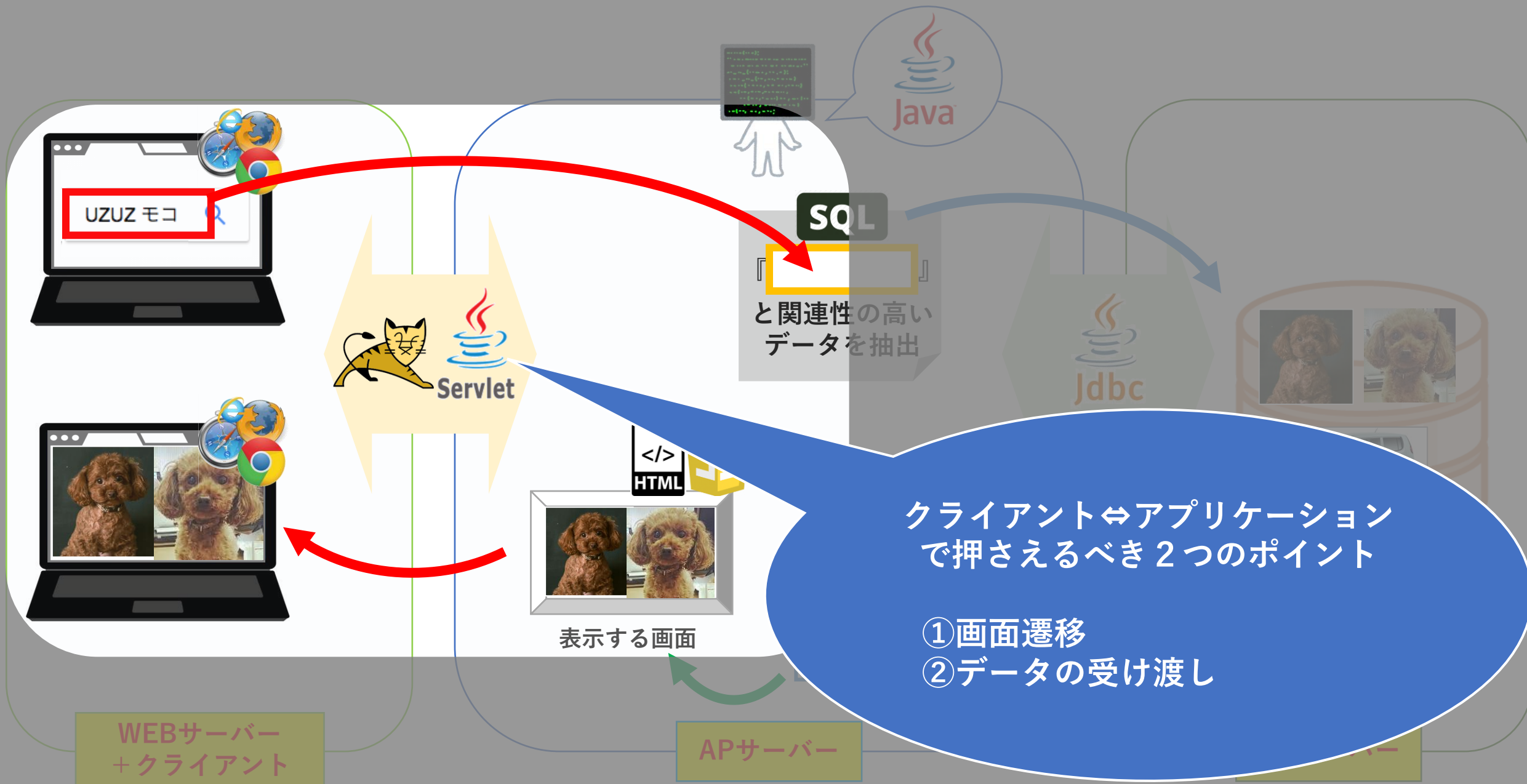
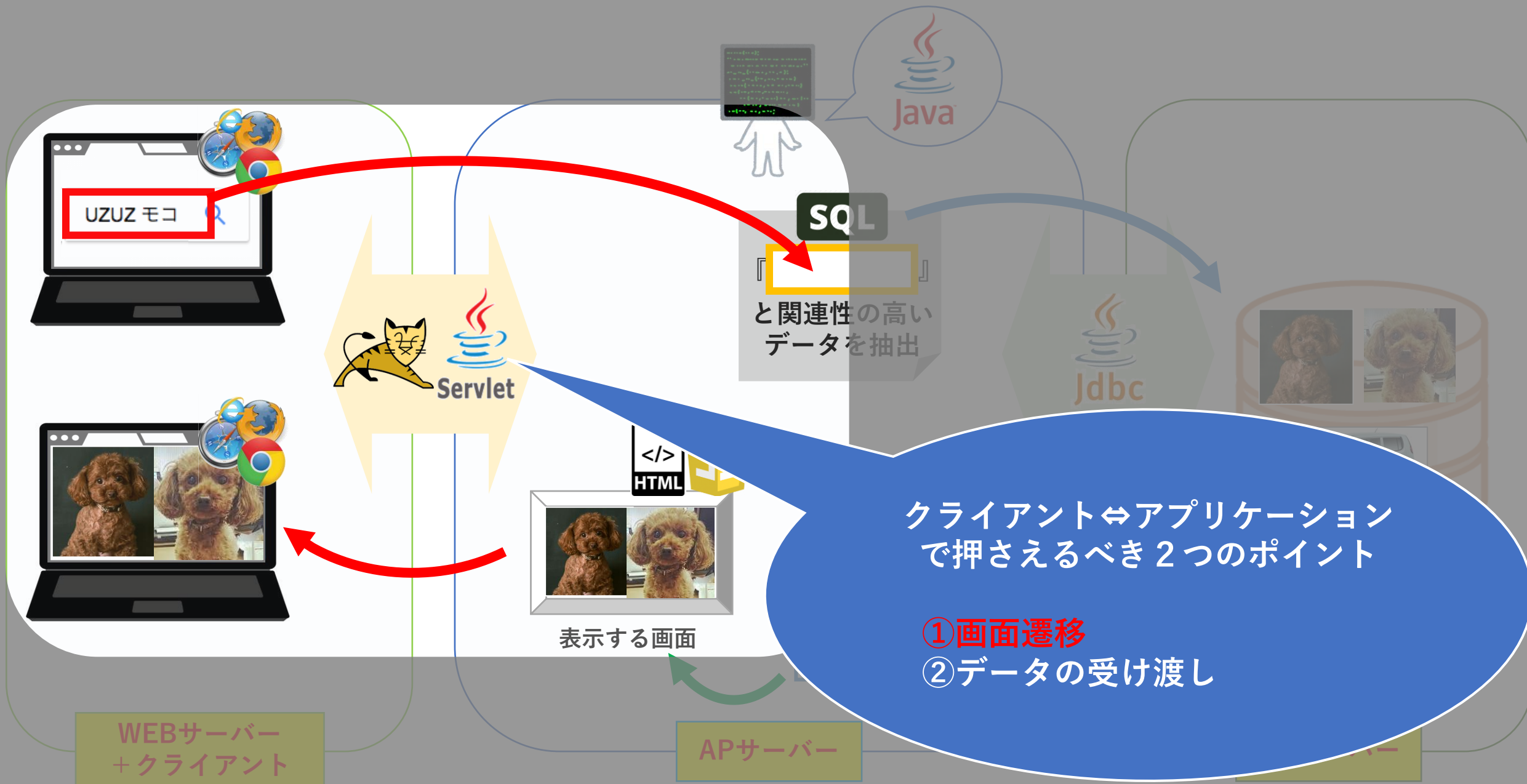


