プログラミング Java 3

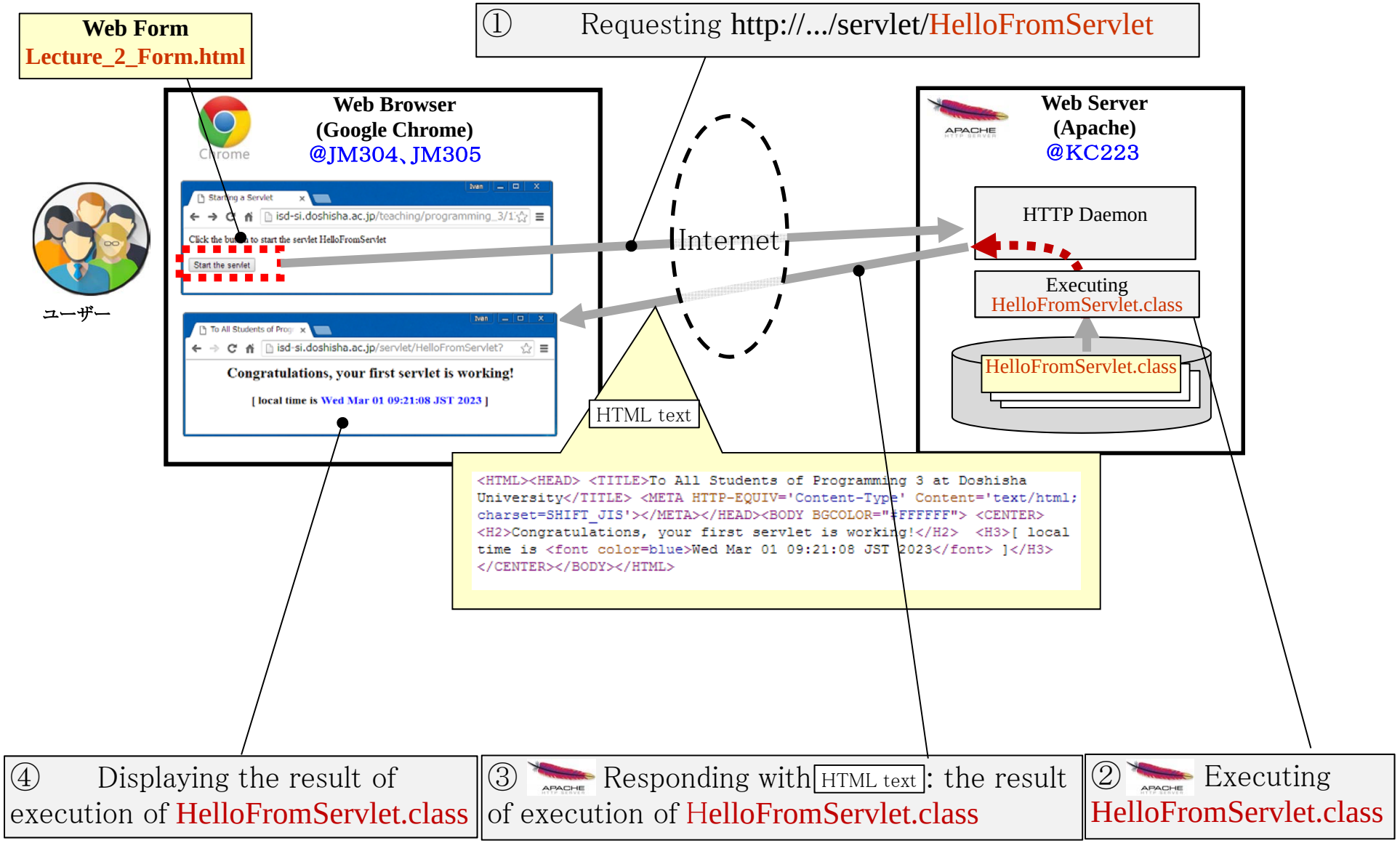
第3回:
サーブレットの構造入門:
HttpServletRequest & HttpServletResponse

