String address) {
21     m.put(id, new Person(name, address));
22   }
23   public void printout() {
24     for(Integer i : this.m.keySet()) {
25       System.out.printf("%d, ", i);
26       this.m.get(i).printout();
27     }
28   }
29 }
```

プログラムの続き（**コントローラー**部分）

```
30 public class Main {
31   public static void main(String[] args) throws Exception {
32     HashMap<Integer, Person> m = new HashMap<Integer, Person>();
33     Meibo a = new Meibo(m);
34     a.add(1, "XX", "Fukuyama");
35     a.add(2, "YY", "Okayama");
36     a.printout();
37   }
38 }
```

④ 実行し結果を確認

```
1, XX Fukuyama
2, YY Okayama



...Program finished with exit code 0
Press ENTER to exit console.
```

次に行うこと

- モデル部分： Person クラス，Meibo クラス
- コントローラー部分：　Main クラス
- **ビュー部分：　View クラス（新しく追加）**

次のプログラムを**書き加える**

```
30 class View {
31   ArrayList<Person> v;
32   public View() {
33   }
34   public void update(Meibo meibo) {
35     this.v = new ArrayList<Person>();
36     for(Integer i: meibo.m.keySet()) {
37       v.add(meibo.m.get(i));
38     }
39   }
40   public void printout() {
41     for(Person p: this.v) {
42       System.out.printf("%s %s\n", p.name, p.address);
43     }
44   }
45 }
46 public class Main {
47   public static void main(String[] args) throws Exception {
```

次のように書き換える（ビューを使うように）

```
46 public class Main {
47   public static void main(String[] args) throws Exception {
48     HashMap<Integer, Person> m = new HashMap<Integer, Person>();
49     Meibo a = new Meibo(m);
50     a.add(1, "XX", "Fukuyama");
51     a.add(2, "YY", "Okayama");
52     View v = new View();
53     v.update(a);
54     v.printout();
55   }
56 }
```

実行し結果を確認