～Webアプリケーション開発講座～
画面遷移と引数の受け渡し







復習



～ 画面遷移の方法 ～

URLとは？

「Uniform Resource Locator」の略称で、ネットワーク上に存在するプログラムや情報資源（文書や画像など）の場所を指し示すために使われます。

http://localhost:8080/Sample5_02_1/SelfIntroduction

～ 画面遷移の方法 ～

URLとは？

「Uniform Resource Locator」の略称で、ネットワーク上に存在するプログラムや情報資源（文書や画像など）の場所を指し示すために使われます。

http://localhost:8080/Sample5_02_1/SelfIntroduction

通信プロトコル

通信方式がhttpプロトコルであることを指しています。

プロトコルとはネットワーク上でデータを通信するための「取り決め」のことを言います。

～ 画面遷移の方法 ～

URLとは？

「Uniform Resource Locator」の略称で、ネットワーク上に存在するプログラムや情報資源（文書や画像など）の場所を指し示すために使われます。

http://localhost:8080/Sample5_02_1/SelfIntroduction

ホスト（サーバー）名

ホスト（≡サーバー）、つまりアクセス先のコンピュータの情報を指定します。

「localhost」は「自分のコンピュータ」という意味を、「8080」はポート番号と言い、ここではApatchのことを指しています。

～ 画面遷移の方法 ～

URLとは？

「Uniform Resource Locator」の略称で、ネットワーク上に存在するプログラムや情報資源（文書や画像など）の場所を指し示すために使われます。

http://localhost:8080/Sample5_02_1/SelfIntroduction

コンテキスト名

アプリケーションの配置場所や呼び出し方を指します。
eclipseで作成した際はプロジェクト名が自動で設定されます。

～ 画面遷移の方法 ～

URLとは？

「Uniform Resource Locator」の略称で、ネットワーク上に存在するプログラムや情報資源（文書や画像など）の場所を指し示すために使われます。

http://localhost:8080/Sample5_02_1/SelfIntroduction

URLパターン

呼び出すプログラムファイルの情報を指します。
web.xmlというファイルの
servlet-mappingという箇所で設定します。
（詳しくは後述）

～ 画面遷移の方法 ～

URLとは？

「Uniform Resource Locator」の略称で、ネットワーク上に存在するプログラムや情報資源（文書や画像など）の場所を指し示すために使われます。

http://localhost:8080/Sample5_02_1/SelfIntroduction

（「http」という通信方式で）このサーバーのこのアプリケーションのこのファイルを呼び出す



サーバーへの接続情報

サーバーの情報

≪URL≫ http://localhost:8080/Sample5_02_1/SelfIntroduction

URLを入力して
Enter

リクエスト



ファイル
(SelfIntroduction.java)

アプリケーション
(Sample5_02_1)

サーバー

(localhost:8080)

クライアント
(ブラウザ)



ブラウザからURL指定でリクエストされた場合
自動でdoGetメソッドが起動される

リクエスト



クライアント
(ブラウザ)



ファイル
(SelfIntroduction.java)

アプリケーション
(Sample5_02_1)

サーバー
(localhost:8080)