Ivan Tanev

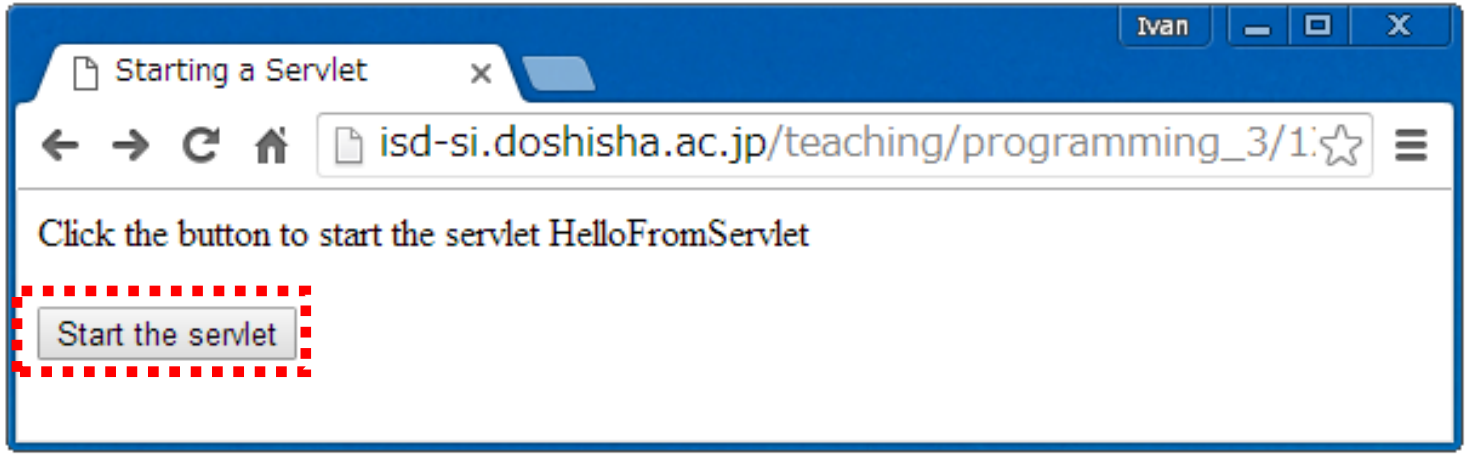
アウトライン

1. 第2回のまとめ
2. サーブレットの構造
3. HttpServletRequest & HttpServletResponse
4. 演習

1. 第2回のまとめ


Web Form
Lecture_2_Form.html
① Requesting http://.../servlet/HelloFromServlet
Web Browser
(Google Chrome)
@JM304、JM305
Chrome
Starting a Servlet
isd-si.doshisha.ac.jp/teaching/programming_3/1/
Click the button to start the servlet HelloFromServlet
Start the servlet
To All Students of Prog
isd-si.doshisha.ac.jp/servlet/HelloFromServlet?
Congratulations, your first servlet is working!
[local time is Wed Mar 01 09:21:08 JST 2023]
ユーザー
Internet
Web Server
(Apache)
@KC223
APACHE
HTTP Daemon
Executing
HelloFromServlet.class
HelloFromServlet.class
HTML text
<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 Wed Mar 01 09:21:08 JST 2023]</H3>
</CENTER></BODY></HTML>
④ Displaying the result of execution of HelloFromServlet.class
③ Responding with HTML text: the result of execution of HelloFromServlet.class
② Executing HelloFromServlet.class

