

# プログラミング Java 3

## 第2回:

## Web Formおよびサーブレット入門


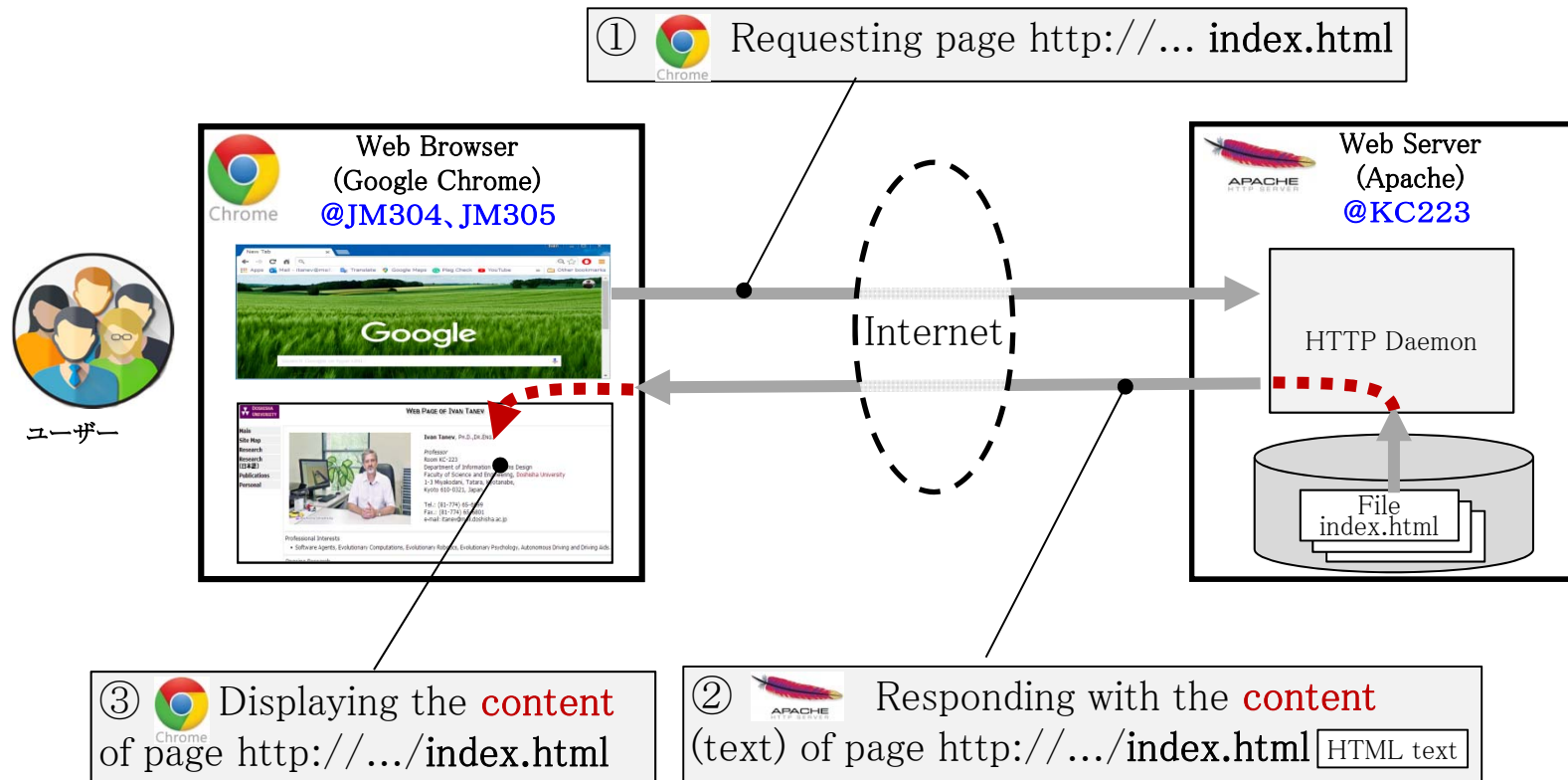
Ivan Tanev



# 講義の構造

1. スタティック(静的な)Webコンテンツ
2. ダイナミック (動的)Webコンテンツ
3. ダイナミックWebコンテンツとサーバレット
4. Webフォーム
5. 演習

# 1. スタティック(静的な)Webコンテンツ

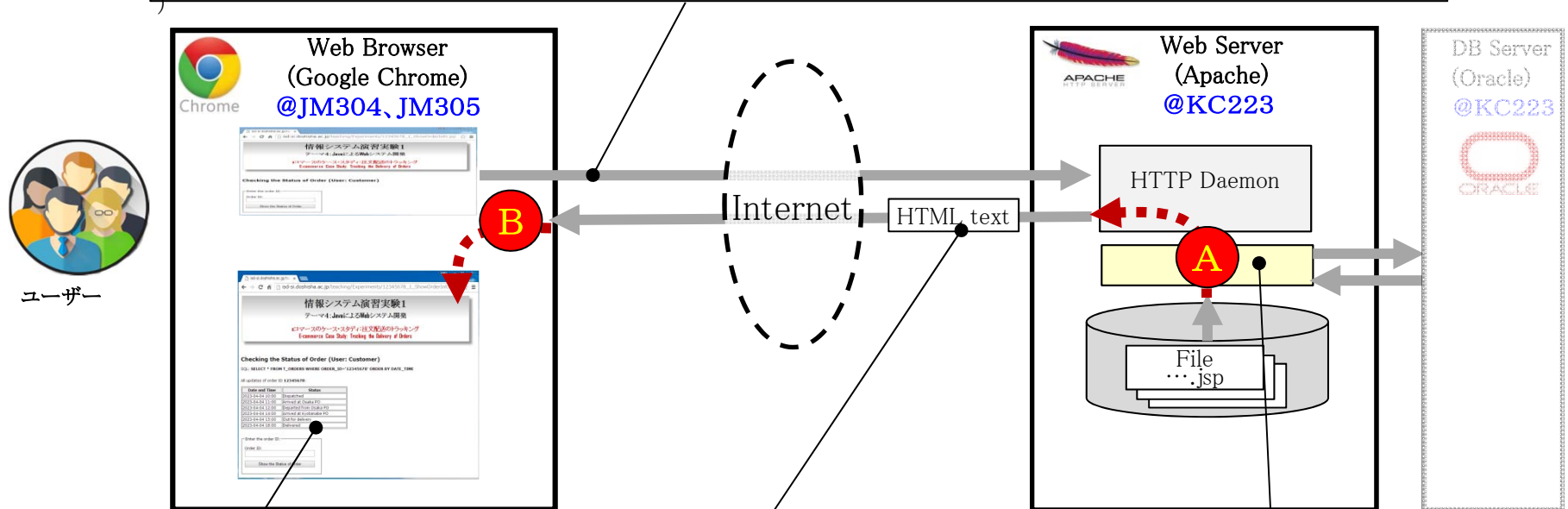



Q: なぜ「静的なWebコンテンツ」と呼ばれているのですか？


A: (a) Web server上に保存された内容と②の時に通信された内容は**同じ**です。  
(b) ②の時に通信された内容と③の時に表示された内容は**同じ**です。  
(a)&(b) => ③の時に表示された内容とWeb server上に保存された内容は**同じ**です。


# 3. ダイナミック (動的)Webコンテンツ

①  Requesting the Order Info page `http://... XXXXXX_1_Show_Order_Info.jsp`  
 例: 演習実験1、テーマ4 (JavaによるWebシステム開発eコマースのケース・スタディ: 注文配送のトラッキング)



④  Displaying the **result of execution** of `http://... .jsp`

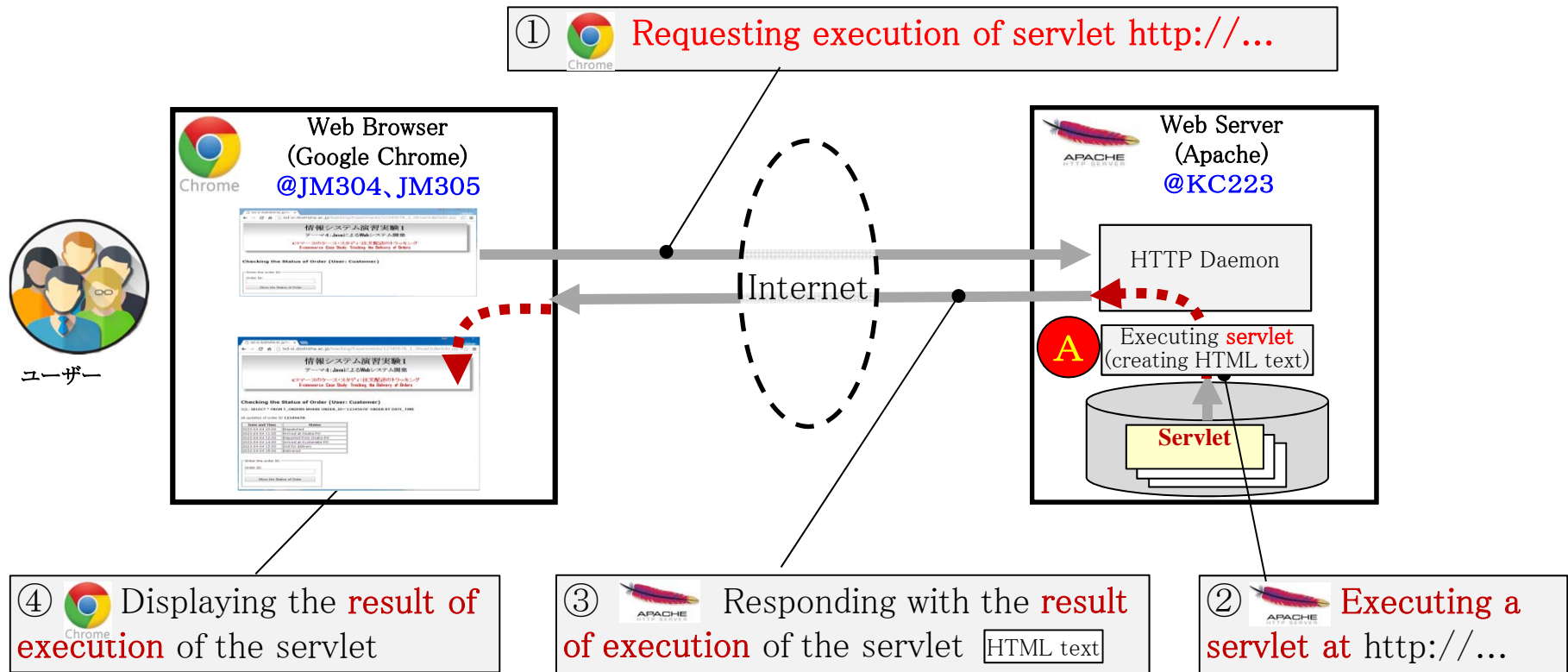
③  Responding with the **result of execution** of jsp page `http://... .jsp`