UZUZ

ENTERTAIN
YOUR CAREER

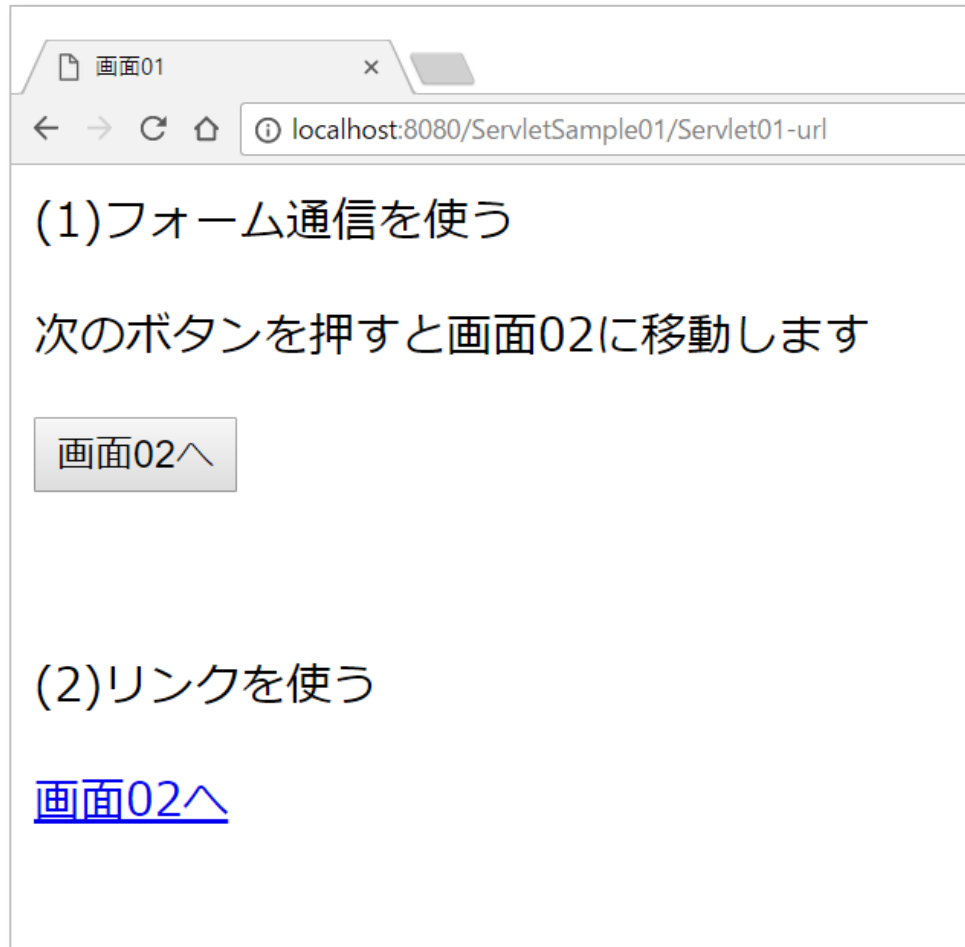


MOCO



～ サンプルを動かしてみよう ～

画面遷移



左のようなWebページを
eclipseから動作させてみましょう！

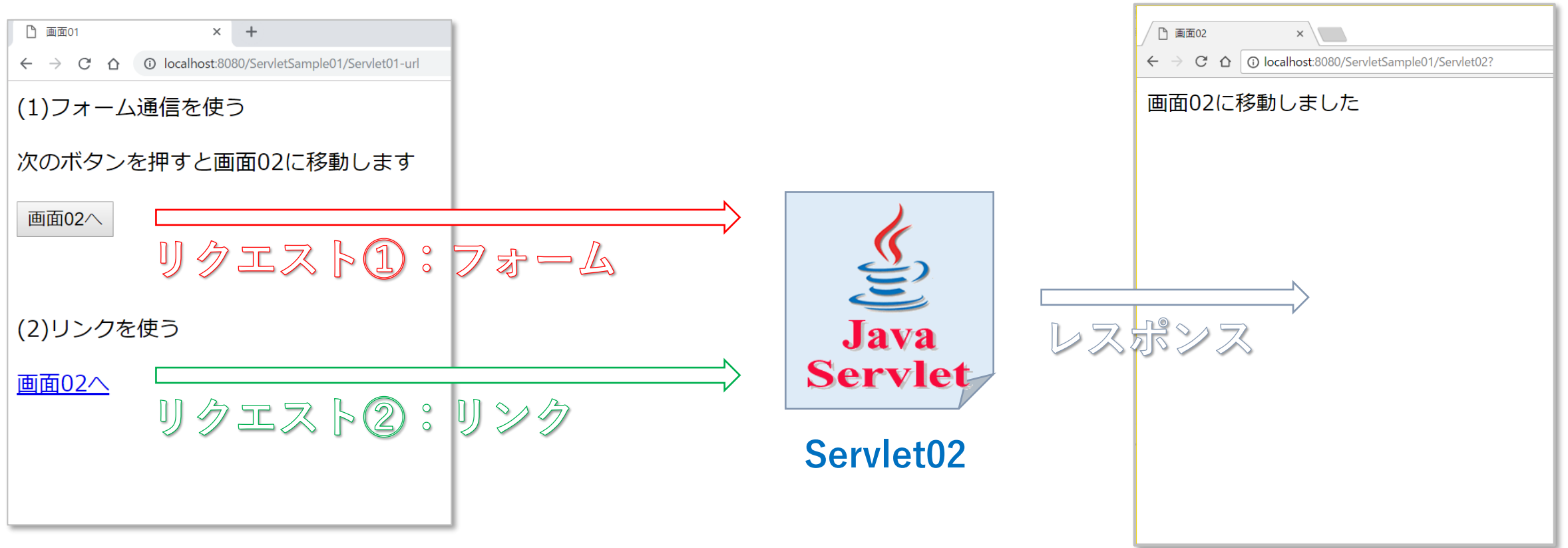
～ サンプルを動かしてみよう ～

手順

- (1) eclipse上に新しい動的Webプロジェクトを作成する。
プロジェクト名：Sample_5_03_1
- (2) プロジェクトの srcパッケージ直下に workパッケージを作成する。
- (3) ドライブのSample_5_03_1フォルダからソースコードを取得して
workパッケージ直下にインポートする。
 - Servlet01.java
 - Servlet02.java
- (4) 同じく web.xmlをドライブから取得して置き換える。
WebContent > WEB-INF > web.xml
- (5) Servlet01.javaを選択し、サーバーを起動してWebページを開く。

～ サンプルを動かしてみよう ～

ページ間遷移における2つのリクエスト方法

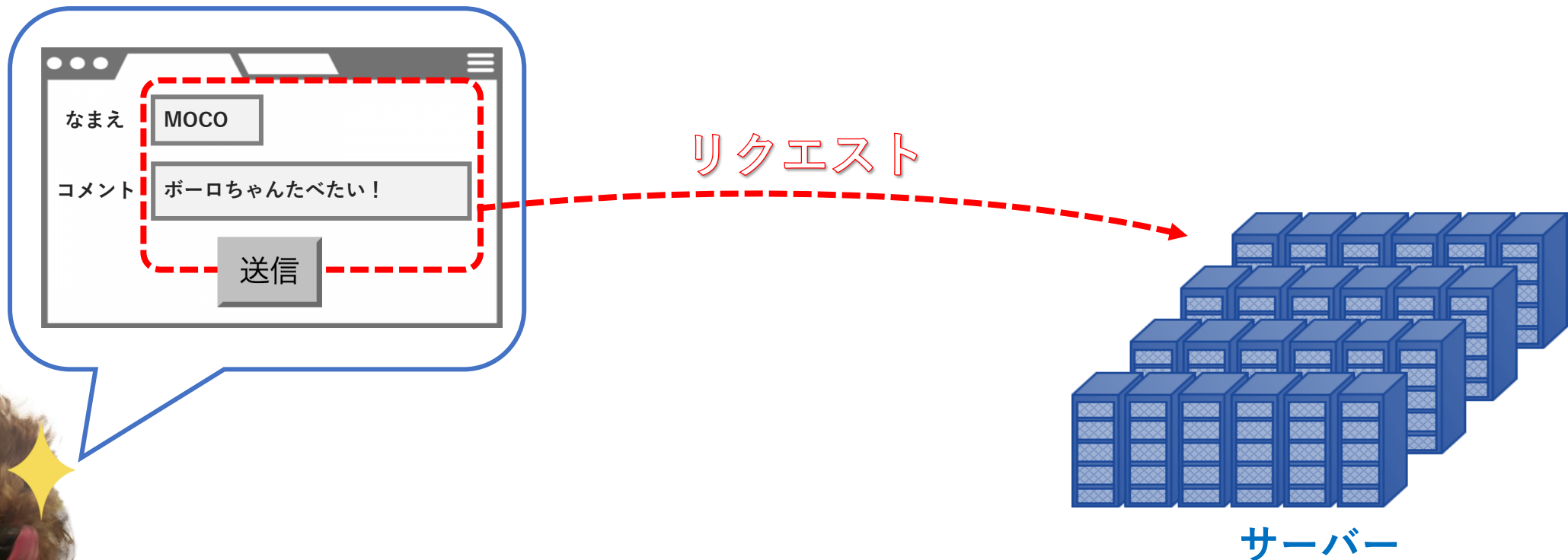


※使い分けに関してはこの授業の次の章で扱います。

～ フォームとは？～

フォーム

ユーザーがブラウザ上で入力したデータをリクエストとしてサーバーに送るための技術です。HTMLの複数のタグを用いて作成します。



～ フォームとは？～

フォームの構造

formタグ

<form>～</form>内で入力された情報をリクエストとしてサーバーに送信することができます。

```
<form action=¥"Servlet02¥" method=¥"get¥">
```

```
<p>次のボタンを押すと画面02に移動します</p>
```

```
<input type=¥"submit¥" value=¥"画面02へ¥">
```

```
</form>
```

～ フォームとは？～

フォームの構造

formタグ

<form>～</form>内で入力された情報をリクエストとしてサーバーに送信することができます。

action属性

送信先のプログラムのURLパターンを指定します。（『/』は不要）

```
<form action=¥"Servlet02¥" method=¥"get¥">
```

```
<p>次のボタンを押すと画面02に移動します</p>
```

```
<input type=¥"submit¥" value=¥"画面02へ¥">
```

```
</form>
```

～ フォームとは？～

フォームの構造

formタグ

<form>～</form>内で入力された情報をリクエストとしてサーバーに送信することができます。

method属性

起動するリクエストメソッドを指定します。「get」と書けばdoGetメソッド、「post」と書けばdoPostメソッドが起動されます。
method属性を書かない場合は自動でdoGetメソッドが起動されます。

```
<form action=¥"Servlet02¥" method=¥"get¥">
```

```
<p>次のボタンを押すと画面02に移動します</p>
```

```
<input type=¥"submit¥" value=¥"画面02へ¥">
```


```
</form>
```

～ フォームとは？～

フォームの部品（inputタグ）

inputタグ

フォーム（<form>～</form>）を構成する様々な入力部品を作成する際に使用します。

ここで  という送信ボタンを作成しています。


```
<form action=¥"Servlet02¥" method=¥"get¥">  
<p>次のボタンを押すと画面02に移動します</p>  
<input type=¥"submit¥" value=¥"画面02へ¥">  
</form>
```