```
XX Fukuyama
YY Okayama



...Program finished with exit code 0
Press ENTER to exit console.
```

モデルとビューの分離

- モデルの中の**データ**を，**全て見せる必要がない**（すべてを見せたくない）**場合などに有効となる考え方**

- 「プログラムが作成しやすくなる」（私の見解）

  例）「『表示にこだわりたい』という場合，
  モデルとビューを分離しておけば，
  ビューのプログラムに集中できる」
  という考え方も

15-4. MVC モデルの応用

- **フレームワーク**：アプリケーションの土台となるソフトウエア

- MVC モデルに適する Java 言語フレームワークも多数ある

  Struts

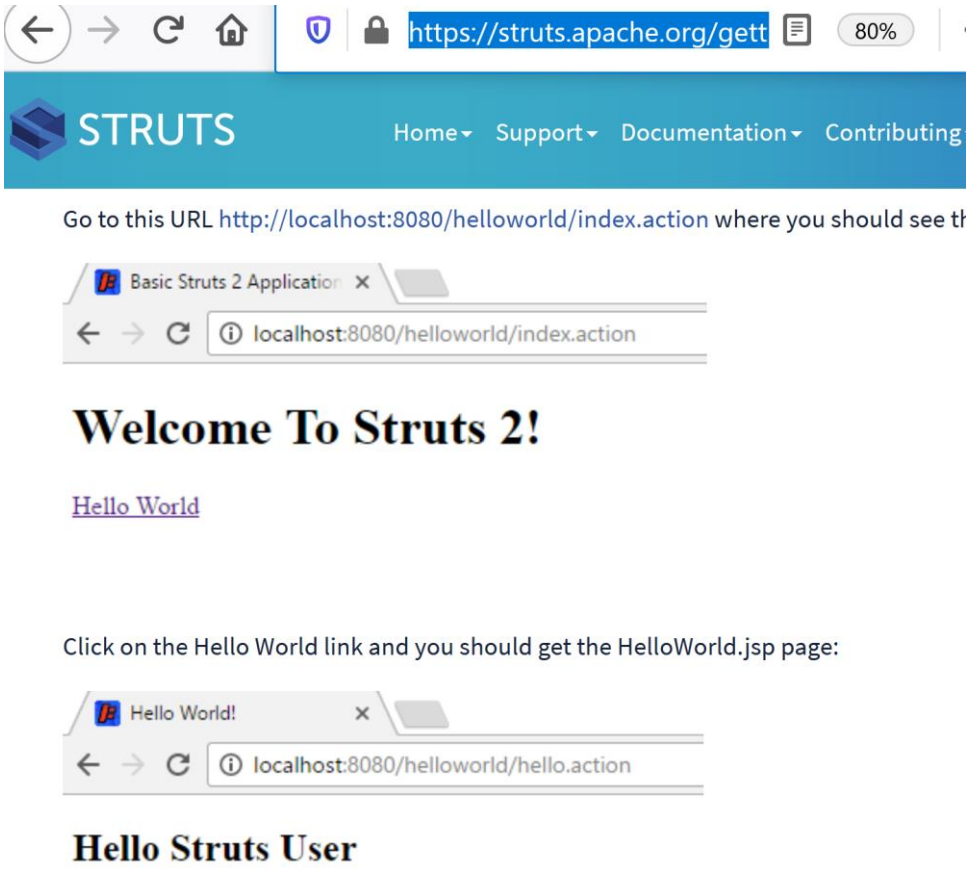
  JSF (Java Server Faces)

  Spring Framework

Java の標準機能外であるが、インストールは簡単

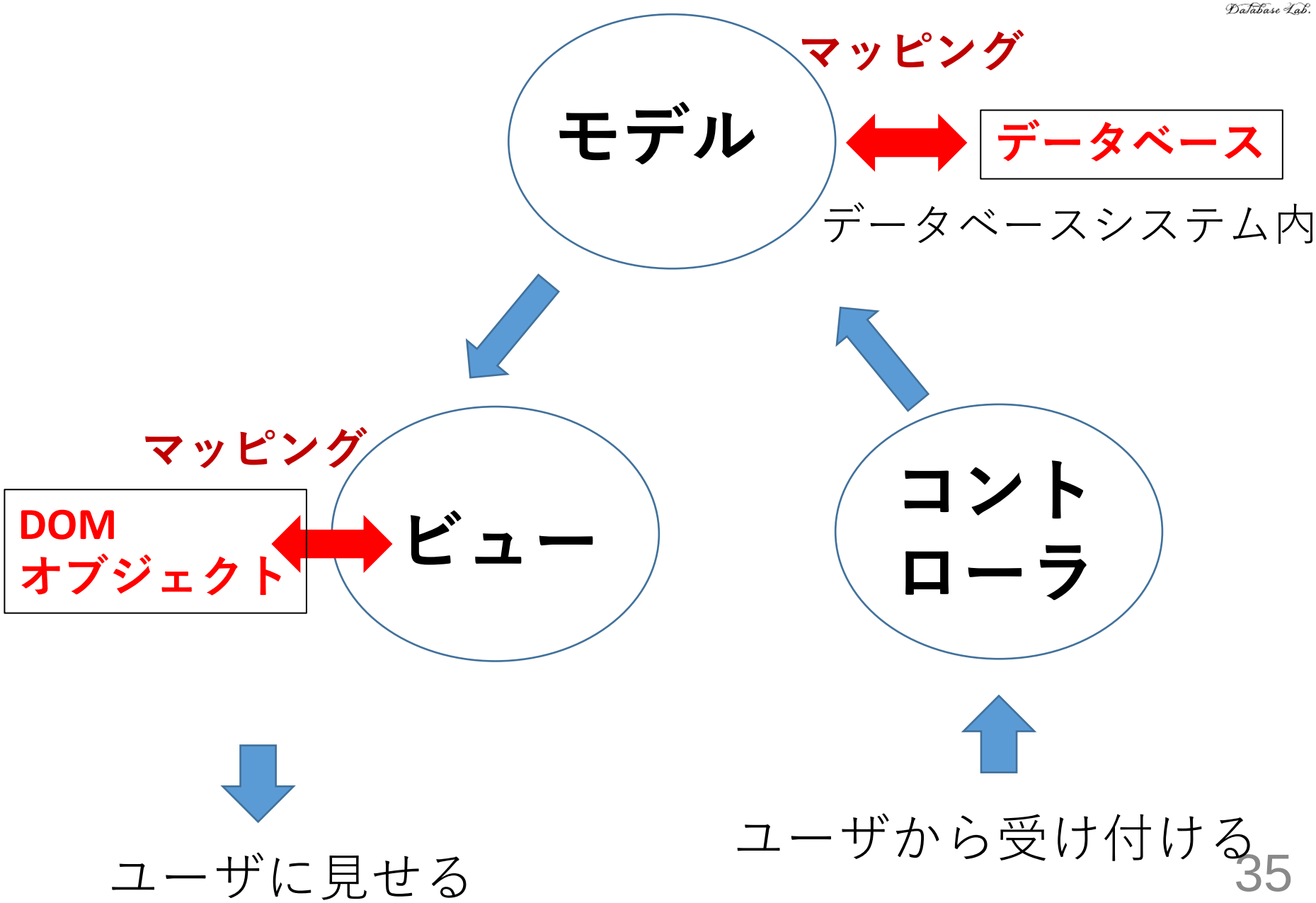
Struts 2

https://struts.apache.org/getting-started/hello-world-using-struts2.html

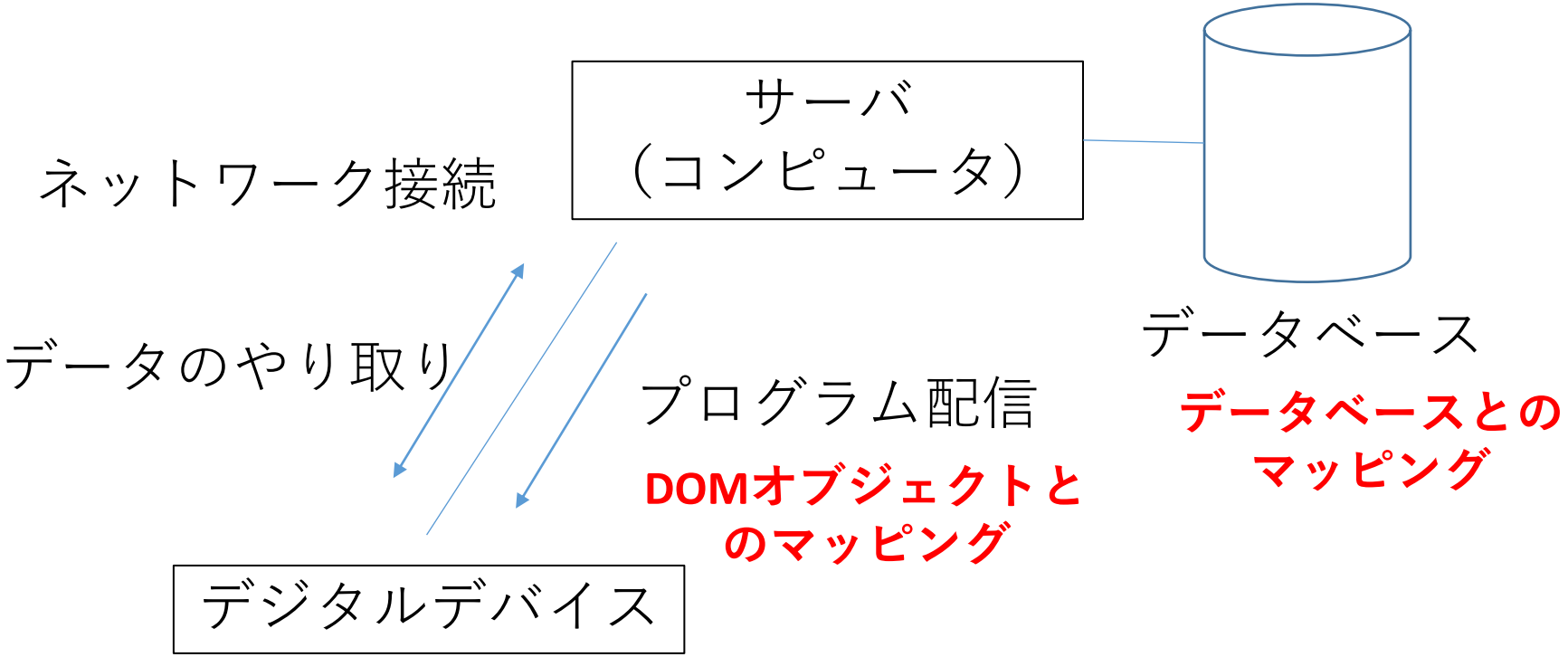


- Web アプリケーションのワーク
- HTML, Java の組み合わせでアプリケーションを作成

15-5. オブジェクトのマッピング


マッピング
モデル
データベース
データベースシステム内
マッピング
DOM
オブジェクト
ビュー
コント
ローラ
ユーザに見せる
ユーザから受け付ける