②  **Executing jsp** page `http://... .jsp`

Q: **A** のデータ処理はどのように実現されていますか？  
 A: サーバサイドスクリプト: Web server上に実行されたコード(script)です。例えば、Common Gateway Interface (CGI), Java servlet, Java Server Pages (jsp), Active Server Pages (asp), Server Side Java Scriptなど。

Q: **B** のデータ処理はどのように実現されていますか？  
 A: クライアントサイドスクリプト: Web browser上に実行されたコード(script)です。例えば、Java Script, ActiveX, Visual Basic Scriptなど。

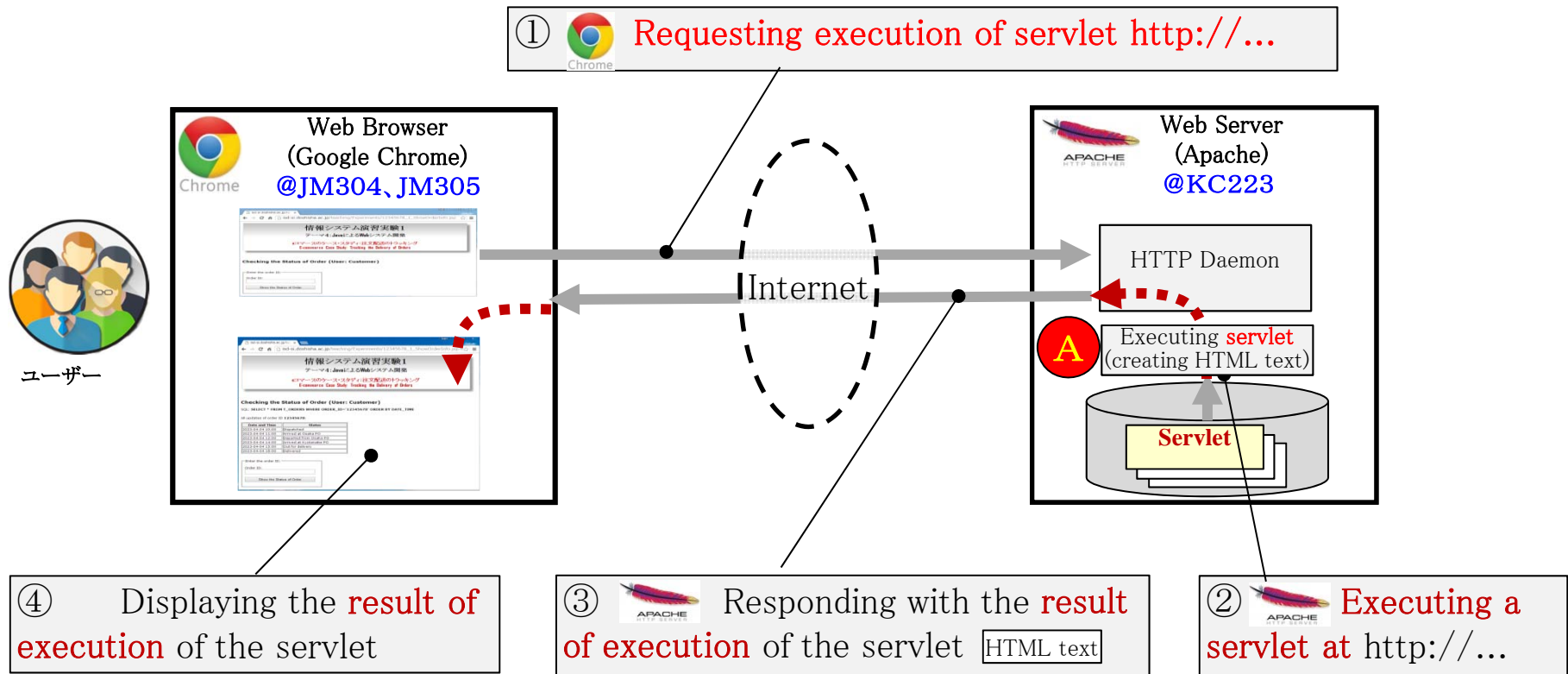
### 3. ダイナミック (動的)Webコンテンツとサーブレット



Q: のデータ処理はどのように実現されていますか？

A: サーバサイドスクリプト: Web server上に実行されたコード(script)です. 例えば、Java servlet.

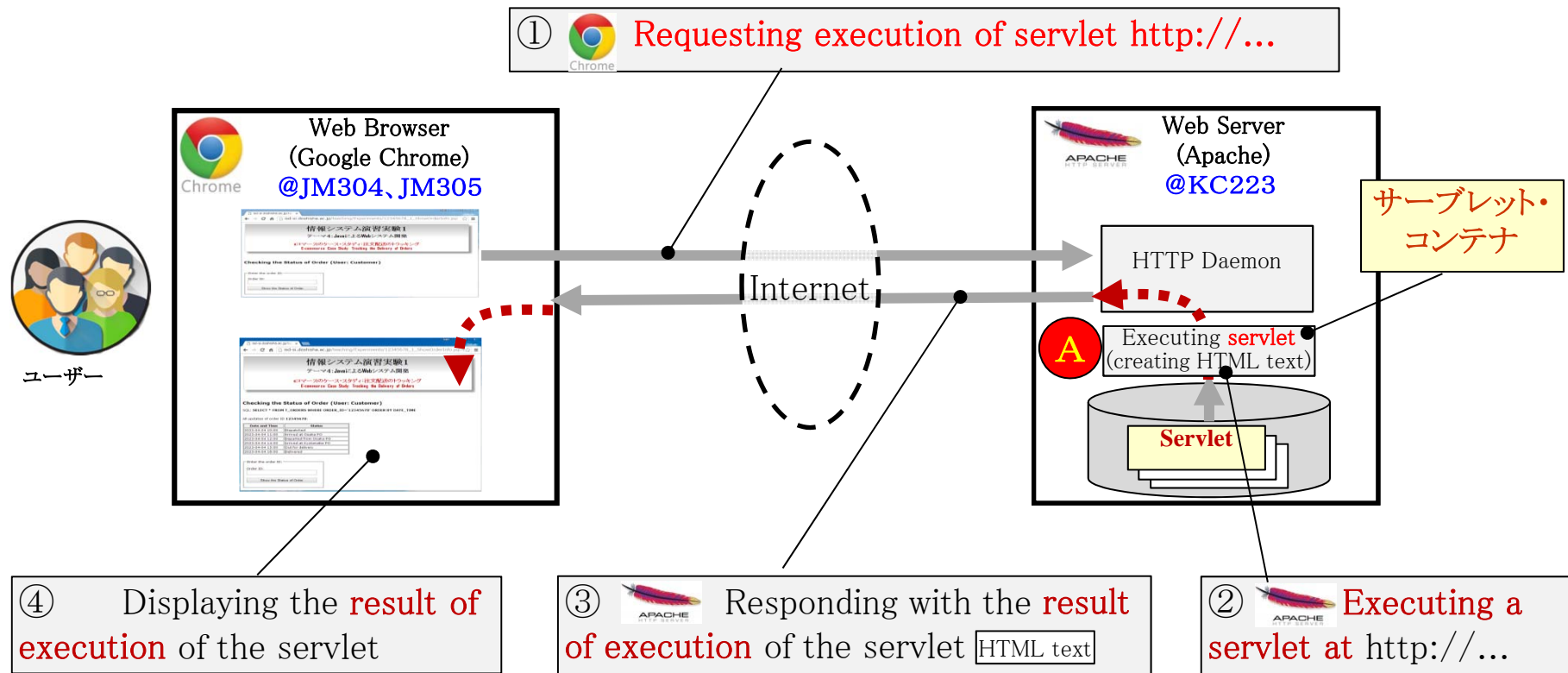
### 3. ダイナミック (動的)Webコンテンツとサーブレット



Q: サーブレットの主な特徴は何ですか?