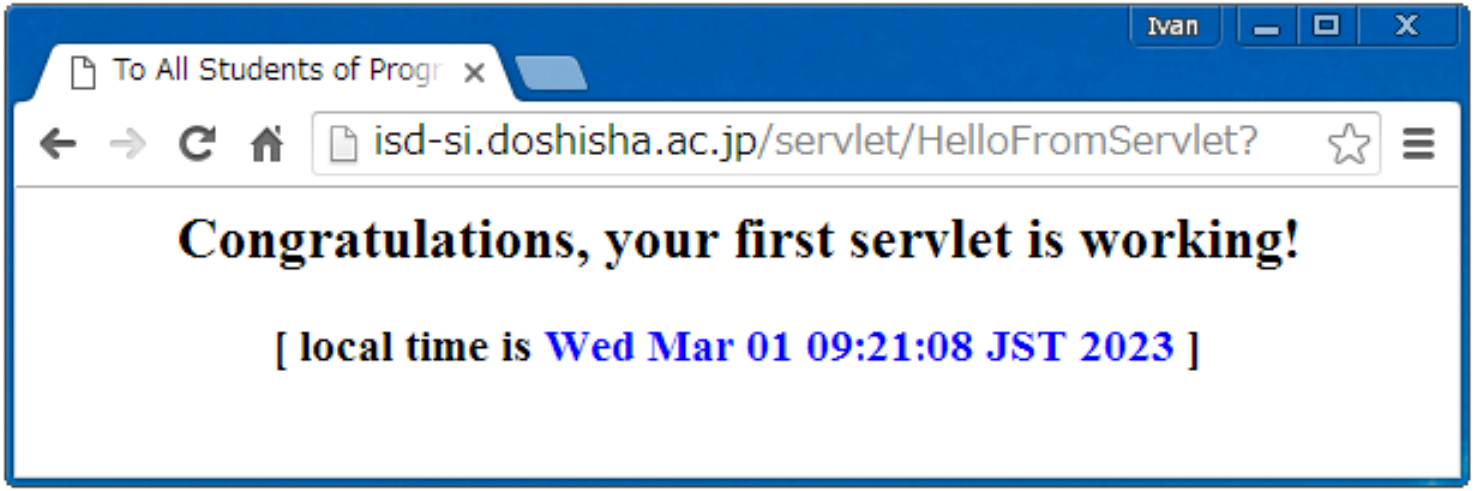
1. 第2回のまとめ



サーブレットHelloFromServlet呼び出す前（Lecture_2_Form.htmlの画面）

URL: http://isd-si.doshisha.ac.jp/teaching/programming_3/1XXXXXX/Lecture_2_Form.html

1XXXXXXX =自分の学籍番号



サーブレットHelloFromServlet呼び出す後

2. サーブレットの構造

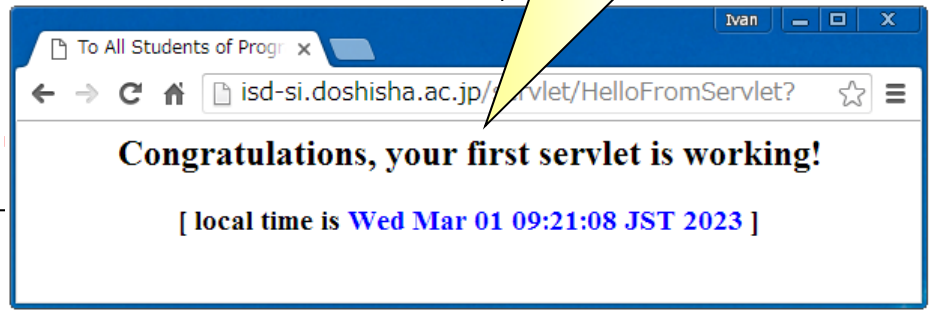
サーブレット**HelloFromServlet**のJavaソース

```
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 clien
        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>");
    }
}
```

Web Browserへの送られた
サーブレットの実行結果

```
<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>
```