～ フォームとは？～

フォームの部品 (inputタグ)

inputタグ

フォーム (<form>～</form>) を構成する様々な入力部品を作成する際に使用します。

ここで  という送信ボタンを作成しています。

type属性「submit」

type属性を“submit”にすることでボタンの部品を作ることができます。

```
<input type="submit" value="画面02へ">
```


```
</form>
```

～ フォームとは？～

フォームの部品（inputタグ）

inputタグ

フォーム（<form>～</form>）を構成する様々な入力部品を作成する際に使用します。

ここで  という送信ボタンを作成しています。

value属性

ここで指定した文字列がボタンに表示されます。

```
<input type="submit" value="画面02へ">
```

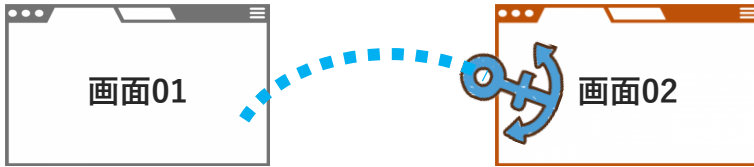
```
</form>
```

～ リンクとは？～

リンク (aタグ)

aタグ

リンクを作成する際に使用します。
aはanchor（いかり）の略です。



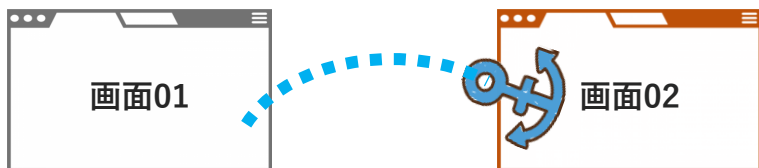
`画面02へ`

～ リンクとは？～

リンク (aタグ)

aタグ

リンクを作成する際に使用します。
aはanchor（いかり）の略です。



href属性

送信先のプログラムのURLパターン
を指定します。（『/』は不要）

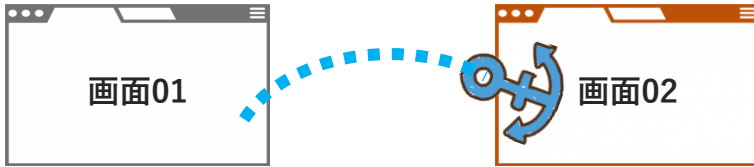
`画面02へ`

～ リンクとは？～

リンク (aタグ)

aタグ

リンクを作成する際に使用します。
aはanchor（いかり）の略です。



aタグで囲われた文字列にリンクとしての機能が付与されます。

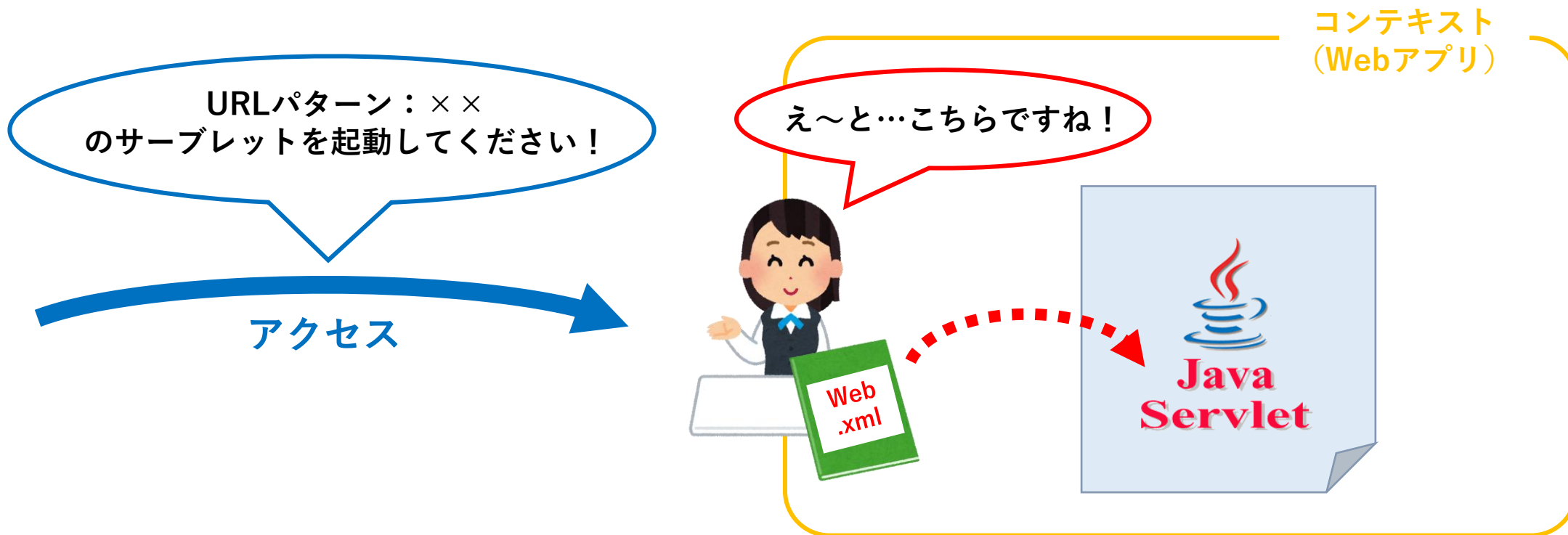
[画面02へ](#)

`画面02へ`

～ Web.xmlとは？～

Web.xml

XML形式で記述された、Webアプリケーションを動作させるための設定ファイルです。
XMLは「eXtensible Markup Language」の略称で、データを管理できないHTMLを拡張する
目的で考案されたマークアップ言語です。



～ Web.xml とは？ ～

```
<course>
```

```
  <course-name>プログラマーコース</course-name>
```

```
  <student>
```

```
    <name>モコ</name>
```

```
    <age>5</age>
```

```
    <sex>メス</sex>
```

```
  </student>
```

```
  <student>
```

```
    <name>ポチ</name>
```

```
    <age>6</age>
```

```
    <sex>オス</sex>
```

```
  </student>
```

```
</course>
```

～ Web.xmlとは？～

Web.xmlの内容

```
<servlet>  
  <servlet-name>Servlet01-name</servlet-name>  
  <servlet-class>work.Servlet01</servlet-class>  
</servlet>  
<servlet-mapping>  
  <servlet-name>Servlet01-name</servlet-name>  
  <url-pattern>/Servlet01-url</url-pattern>  
</servlet-mapping>
```

« web.xml 3～10行目 »

～ Web.xmlとは？～

Web.xmlの内容

```
<servlet>
```

```
<servlet-name>Servlet01-name</servlet-name>
```

```
<servlet-class>work.Servlet01</servlet-class>
```

```
</servlet>
```

サーブレットに関する設定

あだ名 『Servlet01-name』
クラス名 『work.Servlet01』

```
<servlet-mapping>
```

```
<servlet-name>Servlet01-name</servlet-name>
```

```
<url-pattern>/Servlet01-url</url-pattern>
```

```
</servlet-mapping>
```