A: 次のスライドに参考。

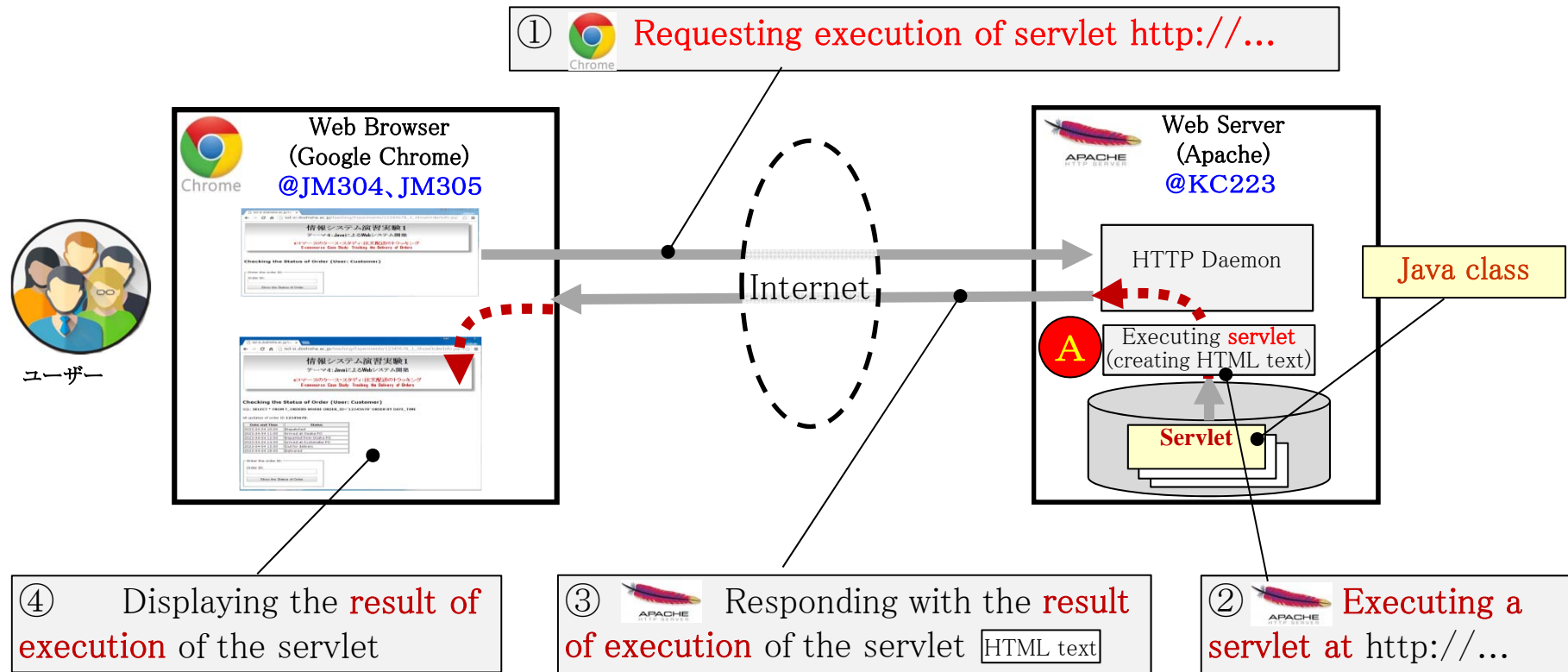
### 3. ダイナミック (動的)Webコンテンツとサーブレット



#### サーブレットの特徴 (1/2)

- Sun Microsystemsで開発されています。
- サーブレットはサーバサイドスクリプトである。
- 実行されている所は:  
サーブレット・コンテナおよびサーブレット・エンジンと呼ばれています。
- このスクリプトはJava言語で開発されています。

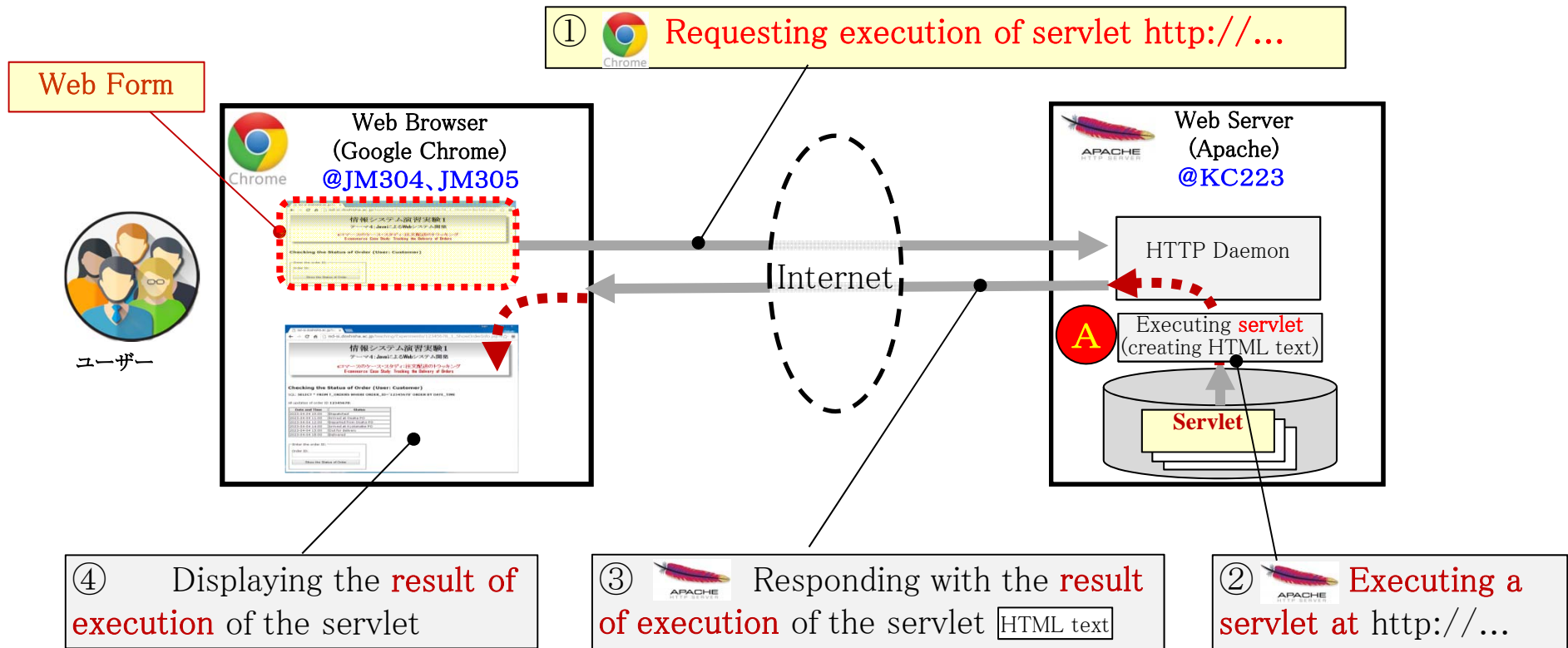
### 3. ダイナミック (動的)Webコンテンツとサーブレット



#### サーブレットの特徴 (2/2)

- Java言語:サーブレットはwrite once, run anywhereのJava classです。
- マルチスレッド
- サーブレットの実行されたダイナミックな結果は:
  - ユーザ入力とWebサーバのダイナミックな状態が関係ある
  - ユーザ入力とDBのダイナミックな状態が関係ある