Web Browserでの表示された
サーブレットの実行結果



2. サーブレットの構造

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 2 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>");
    }
}
```

Packages (classes and interfaces)

Abstract class of package javax.servlet.http
Default implementation of servlet methods

Method overridden

Request object

Request interface

Response object

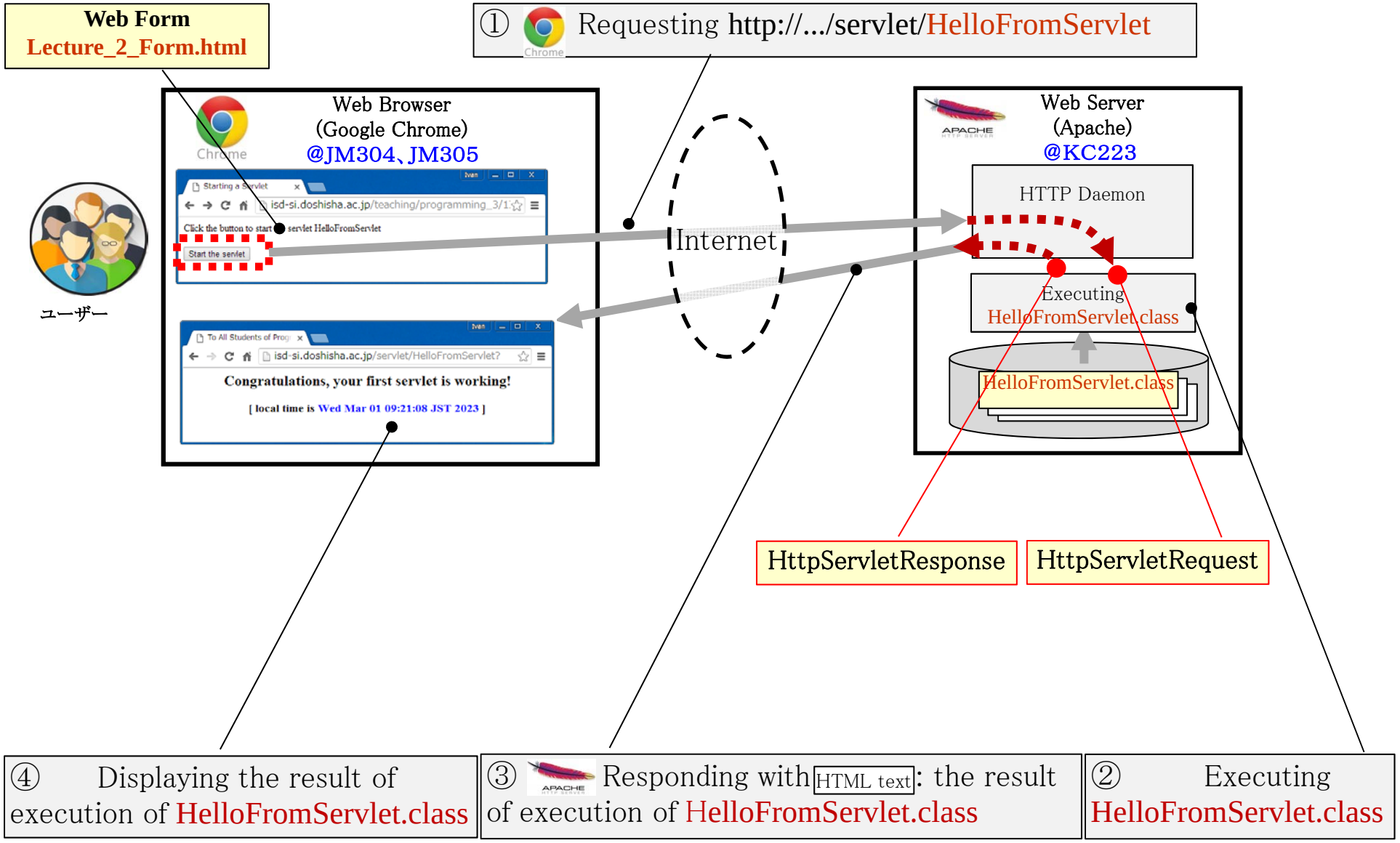
Response interface

Methods of **Response** object

Character-based output stream

Method of output stream

3. HttpServletRequest &HttpServletResponse



3. HttpServletRequest &HttpServletResponse

HttpServletResponseの処理

⚠ HttpServletRequestの処理：第4回の授業

javax.servlet

Interface ServletResponse

例：
```
public void service (
        HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException
```

All Known Subinterfaces:

HttpServletResponse

All Known Implementing Classes:

ServletResponseWrapper

---

public interface **ServletResponse**

Defines an object to assist a servlet in sending a response to the client. The servlet container creates a `ServletResponse` object and passes it as an argument to the servlet's `service` method.

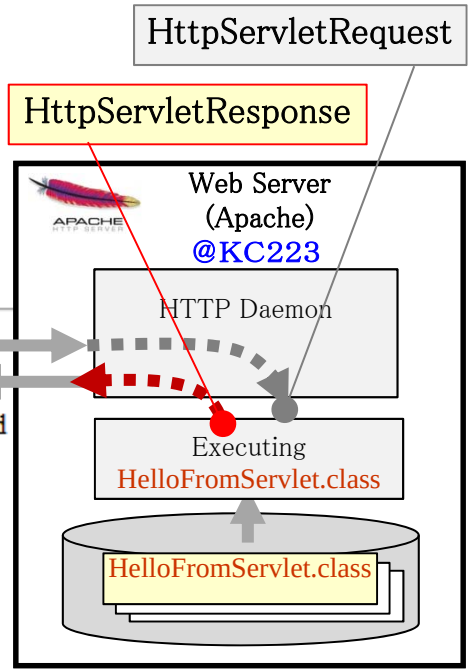
To send binary data in a MIME body response, use the `ServletOutputStream` returned by `getOutputStream()`. To send character data, use the `PrintWriter` object returned by `getWriter()`. To mix binary and text data, for example, to create a multipart response, use a `ServletOutputStream` and manage the character sections manually.

The charset for the MIME body response can be specified with `setContentType(java.lang.String)`. For example, "text/html; charset=Shift_JIS". The charset can alternately be set using `setLocale(java.util.Locale)`. If no charset is specified, ISO-8859-1 will be used. The `setContentType` or `setLocale` method must be called before `getWriter` for the charset to affect the construction of the writer.

See the Internet RFCs such as RFC 2045 for more information on MIME. Protocols such as SMTP and HTTP define profiles of MIME, and those standards are still evolving.

See Also:

`ServletOutputStream`

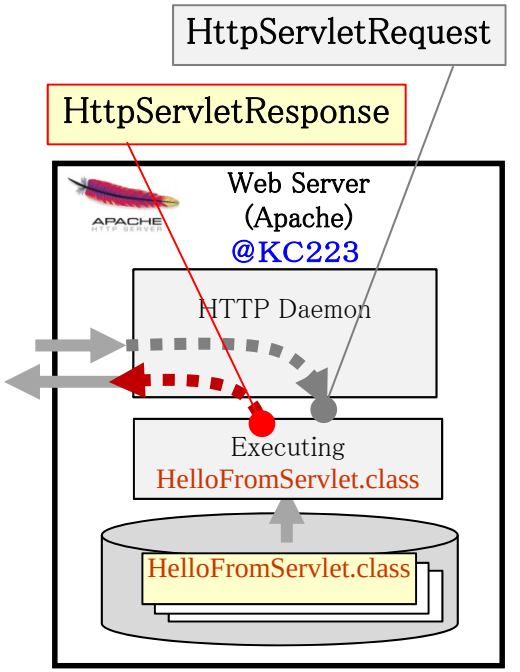


https://docs.oracle.com/javaee/1.3/api/javax/servlet/ServletResponse.html

3. HttpServletRequest &HttpServletResponse