情報システムのアーキテクチャ



・画面表示
・画面，マウス，キーボードでの操作

オブジェクトのマッピングを行う理由

① データベースとのマッピング

Java オブジェクトを，データベース（リレーショナルデータベースシステム内）に**マッピング**

リレーショナルデータベースのデータ検索結果などを Java で簡単に扱えるように

② DOM オブジェクトとのマッピング

Java オブジェクトを，**DOMオブジェクト（Web ブラウザと相性が良い）**に**マッピング**

Web プログラムのダイナミック化

マッピングのための技術

- Java 言語

① **データベース向け**： SQL Alchemy など

② **DOM 向け**：DOM, SAX など（いずれも Java の標準）

- Python 言語

① **データベース向け**： SQLAlchemy など

② **DOM 向け**：dom パッケージなど

15-6. Java でのオブジェクトのマッピング

トピックス

- **リレーショナルデータベース**とのマッピング
- **XML, HTML, DOM**
- **DOM オブジェクト**とのマッピング

Spring JDBC のマッピングの例

Java
オブジェクト

リレーショナル
データベースの
テーブル

マッピングを
行う Java プログラム

Spring JDBC
のライブラリ

Spring JDBC のマッピングの例

Java
オブジェクト

リレーショナル
データベースの
テーブル

- リストオブジェクト
- 要素は Employee オブジェクト．属性は，id, name ,salary, joined

マッピングを
行う Java プログラム

id	name	salary	joined