### 3. ダイナミック (動的)Webコンテンツとサーブレット



Q: Webブラウザからサーブレットを呼び出す方法は？

A: **Web Form**からサーブレットを呼び出す(次のスライドに参考)。

## 4. Webフォーム

http://tangorin.com/

1

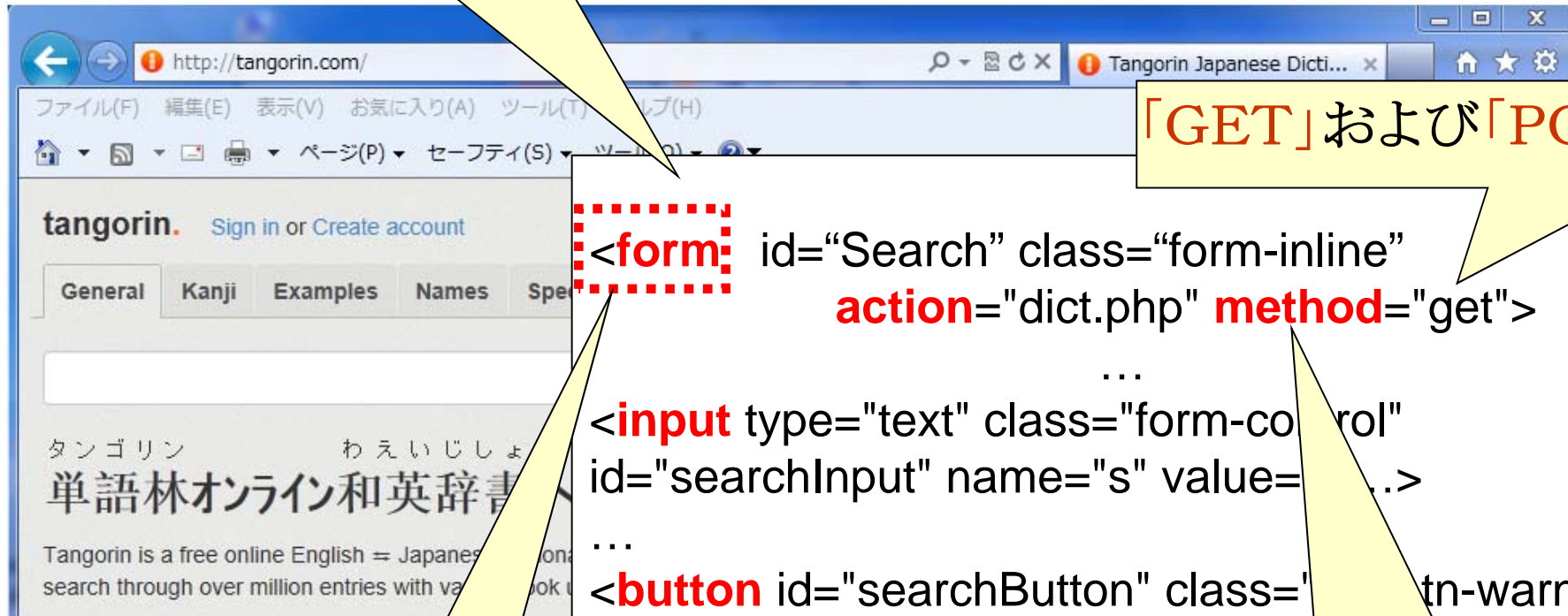
2 「Enter」キーを押す

3 「表示」から「ソース(C)」を開いて下さい。

The screenshot shows a web browser window with the URL <http://tangorin.com/> in the address bar. The browser's menu bar includes options like 'ファイル(F)', '編集(E)', '表示(V)', 'お気に入り(A)', 'ツール(T)', and 'ヘルプ(H)'. The website content includes a search bar with a 'Search' button, a 'Settings' panel with several checked options, and a header with the text 'タンゴリン わえいじしょ 単語林オンライン和英辞書へようこ'. Three red callout boxes with numbers 1, 2, and 3 are overlaid on the image. Callout 1 points to the address bar, callout 2 points to the search button, and callout 3 points to the '表示' menu item.

## 4. Webフォーム

ページのソース



「GET」および「POST」

```
<form id="Search" class="form-inline"
      action="dict.php" method="get">
```

```
...
<input type="text" class="form-control"
       id="searchInput" name="s" value="...">
```

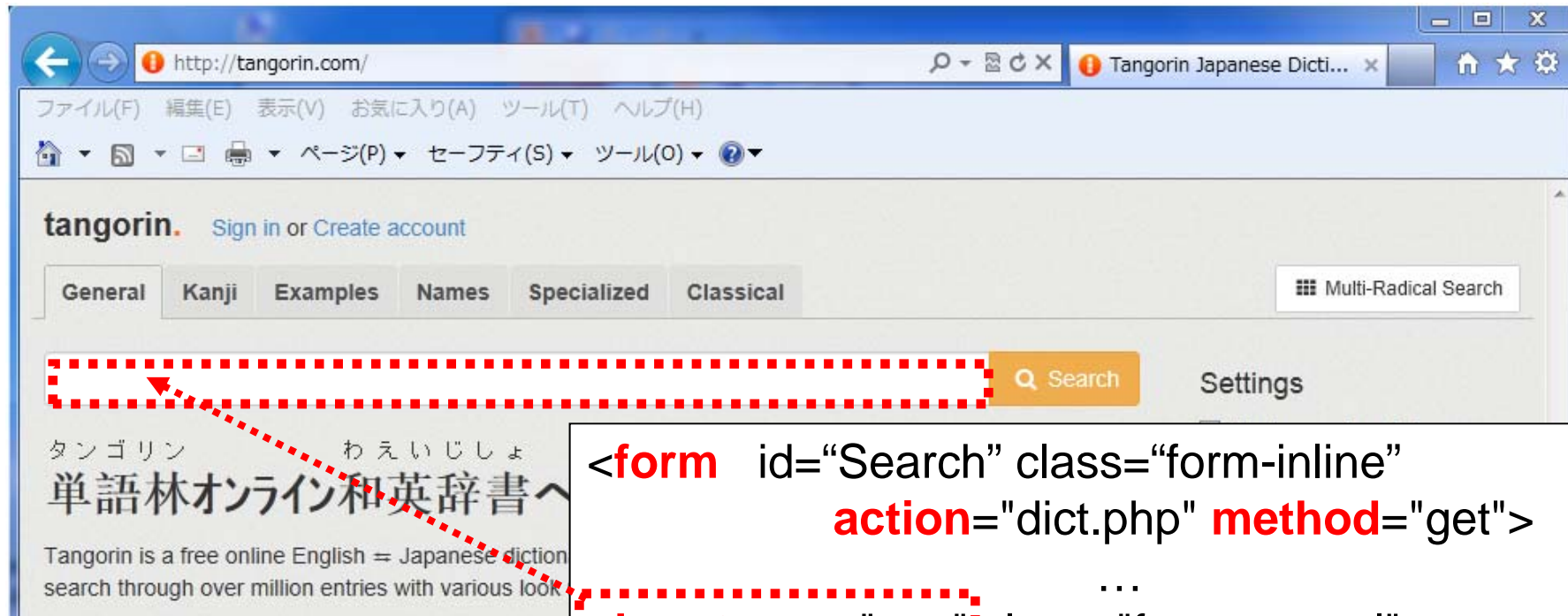
```
...
<button id="searchButton" class="btn btn-warning"
        tabindex="3" type="submit">
```

```
</form>
```

「FORM」はHTML  
タグ名である

「METHOD」はHTMLパラメーター名  
である

## 4. Webフォーム



```
<form id="Search" class="form-inline"
      action="dict.php" method="get">
```

```
...
<input type="text" class="form-control"
       id="searchInput" name="s" value="" ...>
```

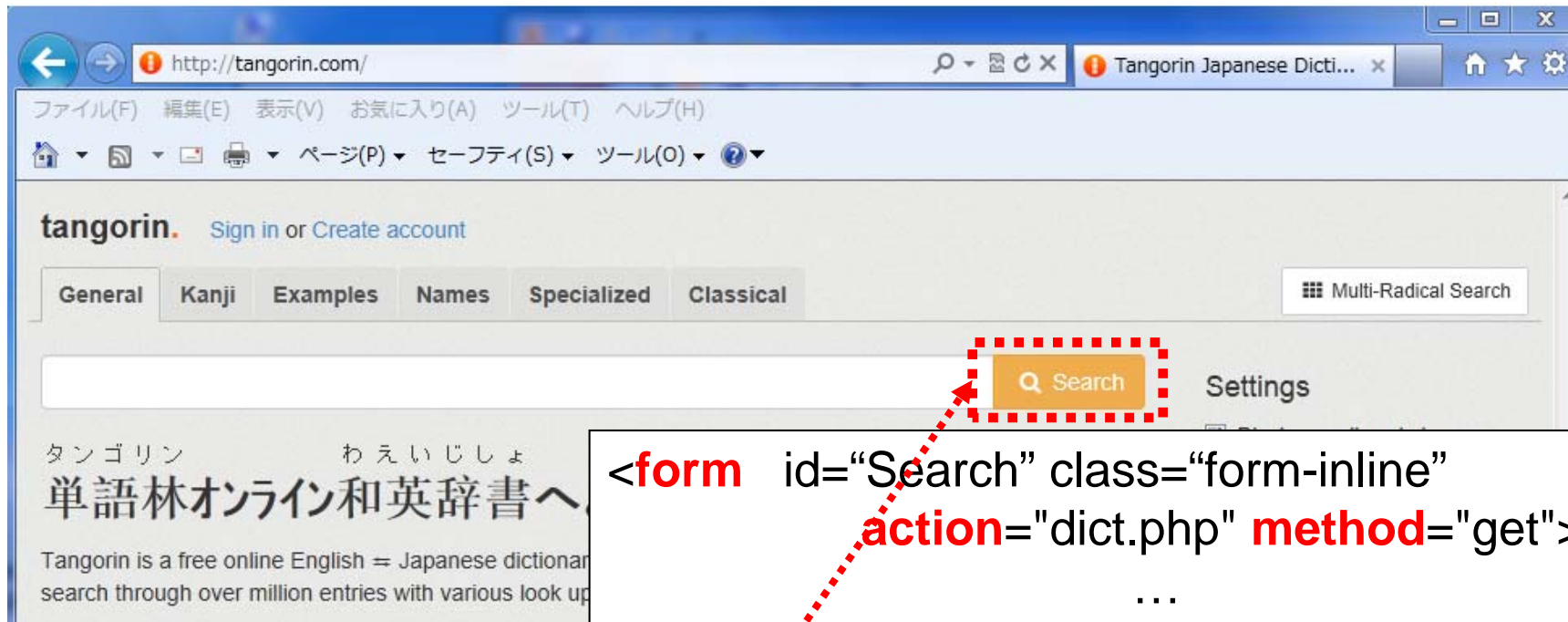
```
...
<button id="searchButton" class="btn btn-warning"
        tabindex="3" type="submit">
```

```
...
</form>
```