サーブレットの関連付けに関する設定

あだ名 『Servlet01-name』
URL 『/Servlet01-url』

≪ web.xml 3～10行目 ≫

～ Web.xmlとは？～

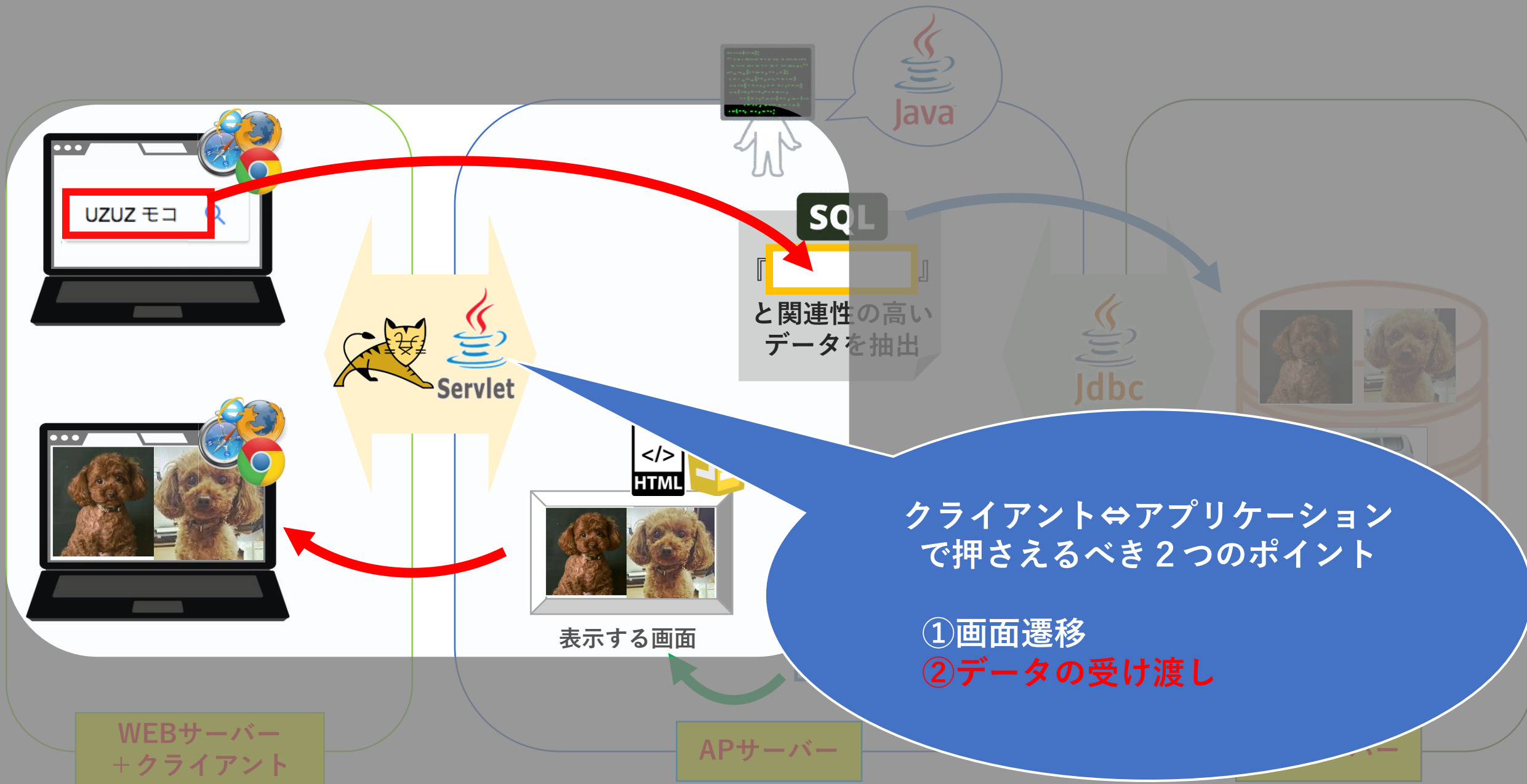
Web.xmlの内容

```
<servlet>
  <servlet-name>Servlet01-name</servlet-name>
  <servlet-class>work.Servlet01</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Servlet01-name</servlet-name>
  <url-pattern>/Servlet01-url</url-pattern>
</servlet-mapping>
```

このサーブレットクラスの
あだ名は〇〇です！
URLパターンは××です！

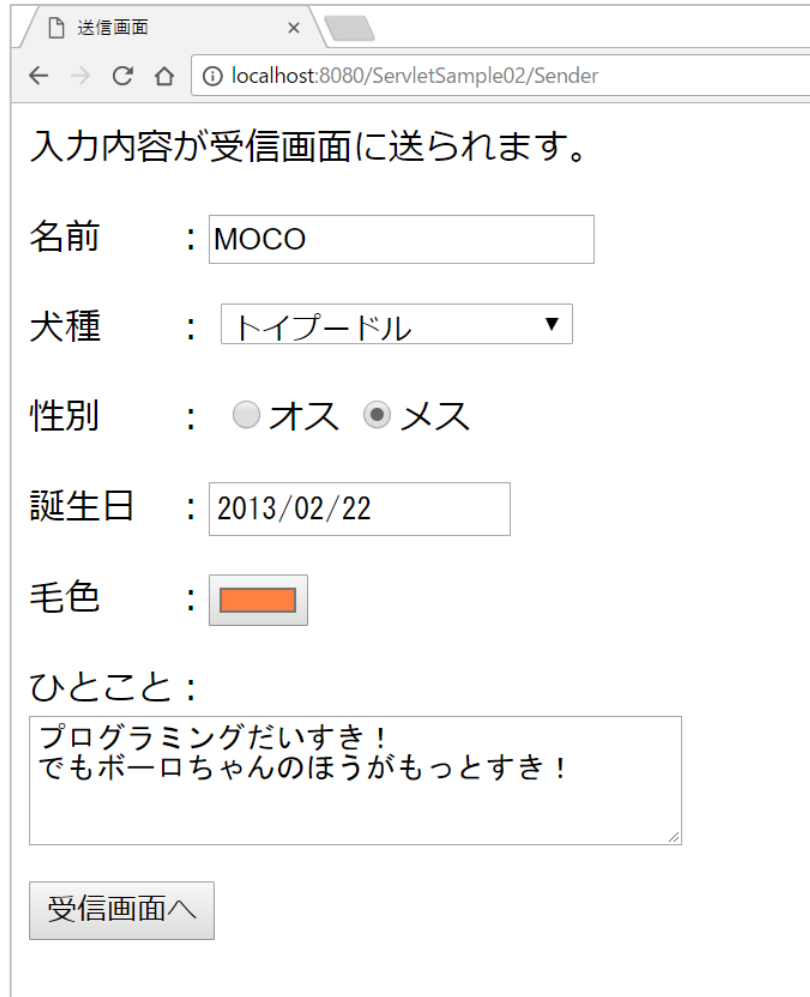
データ
(設定情報)

« web.xml 3～10行目 »



～ サンプルを動かしてみよう ～

画面間のデータの受け渡し



送信画面

localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 :

犬種 :

性別 : ☐ オス ☒ メス

誕生日 :

毛色 :

ひとこと :