```
List<Employee> employeeList = jdbcTemplate.query(
   "SELECT * FROM Employee",
    (rs, rowNum) -> {
       int id = rs.getInt("id");
       String name = rs.getString("name");
       BigDecimal salary = rs.getBigDecimal("salary");
       LocalDate joined =  rs.getDate("joined").toLocalDate();
       return new Employee(id, name, salary, joined);    } );
```

- テーブルを丸ごと読み込んで Java のリストオブジェクト化
- 「SELECT ・・・」のところには条件を指定可能

※ 記事をアップしています https://www.kkaneko.jp/pro/jdbc/springjdbc.html

XMLとは

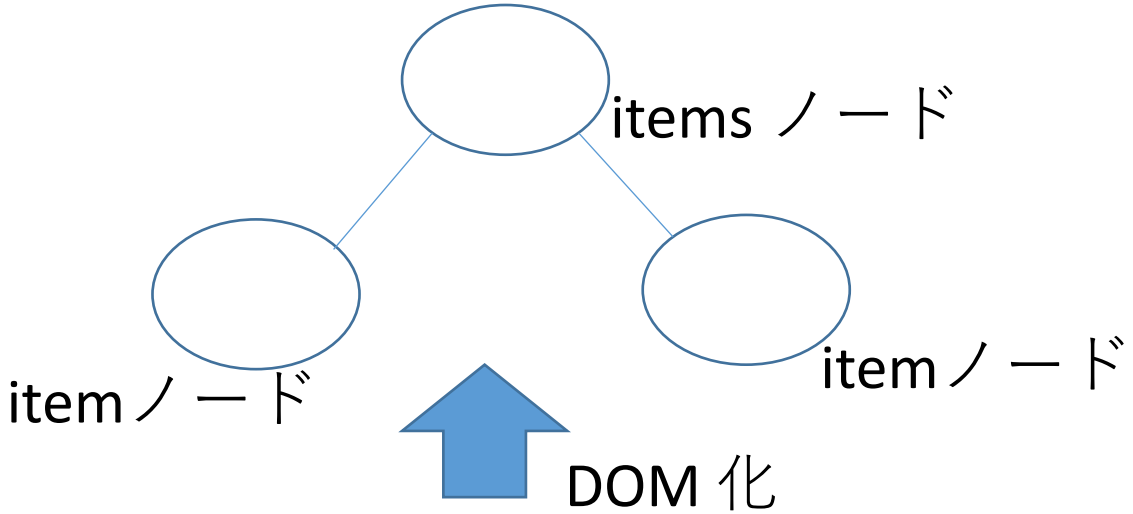
- XML とは eXtensible Markup Language のこと
- **タグ**，**属性**を使い文書を書く