「INPUT」はHTML  
タグ名である

「TYPE」、「ID」はHTMLの  
「INPUT」タグのパラメター  
名である

## 4. Webフォーム



```
<form id="Search" class="form-inline"  
      action="dict.php" method="get">
```

```
...  
<input type="text" class="form-control"  
       id="searchInput" name="s" value="" ...>
```

```
...  
<button id="searchButton" class="btn btn-warning"  
        tabindex="3" type="submit">
```

```
</form>
```

「**BUTTON**」はHTML  
タグ名である

「**TYPE**」, 「**ID**」はHTMLの  
「**BUTTON**」タグのパラメーター名である

## 4. Webフォーム



「ACTION」はHTMLパラメーター名である

```
<form id="Search" class="form-inline"
      action="dict.php" method="get">
```

```
<input type="text" class="form-control"
       id="searchInput" name="s" value="" ...>
```

```
<button id="searchButton" class="btn btn-warning"
        tabindex="3" type="submit">
```

```
</form>
```

「searchButton」ボタンを押すとformの「searchInput」のデータはこの「dict.php」のスキriptへ送られています。このスキriptはServer Sideで実行されています。

## 4. Webフォーム




```
<form id="Search" class="form-inline"
      action="dict.php" method="get">
  ...
  <input type="text" class="form-control"
        id="searchInput" name="s" value="" ...>
  ...
  <button id="searchButton" class="btn btn-warning"
         tabindex="3" type="submit">
  ...
</form>
```

Webフォームには下記の特徴が定義されています:

- 何時(何のイベントの時に)サーバサイドスクリプトを実行するか? A: `<button id="a" ... type="submit">`
- 何のユーザー入力されたパラメータはスクリプトにおくるか? A: `<input id="b" ...>` 何のサーバサイドスクリプトを実行するか? A: `<form action="c" ...>`

# 5. 演習

A. Servletを実行するため、Webフォームを作り出す  
(  Notepadで)

notepad

B. WebフォームをWebサーバにUpload  
(  FTP Clientで)



C. Webフォームのアクセス  
(  Google Chromeで)

Chrome

D. サーブレットのソース(java)を確認  
(  Notepadで)

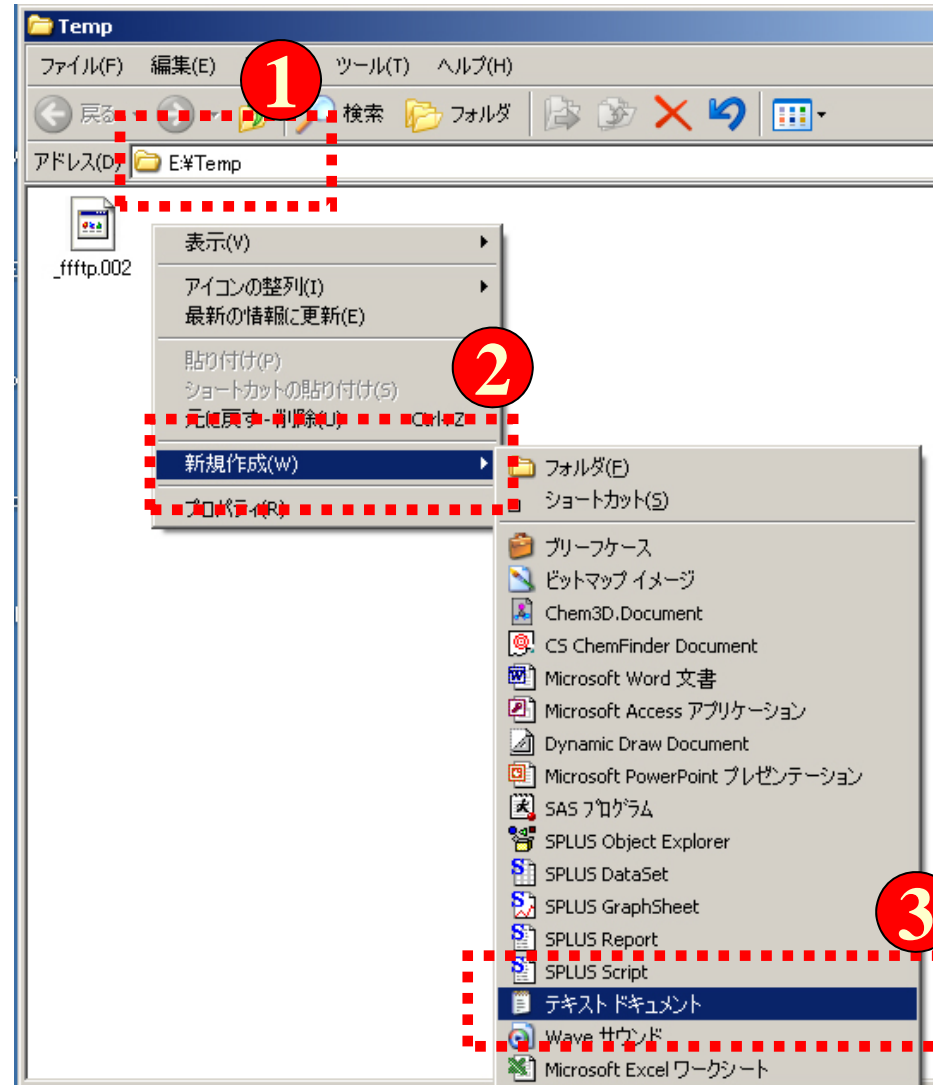
notepad

# 5. 演習

## A. Webフォームを作り出す

 **Notepad**で

notepad



# 5. 演習

## A. Webフォームを作り出す



notepad

Notepadで

Webフォームのソース:

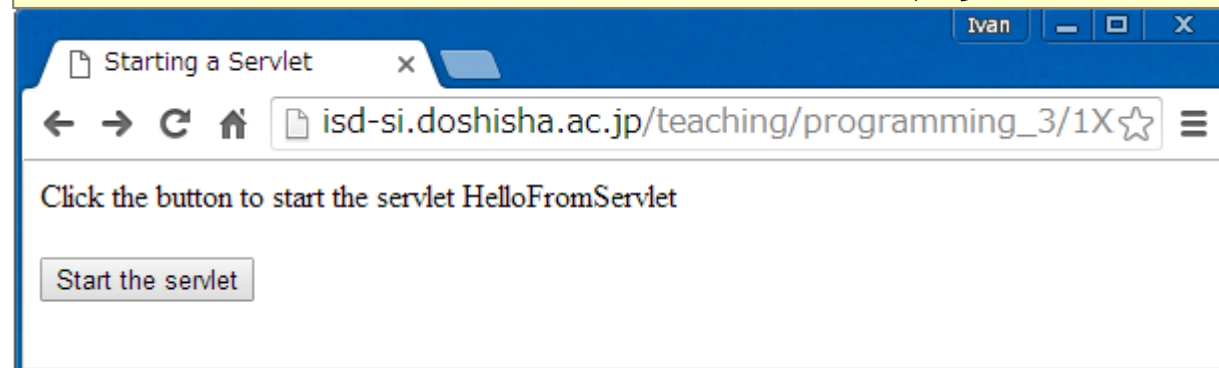
```
Lecture_2_Form - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
<HTML>
<HEAD><TITLE>Starting a Servlet</TITLE></HEAD>
<BODY>
  <FORM ACTION = "/servlet/HelloFromServlet" METHOD = "get">
    <P><LABEL>Click the button to start the servlet HelloFromServlet<BR><BR>
      <INPUT TYPE = "submit" VALUE = "Start the servlet" />
    </LABEL></P>
  </FORM>
</BODY>
</HTML>
```



Webフォームには下記の特徴が定義されています:

- 何時(何のイベントの時に)サーバサイドスクリプトを実行するか? A: **<input type="submit" ...>**
- 何のユーザー入力されたパラメータはスクリプトにおくるか? A: **<input id= " " ...>**
- 何のサーバサイドスクリプトを実行するか? A: **<form action=" " ...>**

WebフォームのWeb-browser上の表現:



# 5. 演習

## A. Webフォームを作り出す

 **Notepad**で

notepad

Virtual path of **HelloFromServlet.class** (URL): **/servlet/**

Physical path (@ Web Server): **C:\ORACLE\ora9.2\Apache\Jserv\servlets\**

```
Lecture_2_Form - メモ...
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
<HTML>
<HEAD><TITLE>Starting a Servlet</TITLE></HEAD>
<BODY>
  <FORM ACTION = "/servlet/HelloFromServlet" METHOD = "get">
    <P><LABEL>Click the button to start the servlet HelloFromServlet<BR><BR>
    <INPUT TYPE = "submit" VALUE = "Start the servlet" />
  </LABEL></P>
</FORM>
</BODY>
</HTML>
```