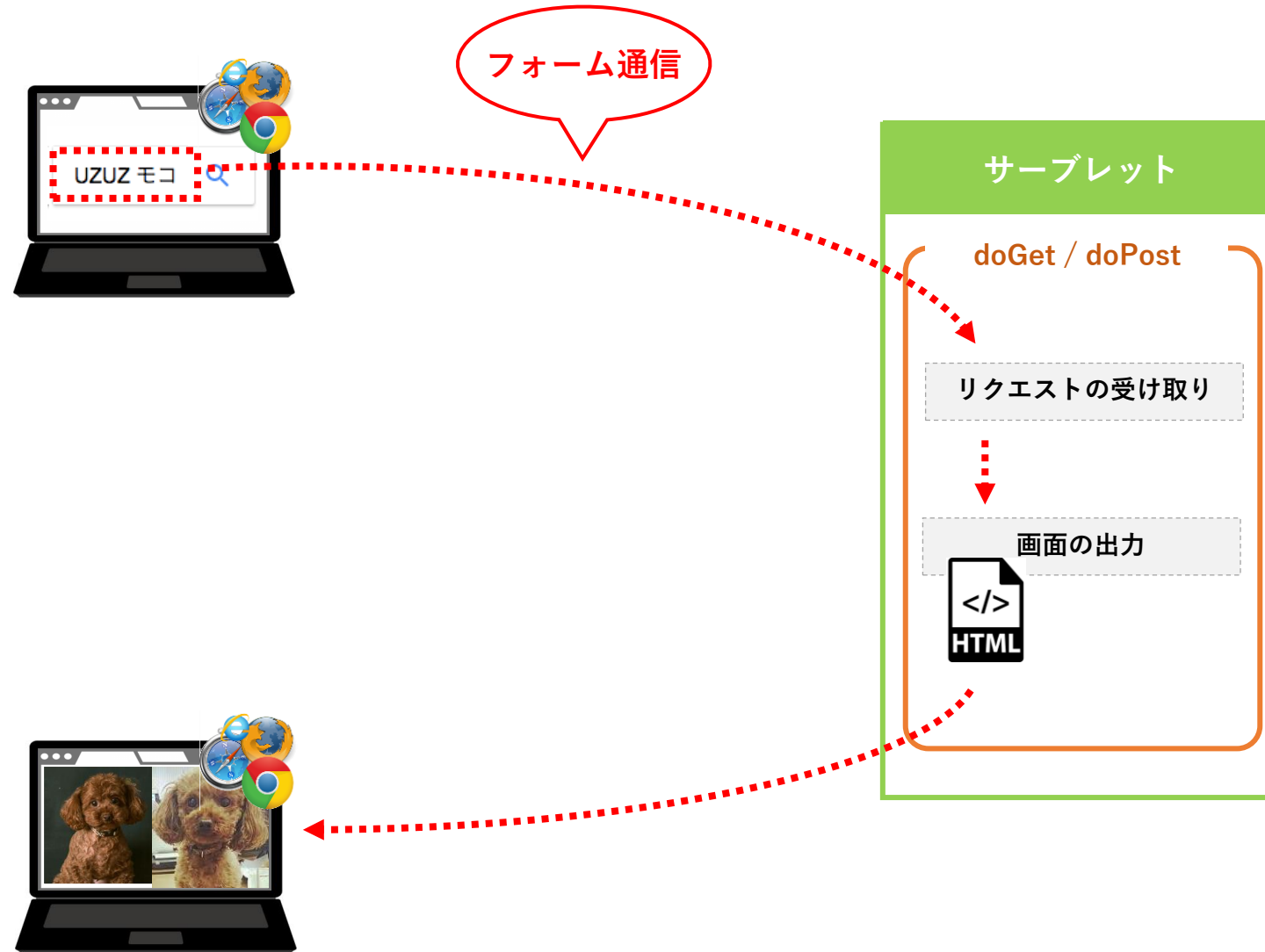
左のようなWebページを
eclipseから動作させてみましょう！

～ サンプルを動かしてみよう ～

手順

- (1) eclipse上に新しい動的Webプロジェクトを作成する。
プロジェクト名: **Sample_5_03_2**
- (2) プロジェクトの srcパッケージ直下に workパッケージを作成する。
- (3) ドライブのSample_5_03_2フォルダからソースコードを取得して
workパッケージ直下にインポートする。
 - **Sender.java**
 - **Receiver.java**
- (4) 同じく **web.xml** をドライブから取得して置き換える。
WebContent > WEB-INF > web.xml
- (5) Sender.javaを実行し、サーバーを起動してWebページを開く。

～ リクエストの受け取り→ページ遷移の流れ ～



≪ Senderクラス ≫

```
42 out.println("
43 out.println("
44 out.println("
45 out.println("
46 out.println("
47 out.println("
48 out.println("
49 out.println("
50 out.println("
```

ユーザーから入力を受け付けるための
部品の情報

```
57 out.println("
58 out.println("
59 out.println("
60 out.println("
61 out.println("
62 out.println("
63 out.println("
64 out.println("
65 out.println("
66 out.println("
67 out.println("
68 out.println("
69 out.println("
70 out.println("
71 out.println("
72 out.println("
73 out.println("
74 out.println("
```

```
<form action=%"Receiver%" method=%"post%">
<p>
<label>名前      : <input type=%"text%" name=%"NAME%"></label>
</p>
<p>
<label>犬種      : </label>
<select name=%"DOG_TYPE%">
<option value=%"S%">シバイヌ</option>
<option value=%"T%">トイプードル</option>
<option value=%"G%">ゴールデンレトリバー</option>
<option value=%"O%">その他</option>
</select>
</p>
<p>
<label>性別      : </label>
<input type=%"radio%" name=%"GENDER%" value=%"1%">オス
<input type=%"radio%" name=%"GENDER%" value=%"2%">メス
</p>
<p>
<label>誕生日    : <input type=%"date%" name=%"BIRTHDAY%"></label>
</p>
<p>
<label>毛色      : <input type=%"color%" name=%"COLOR%"></label>
</p>
<p>
<label>ひとこと : </label>
<br>
<textarea name=%"COMMENT%" rows=%"4%" cols=%"40%"></textarea>
</p>
<p>
<label><input type=%"submit%" value=%"受信画面へ%"></label>
</p>
</form>
```

formタグ

送信ボタン

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 : MOCO

犬種 : トイプードル

性別 : ☐ オス ☒ メス

誕生日 : 2013/02/22

毛色 :

ひとこと:

プログラミングだいすき！
でもボーロちゃんのほうがもっとすき！

受信画面へ

formタグで囲っている範囲

リクエスト
(フォーム)

受信画面

localhost:8080/ServletSample02/Receiver

以下の内容が送られてきました。

▼name属性『NAME』として受け取った値
MOCO

▼name属性『DOG_TYPE』として受け取った値
T

▼name属性『GENDER』として受け取った値
2

▼name属性『BIRTHDAY』として受け取った値
2013-02-22

▼name属性『COLOR』として受け取った値
#ff8000

▼name属性『COMMENT』として受け取った値
プログラミングだいすき！ でもボーロちゃんのほうがもっとすき！

[送信画面に戻る](#)

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

← → ↺ ⌂ localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前

犬種 :

性別 : ☐ オス ☒ メス

誕生日 :

毛色 :

ひとこと :

プログラミングだいすき！
でもボーロちゃんのほうがもっとすき！

受信画面へ

```
<input type="text" name="NAME" >
```

《 Senderクラス 44行目 》

入力ボックスに
入力された値

NAME

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

← → ↺ ⌂ localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 : MOCO

犬種 : トイプードル ▼

性別 : ☐ オス ☒ メス

誕生日 : 2013/02/22

毛色 :

ひとこと :
プログラミングだいすき！
でもボーロちゃんのほうがもっとすき！

受信画面へ

```
<select name=¥"DOG_TYPE¥">
<option value=¥"S¥">シバイヌ</option>
<option value=¥"T¥">トイプードル</option>
<option value=¥"G¥">ゴールデンレトリバー</option>
<option value=¥"O¥">その他</option>
</select>
```

≪ Senderクラス 48～53行目 ≫

選択された値に
対応するvalue値

DOG_TYPE

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 : MOCO

犬種 : トイプードル ▼

性別 : ☐ オス ☒ メス

誕生日 : 2013/02/22

毛色 : 