```
<items>
<item id="001">XX</item>
<item id="002">YY</item>
</items>
```

item, items はタグ
id は属性

DOMとは

- DOM とは Document Object Model のこと
- **DOM では，ノードが階層構造をなすと考える**




```
<items>
<item id="001">XX</item>
<item id="002">YY</item>
</items>
```

item, items はタグ
id は属性

HTMLをプログラムで扱う理由

- **Webブラウザ**での表示を**ダイナミックに変えたい**とき，プログラムを書く

- そのとき，**DOM オブジェクト**を使うのは良い方針

HTML も DOM に準拠しつつある

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0" name="viewport">
<title>サンプル</title>
</head>
</body>
<h1>サンプル</h1>
</body>
</html>
```

HTML ファイル

Web ブラウザで表示

DOM でのマッピングの例 (Java 言語)

Java

DOM
オブジェクト

DOMオブジェクトの読み出し，書き込みを行うプログラム

JDBC の標準ライブラリ

```
<items>
<item id="ID">VALUE</item>
</items>
```

書き込まれたDOMオブジェクト

```
Document d = new XMLDocument();
Element r = document.createElement("items");
d.appendChild(r);
Element e = d.createElement("item");
r.appendChild(e);
Text t = d.createTextNode("VALUE");
e.appendChild(t);
e.setAttribute("id", "ID");
```

書き込みの例

DOM でのマッピングの例 (Python 言語)

Python

DOM
オブジェクト

DOMオブジェクトの読み出し，書き込みを行うプログラム

```
<items>
<item id="001">XX</item>
<item id="002">YY</item>
</items>
```

```
import xml.etree.ElementTree as ET
all = ET.Element('items')
x = ET.SubElement(all, 'item', {'id':'001'})
x.text = 'XX'
y = ET.SubElement(all, 'item', {'id':'002'})
y.text = ‘YY'
ET.dump(all)
```

書き込みの例

まとめ

- リレーショナルデータベースや、DOMオブジェクトは、Java オブジェクトへの**マッピング可能**
- リレーショナルデータベースや、DOMオブジェクトを、ふつうの Java オブジェクトと同じ感覚で扱える

15-1

```
import java.util.*;

class Circle {
    double x;
    double y;
    double r;
    String color;
    public Circle(double x, double y, double r, String color) {
        this.x = x;
        this.y = y;
        this.r = r;
        this.color = color;
    }
    public void printout() {
        System.out.printf("%f %f %f %s¥n", this.x, this.y, this.r, this.color);
    }
}
public class Main {
    public static void main(String[] args) throws Exception {
        Circle x = new Circle(2, 4, 3, "green");
        Circle y = new Circle(8, 10, 1, "blue");
        x.printout();
        y.printout();
    }
}
```

15-3

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.ArrayList;
class Person {
  String name;
  String address;
  public Person(String name, String address) {
    this.name = name;
    this.address = address;
  }
  public void printout() {
    System.out.printf("%s %s¥n", this.name, this.address);
  }
}
class Meibo {
  HashMap<Integer, Person> m;
  public Meibo(HashMap<Integer, Person> m) {
    this.m = m;
  }
  public void add(int id, String name, String address) {
    m.put(id, new Person(name, address));
  }
  public void printout() {
    for(Integer i : this.m.keySet()) {
      System.out.printf("%d, ", i);
      this.m.get(i).printout();
    }
  }
}
public class Main {
  public static void main(String[] args) throws Exception {
    HashMap<Integer, Person> m = new HashMap<Integer, Person>();
    Meibo a = new Meibo(m);
    a.add(1, "XX", "Fukuyama");
    a.add(2, "YY", "Okayama");
    a.printout();
  }
}
```

15-3

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.ArrayList;
class Person {
  String name;
  String address;
  public Person(String name, String address) {
    this.name = name;
    this.address = address;
  }
  public void printout() {
    System.out.printf("%s %s¥n", this.name, this.address);
  }
}
class Meibo {
  HashMap<Integer, Person> m;
  public Meibo(HashMap<Integer, Person> m) {
    this.m = m;
  }
  public void add(int id, String name, String address) {
    m.put(id, new Person(name, address));
  }
  public void printout() {
    for(Integer i : this.m.keySet()) {
      System.out.printf("%d, ", i);
      this.m.get(i).printout();
    }
  }
}
class View {
  ArrayList<Person> v;
  public View() {
  }
  public void update(Meibo meibo) {
    this.v = new ArrayList<Person>();
    for(Integer i: meibo.m.keySet()) {
      v.add(meibo.m.get(i));
    }
  }
  public void printout() {
    for(Person p: this.v) {
      System.out.printf("%s %s¥n", p.name, p.address);
    }
  }
}
public class Main {
  public static void main(String[] args) throws Exception {
    HashMap<Integer, Person> m = new HashMap<Integer, Person>();
    Meibo a = new Meibo(m);
    a.add(1, "XX", "Fukuyama");
    a.add(2, "YY", "Okayama");
    View v = new View();
    v.update(a);
    v.printout();
  }
}
```

関連ページ

- **Java プログラミング入門**

GDB online を使用

https://www.kkaneko.jp/cc/ji/index.html

- **Java の基本**

Java Tutor, GDB online を使用

https://www.kkaneko.jp/cc/pi/index.html

- **Java プログラム例**

https://www.kkaneko.jp/pro/java/index.html