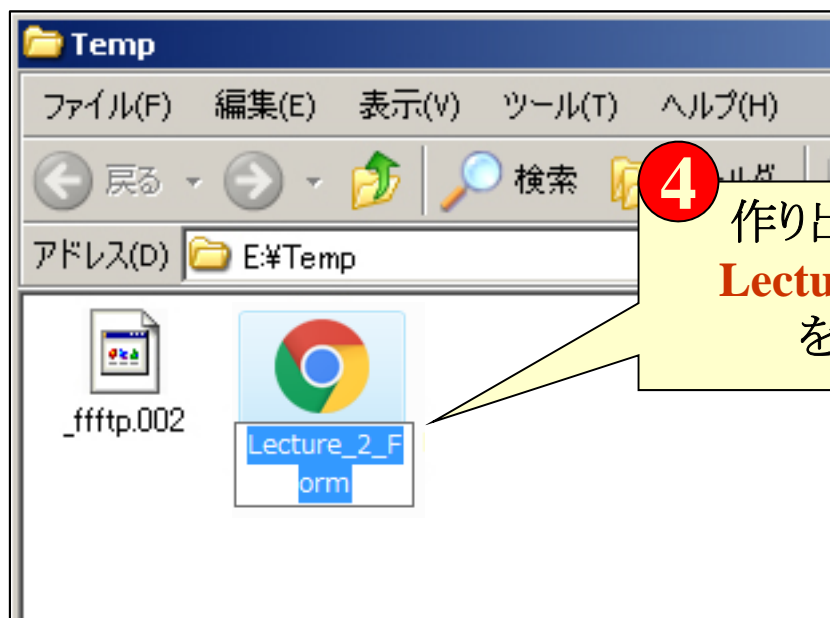
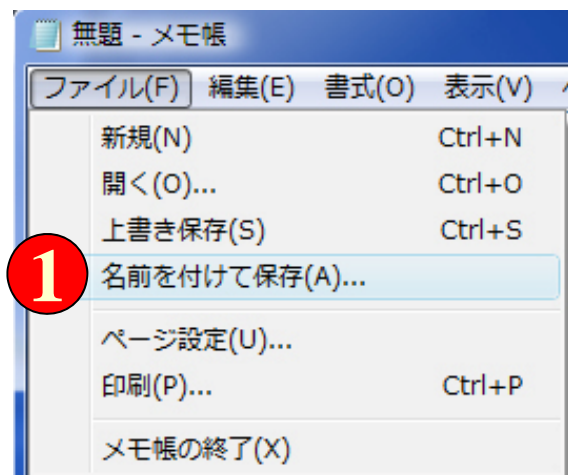
実行されたサーブレット  
**HelloFromServlet.class**  
のコンパイルされたJava Class  
(@ Web Server)

C:\ORACLE\ora9.2\Apache\Jserv\servlets

名前	種類	更新日時
zone.properties.default	DEFAULT ファイル	2004/05/11 9:41
zone.properties	KB Temporary file	2002/04/18 22:03
zone	KB PROPERTIES ファイル	2004/05/11 9:55
Time	1 KB JSP ファイル	2005/04/20 17:32
test.class	2 KB CLASS ファイル	2005/04/20 16:17
IsItWorking.class	2 KB CLASS ファイル	2002/08/20 21:42
IsItWorking	5 KB JAVA ファイル	2002/08/20 21:42
<b>HelloFromServlet.class</b>	<b>2 KB CLASS ファイル</b>	<b>2005/04/20 19:29</b>
HelloFromServlet	2 KB JAVA ファイル	2005/04/20 17:55
AQPropServlet22	3 KB JAVA ファイル	2004/05/11 9:55
AQPropServlet	3 KB .JAVA ファイル	2004/05/11 9:55

# 5. 演習

## A. Webフォームを作り出す



**4** 作り出したファイル名は **Lecture\_2\_Form.html** を確認します。

# 5. 演習

## B. WebフォームをWebサーバにUpload



FTP Clientで

「ホストの設定」「基本」のページでFTPサーバーのパラメーターを記入。  
ホスト名④は「isd-si.doshisha.ac.jp」で、ユーザ名⑤とパスワード⑥は「guest」です。

The screenshot shows the FFTP Client interface. A red circle 1 highlights the menu bar. A 'Host List' dialog box is open, with a red circle 2 around the '新規ホスト(N)...' button. The 'Host Settings' dialog box is also open, showing the '基本' tab. Red circles 3, 4, 5, 6, and 7 highlight the following fields: 'ホストの設定名(T)' (Host Name) with value 'guest', 'ホスト名(アドレス)(N)' (Host Name (Address)) with value 'isd-si.doshisha.ac.jp', 'ユーザー名(U)' (User Name) with value 'guest', 'パスワード/パスフレーズ(P)' (Password/Passphrase) with masked characters, and the 'OK' button.

# 5. 演習

## B. WebフォームをWebサーバにUpload



FTP Clientで

接続してから  
isd-si.doshisha.ac.jp  
のサーバー  
/teaching/programming\_3/  
のフォルダを表示されています。

名前	日付	サイズ	種類	属性	所有者
1116201001	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201002	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201003	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201004	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201005	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201006	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201008	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201009	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201011	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201012	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201013	2021/04/06 8:42	<DIR>		rw-rw-r--	root
1116201014	2021/04/06 8:42	<DIR>		rw-rw-r--	root

150 Opening ASCII mode data connection for /bin/ls (4286 bytes).  
226 Transfer successful.  
ファイル一覧の取得は正常終了しました。(4286 Bytes)

保護されていません ローカル 選択0個 (0B) ローカル空 7.97GB 転送待ちファイル0個

自分のフォルダを選択します

# 5. 演習

## B. WebフォームをWebサーバにUpload



FTP Clientで

Webフォームファイル

E:\temp\Lecture\_2\_Form.html

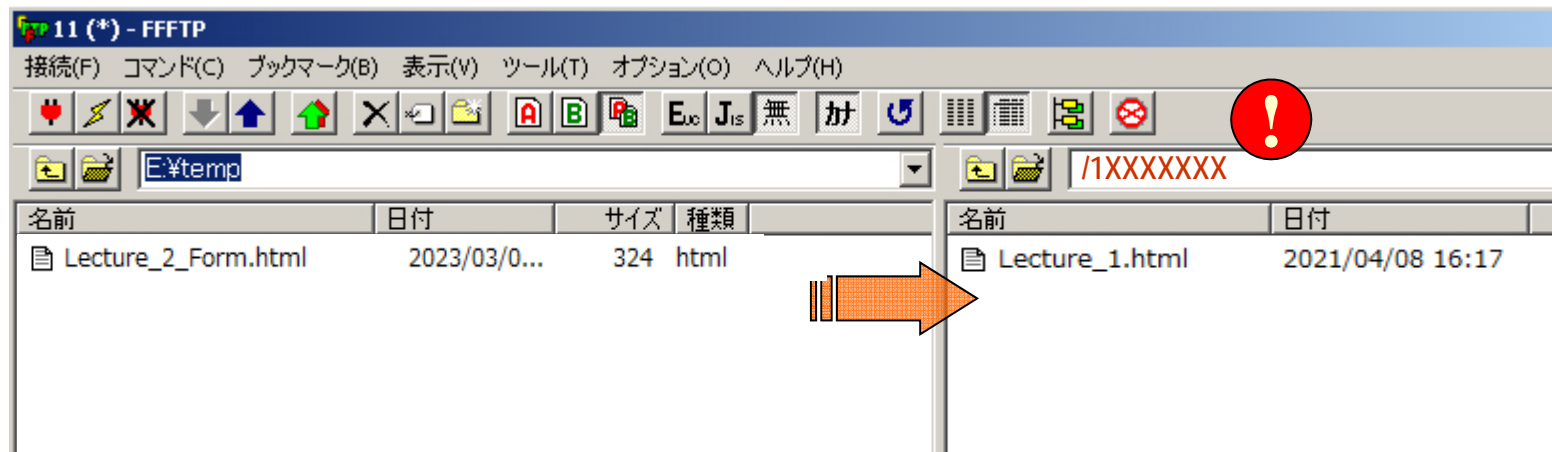
Webサーバのフォルダー

/1XXXXXXXX

にUploadします。



注意: 1XXXXXXXX = 自分の学籍番号



# 5. 演習

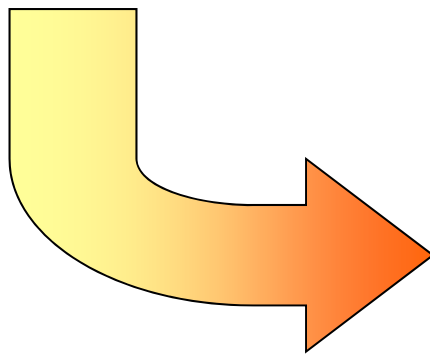
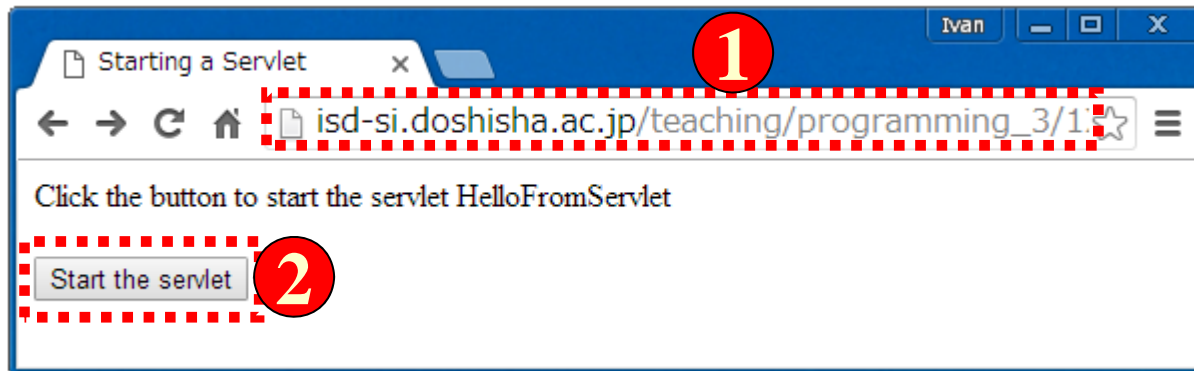
## C. Webフォームのアクセス