ひとこと :

プログラミングだいすき！
でもボーロちゃんのほうがもっとすき！

受信画面へ

```
<input type="radio" name="GENDER" value="1">オス  
<input type="radio" name="GENDER" value="2">メス
```

« Senderクラス 57～58行目 »

選択された値に
対応するvalue値

GENDER

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

← → ↺ ⌂ localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 :

犬種 :

性別 : ☐ オス ☒ メス

誕生日 :

毛色 :

ひとこと :

プログラミングだいすき！
でもボーロちゃんのほうがもっとすき！

受信画面へ

<input type="date" name="BIRTHDAY">

≪ Senderクラス 61行目 ≫

入力ボックスに
入力された値

BIRTHDAY

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 :

犬種 :

性別 : ☐ オス ☒ メス

誕生日 :

毛色 :

ひとこと :

`<input type="color" name="COLOR">`

《 Senderクラス 64行目 》

入力ボックスに
入力された値

COLOR

～ フォームのしくみを理解しよう ～

フォームのしくみ

送信画面

localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前 :

犬種 :

性別 : ☐ オス ☒ メス

誕生日 :

毛色 :

ひとこと :

でもボーロちゃんのほうがもっとすき！

```
<textarea name="COMMENT" rows="4" cols="40"></textarea>
```

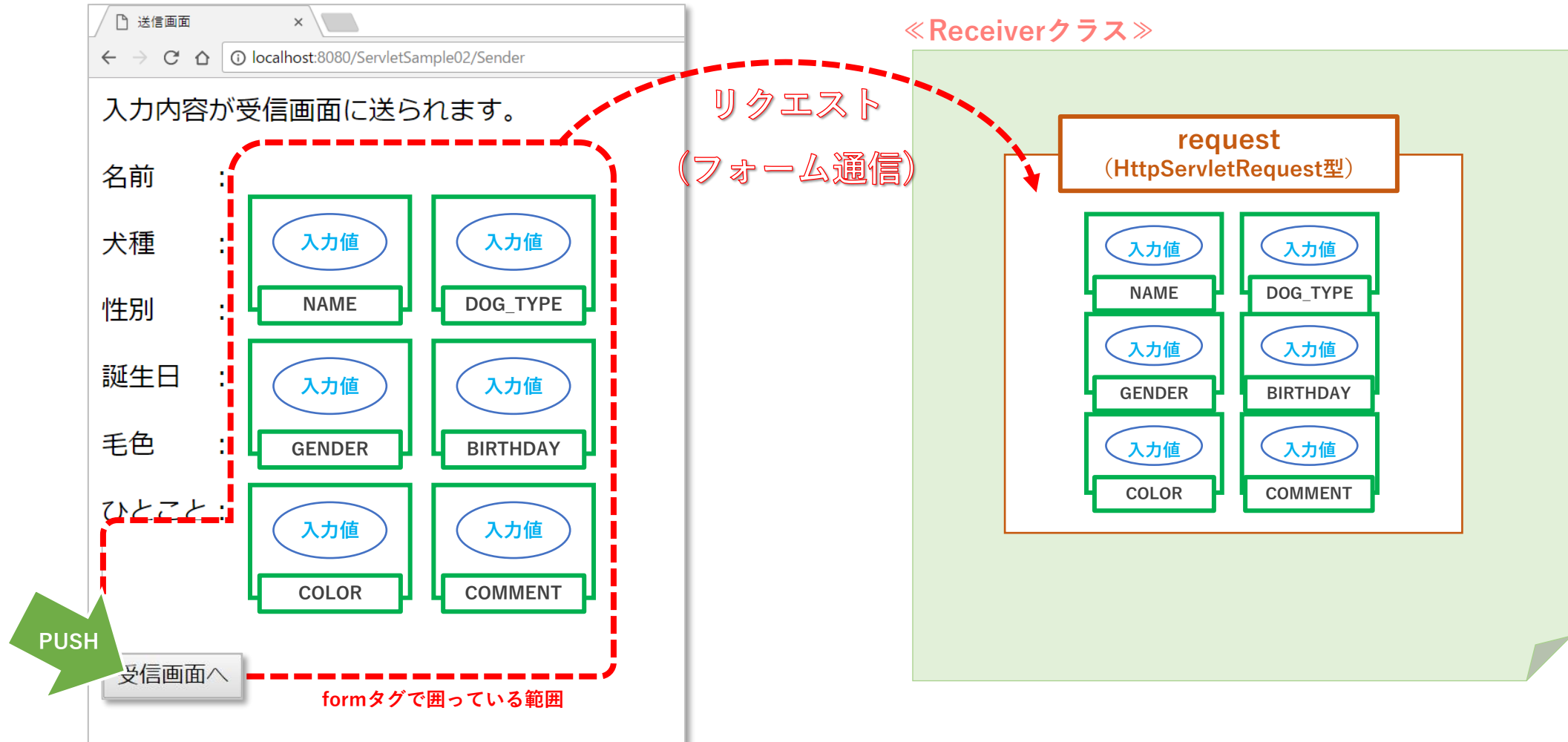
« Senderクラス 69行目 »

入力ボックスに
入力された値

COMMENT

～ フォームのしくみを理解しよう ～

フォームのしくみ



～ フォームのしくみを理解しよう ～

フォームのしくみ

入力値とその名前を合わせて
リクエストパラメータ
と言います。

送信画面

localhost:8080/ServletSample02/Sender

入力内容が受信画面に送られます。

名前	:	入力値	入力値
犬種	:	NAME	DOG_TYPE
性別	:	入力値	入力値
誕生日	:	GENDER	BIRTHDAY
毛色	:	入力値	入力値
ひとこと	:	COLOR	COMMENT

PUSH 受信画面へ

formタグで囲っている範囲

～ doGet と doPost ～

フォームの中で起動するメソッド（doGet/doPost）を指定する

≪ Senderクラス 42行目 ≫

```
<form action=#{ "Receiver" } method=#{ "post" }>
```

Receiverクラスに
リクエストを送る

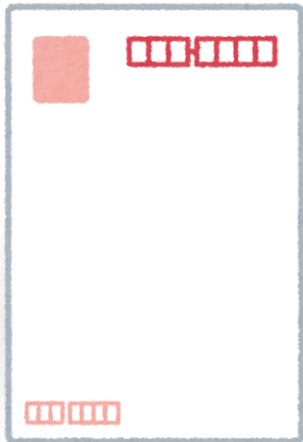
doPostメソッド
を起動

～ doGet と doPost ～

doGet と doPost

《見える場所に記載》

doGet



《見えない場所に記載》

doPost



- 入力した文字情報などをサーバー側に送る処理を行う場合は主にdoPostメソッドを使用します。

※doGetとdoPostの違い

- ・ doGet : 主に画面を表示させるリクエストを出す際に使用する
- ・ doPost : 主にサーバーに顧客の入力データを送る際に使用する

→ データを送受信するための方式が違う

- doGet(GET通信) : URLに送信データを組み込む

(イメージ)

<https://www.XXX.com/sample?name=moco&gender=2&age=3>

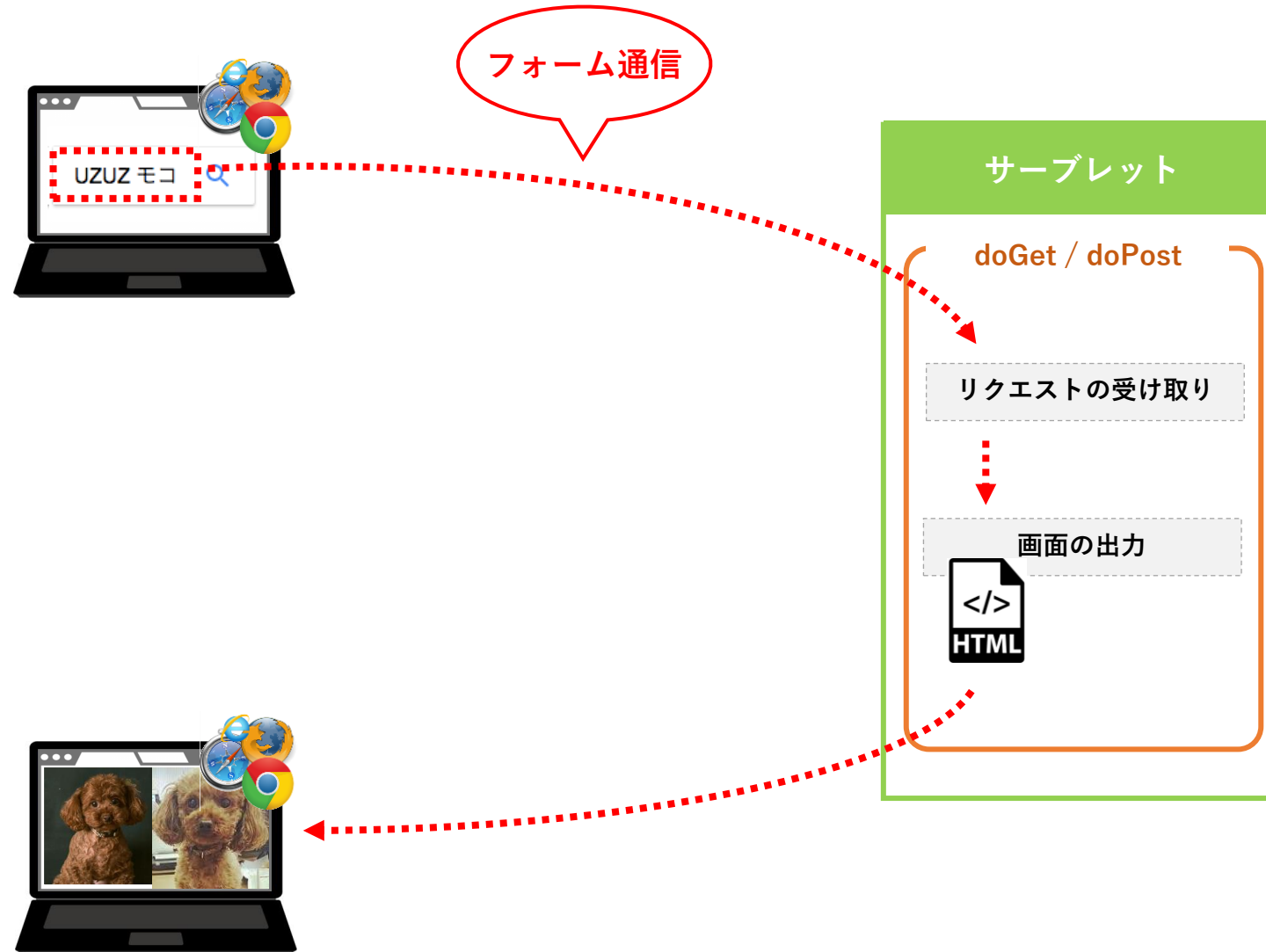
→ 履歴等に残るためセキュリティが…

- doPost(POST通信) : HTTPリクエストのメッセージボディに組み込む

→ 履歴等に残らない

※ただしセキュリティが高いわけではない

～ リクエストの受け取り→ページ遷移の流れ ～



《 Receiverクラス 42～85行目 》

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

    response.setContentType("text/html;charset=UTF-8");
    request.setCharacterEncoding("UTF-8"); // HTTP通信で引数を受け取る際の文字コードの指定

    // 引数の受け取り
    String receiveParameterName      = request.getParameter("NAME");
    String receiveParameterDogType   = request.getParameter("DOG_TYPE");
    String receiveParameterGender     = request.getParameter("GENDER");
    String receiveParameterBirthday  = request.getParameter("BIRTHDAY");
    String receiveParameterColor     = request.getParameter("COLOR");
    String receiveParameterComment    = request.getParameter("COMMENT");

    PrintWriter out = response.getWriter();

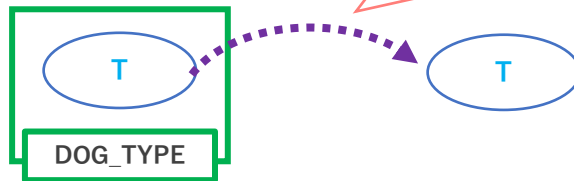
    out.println("<html>");
    out.println("<head>");
    out.println("    <title>受信画面</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("    <p>以下の内容が送られてきました。</p>");
    out.println("    <p>");
    out.println("        ▼name属性『NAME』として受け取った値<br>" + receiveParameterName);
    out.println("    </p>");
    out.println("    <p>");
    out.println("        ▼name属性『DOG_TYPE』として受け取った値<br>" + receiveParameterDogType);
    out.println("    </p>");
    out.println("    <p>");
    out.println("        ▼name属性『GENDER』として受け取った値<br>" + receiveParameterGender);
    out.println("    </p>");
    out.println("    <p>");
    out.println("        ▼name属性『BIRTHDAY』として受け取った値<br>" + receiveParameterBirthday);
    out.println("    </p>");
    out.println("    <p>");
    out.println("        ▼name属性『COLOR』として受け取った値<br>" + receiveParameterColor);
    out.println("    </p>");
    out.println("    <p>");
    out.println("        ▼name属性『COMMENT』として受け取った値<br>" + receiveParameterComment);
    out.println("    </p>");
    out.println("    <a href=\"%Sender%\">送信画面に戻る</a>");
    out.println("</body>");
    out.println("</html>");
}
```

リクエストの
受け取り

受け取ったデータを元に
表示画面を作成

～ リクエストの受け取り ～

リクエストパラメータの取得



`request.getParameter("DOG_TYPE")`

受信画面

localhost:8080/ServletSample02/Receiver

以下の内容が送られてきました。

- ▼name属性『NAME』として受け取った値
MOCO
- ▼name属性『DOG_TYPE』として受け取った値
T
- ▼name属性『GENDER』として受け取った値
2
- ▼name属性『BIRTHDAY』として受け取った値
2013-02-22
- ▼name属性『COLOR』として受け取った値
#ff8000
- ▼name属性『COMMENT』として受け取った値
プログラミングだいすき！ でもボーロちゃんのほうがもっとすき！

[送信画面に戻る](#)

【演習】

「Sender.java」 「Receiver.java」 を書き換えて
オリジナルの投稿&表示ページを作ってみましょう！



【演習】

手順

- (1) eclipse上に新しい動的Webプロジェクトを作成する。
プロジェクト名：Ex_5_03
- (2) プロジェクトの srcパッケージ直下に workパッケージを作成する。
- (3) プロジェクト「Sample_5_03_2」を参考に演習に取り組む。
 - ・ Javaリソース > src > work 直下にjavaソースコードを配備
 - ・ WebContent > WEB-INF 直下にweb.xmlを配備