Google Chrome, or Internet Explorerで

① URL: [http://isd-si.doshisha.ac.jp/teaching/programming\\_3/1XXXXXXX/Lecture\\_2\\_Form.html](http://isd-si.doshisha.ac.jp/teaching/programming_3/1XXXXXXX/Lecture_2_Form.html)

注意: 1XXXXXXX = 自分の学籍番号



# 5. 演習

## D. サブレットのソース(java)を確認



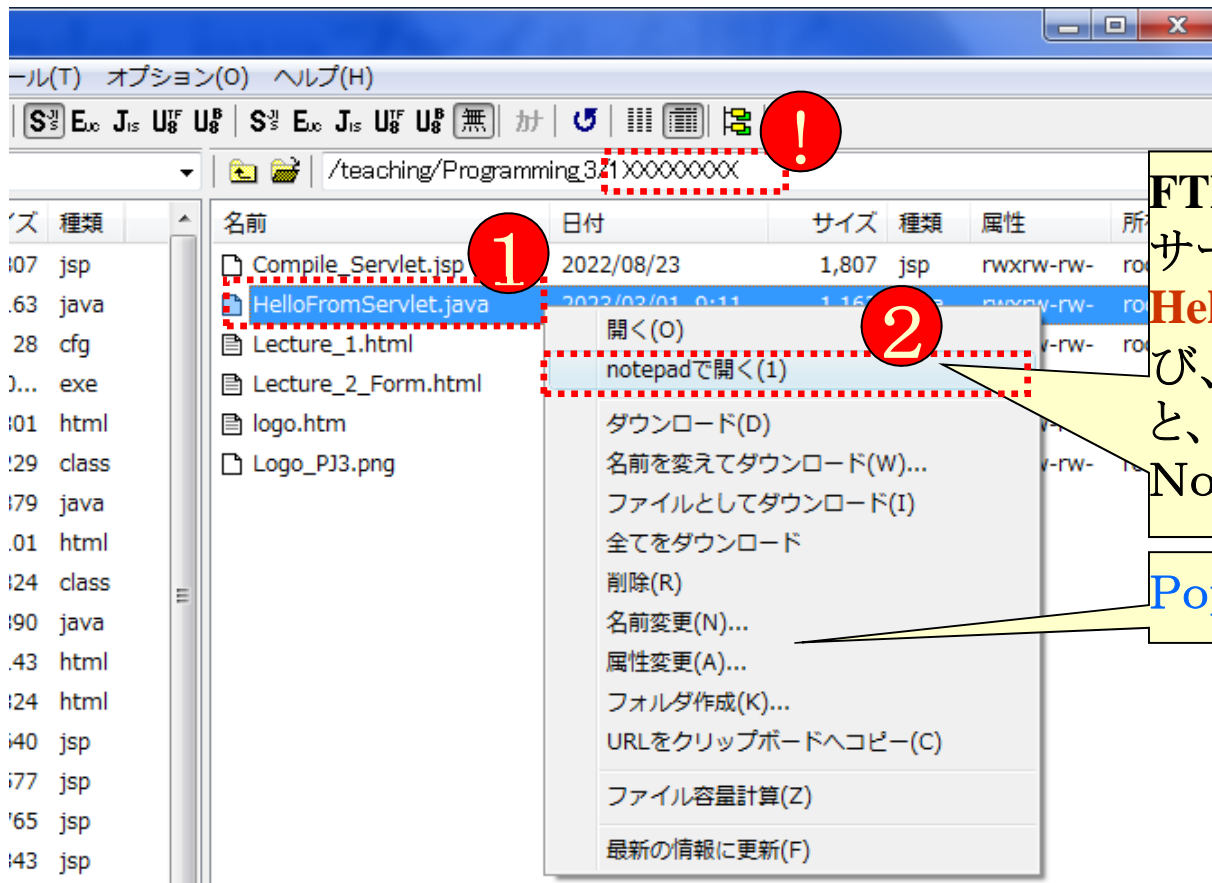
FTP Client の右のパネルに表示された HelloFromServlet.java ファイルを開く (Notepadで)。



notepad



注意: 1XXXXXXXX =自分の学籍番号



FTPクライアント  
サブレットソースファイル  
**HelloFromServlet.java**を  
選び、マウスの右ボタンを  
押すと、Pop-upメニューが  
出ます。Notepadで開  
きます。

Pop-upメニュー

# 5. 演習

## D. サーブレットのソース(java)を確認

サーブレット **HelloFromServlet** のJavaソース

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloFromServlet extends HttpServlet {

    public void service (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        // set content type and other response header fields first
        response.setContentType("text/html");

        // get the communication channel with the requesting client
        PrintWriter out = response.getWriter();

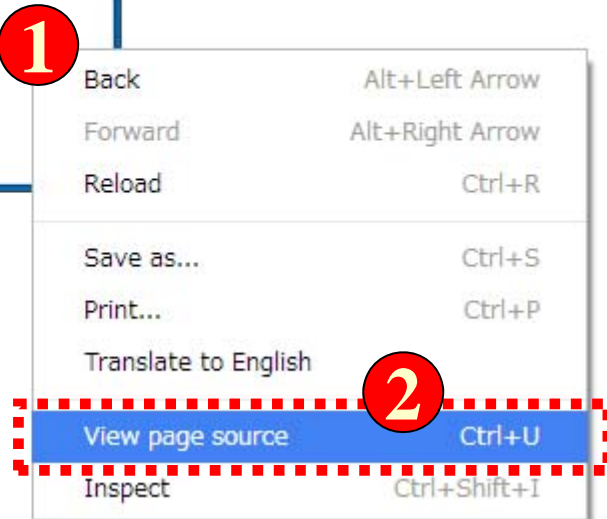
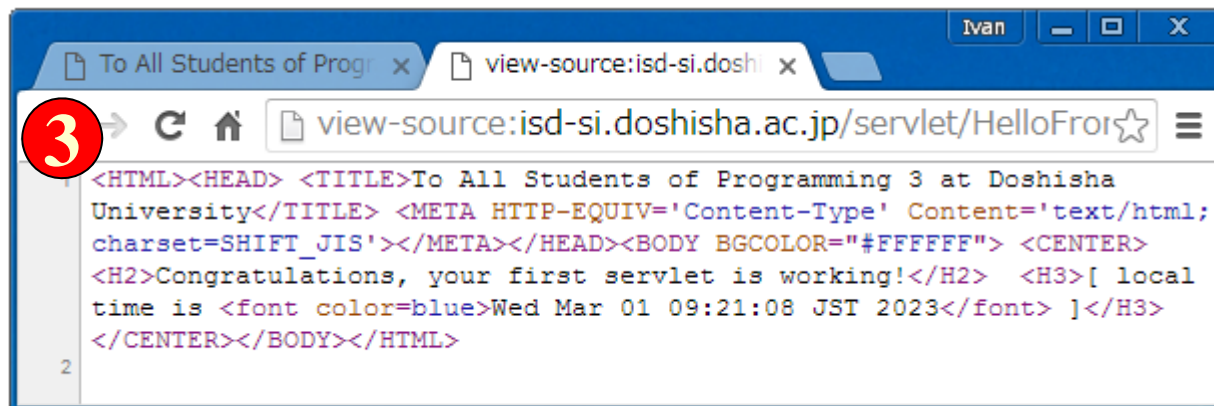
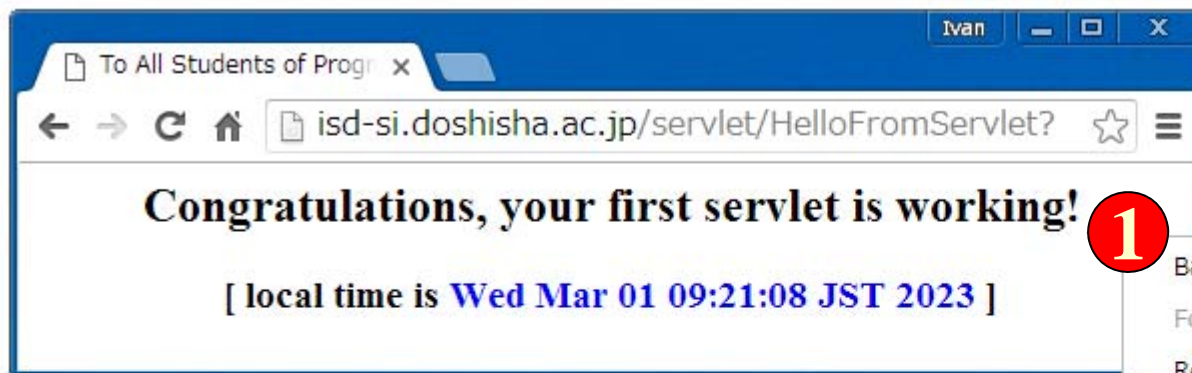
        // write the data
        out.println("<HTML>"
            + "<HEAD>"
            + " <TITLE>To All Students of Programming 3 at Doshisha University</TITLE>"
            + " <META HTTP-EQUIV='Content-Type' Content='text/html; charset=SHIFT_JIS'></META>"
            + "</HEAD>"
            + "<BODY BGCOLOR=\"#FFFFFF\">"
            + " <CENTER>"
            + " <H2>Congratulations, your first servlet is working!</H2>"
            + " <H3>[ local time is <font color=blue>"
            + new java.util.Date()
            + "</font> ]</H3>"
            + " </CENTER>"
            + "</BODY>"
            + "</HTML>");
    }
}
```

# 5. 演習

## D. サーブレットのソース (java) を確認

サーブレットの実行結果を検査します:

- 1 Web Browserの画面にマウスの右ボタンをクリックします、
- 2 Pop-upメニューから:ソースを示す、
- 3 Pageのソース(サーブレットHelloFromServletの実行結果)を確認します。



# 5. 演習

## D. サーブレットのソース(java)を確認

### サーブレットHelloFromServletのJavaソース

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

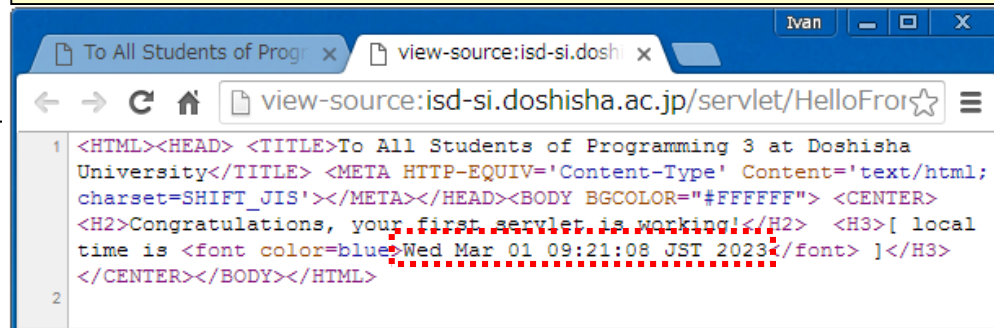
public class HelloFromServlet extends HttpServlet {

    public void service (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        // set content type and other response header fields first
        response.setContentType("text/html");

        // get the communication channel with the requesting client
        PrintWriter out = response.getWriter();

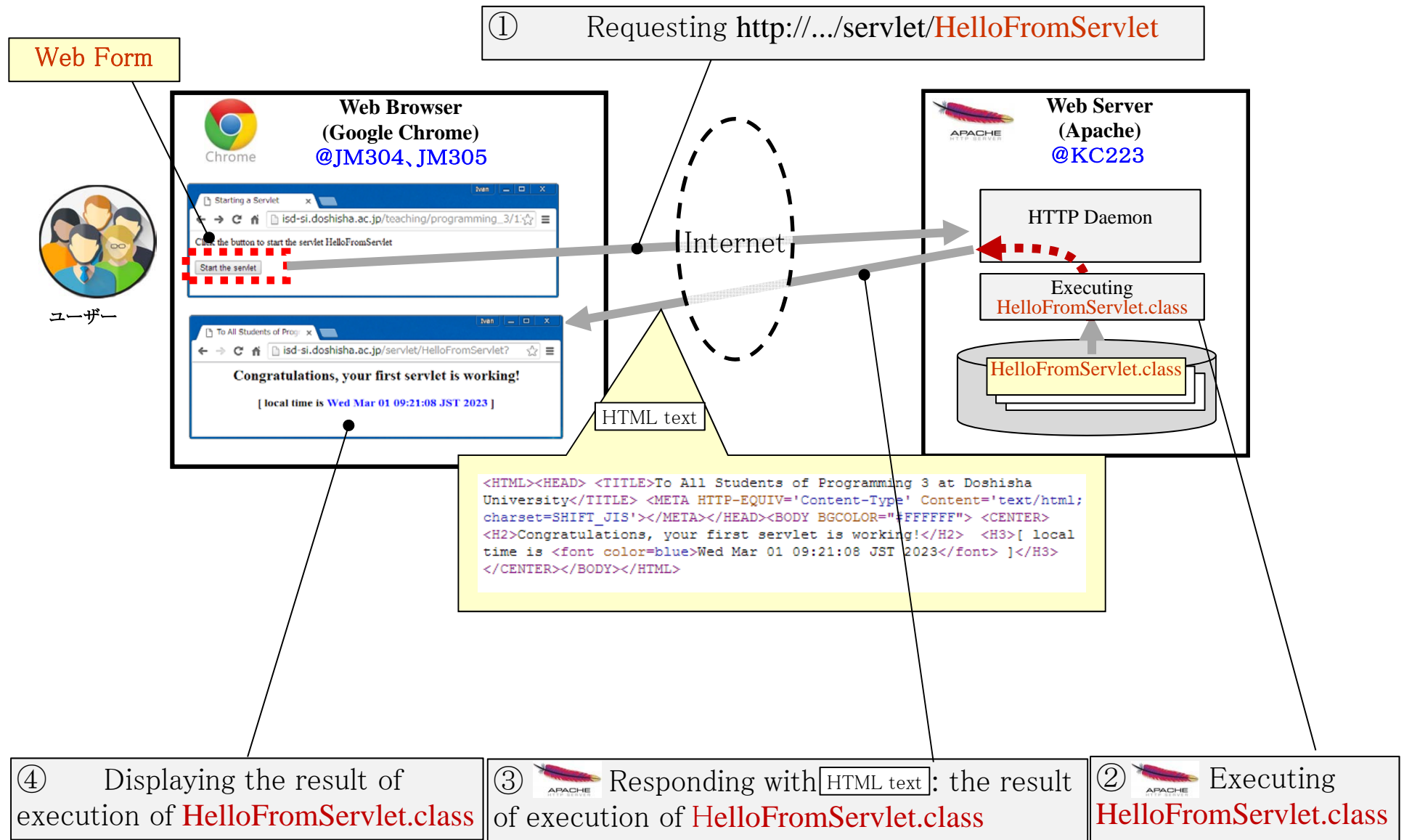
        // write the data
        out.println("<HTML>"
            + "<HEAD>"
            + " <TITLE>To All Students of Programming 3 at Doshisha University</TITLE>"
            + " <META HTTP-EQUIV='Content-Type' Content='text/html; charset=SHIFT_JIS'></META>"
            + "</HEAD>"
            + "<BODY BGCOLOR=\"#FFFFFF\">"
            + " <CENTER>"
            + " <H2>Congratulations, your first servlet is working!</H2>"
            + " <H3>[ local time is <font color=blue>"
            + "new java.util.Date()"
            + "</font> ]</H3>"
            + " </CENTER>"
            + " </BODY>"
            + "</HTML>");
    }
}
```

### サーブレットHelloFromServletの実行結果



```
1 <HTML><HEAD> <TITLE>To All Students of Programming 3 at Doshisha
University</TITLE> <META HTTP-EQUIV='Content-Type' Content='text/html;
charset=SHIFT_JIS'></META></HEAD><BODY BGCOLOR="#FFFFFF"> <CENTER>
<H2>Congratulations, your first servlet is working!</H2> <H3>[ local
time is <font color=blue>Wed Mar 01 09:21:08 JST 2023</font> ]</H3>
</CENTER></BODY></HTML>
2
```

# 5. 演習 (まとめ)



# 3. 演習

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloFromServlet extends HttpServlet {

    public void service (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        // set content type and other response header fields first
        response.setContentType("text/html");

        // get the communication channel with the requesting client
        PrintWriter out = response.getWriter();

        // write the data
        out.println("<HTML>"
            + "<HEAD>"
            + " <TITLE>To All Students of Programming 3 at Doshisha University</TITLE>"
            + " <META HTTP-EQUIV='Content-Type' Content='text/html; charset=SHIFT_JIS'></META>"
            + "</HEAD>"
            + "<BODY BGCOLOR=\"#FFFFFF\">"
            + " <CENTER>"
            + " <H2>Congratulations, your first servlet is working!</H2>"
            + " <H3>[ local time is <font color=blue>"
            + new java.util.Date()
            + "</font> ]</H3>"
            + " </CENTER>"
            + "</BODY>"
            + "</HTML>");
    }
}
```



Q: サーブレットの**構造**? Java Classの**開発とコンパイル方法**?

A: 第3回の講義に説明します。

# 講義の内容の理解のチェック・リスト (まとめ)

- ✓ ダイナミックWebコンテンツとサーブレットの関係、
- ✓ サーブレットの特徴、
- ✓ Web Formの特徴、Web Formからサーブレットを呼び出す(実行)方法:
  - 何時(何のイベントの時に)サーバサイドスクリプトを実行するか? A: `<input type="submit" ...>`
  - 何のサーバサイドスクリプトを実行するか? A: `<form action=" "...>`
  - 何のユーザー入力されたパラメータはサーブレットに通信? A: `<input id=" "...>`
- ✓ Web Formの  作成、 Upload、 アクセス など。

次の授業:

**Lecture #3:** サーブレットの構造、Java Classの開発とコンパイル方法など。

→ **Lecture #4:** 何のユーザー入力されたパラメータはサーブレットに通信?(A: `<input id=" "...>`)

**The End**