Interface ServletResponse

Method Summary	
void	flushBuffer() Forces any content in the buffer to be written to the client.
int	getBufferSize() Returns the actual buffer size used for the response.
java.lang.String	getCharacterEncoding() Returns the name of the charset used for the MIME body sent in this response.
java.util.Locale	getLocale() Returns the locale assigned to the response.
ServletOutputStream	getOutputStream() Returns a ServletOutputStream suitable for writing binary data in the response.
java.io.PrintWriter	getWriter() Returns a PrintWriter object that can send character text to the client.
boolean	isCommitted() Returns a boolean indicating if the response has been committed.
void	reset() Clears any data that exists in the buffer as well as the status code and headers.
void	resetBuffer() Clears the content of the underlying buffer in the response without clearing headers or status code.
void	setBufferSize(int size) Sets the preferred buffer size for the body of the response.
void	setContentLength(int len) Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
void	setContentType(java.lang.String type) Sets the content type of the response being sent to the client.
void	setLocale(java.util.Locale loc) Sets the locale of the response, setting the headers (including the Content-Type's charset) as appropriate.




```
例:PrintWriter out = response.getWriter();
```

```
例:response.setContentType("text/html");
```

3. HttpServletRequest &HttpServletResponse

getWriter

```
public java.io.PrintWriter getWriter()
                            throws java.io.IOException
```

例:`PrintWriter out = response.getWriter();`

Returns a `PrintWriter` object that can send character text to the client. The character encoding used is the one specified in the `charset=` property of the `setContentType(java.lang.String)` method, which must be called *before* calling this method for the charset to take effect.

If necessary, the MIME type of the response is modified to reflect the character encoding used.

Calling flush() on the PrintWriter commits the response.

Either this method or `getOutputStream()` may be called to write the body, not both.

Returns:

a `PrintWriter` object that can return character data to the client

Throws:

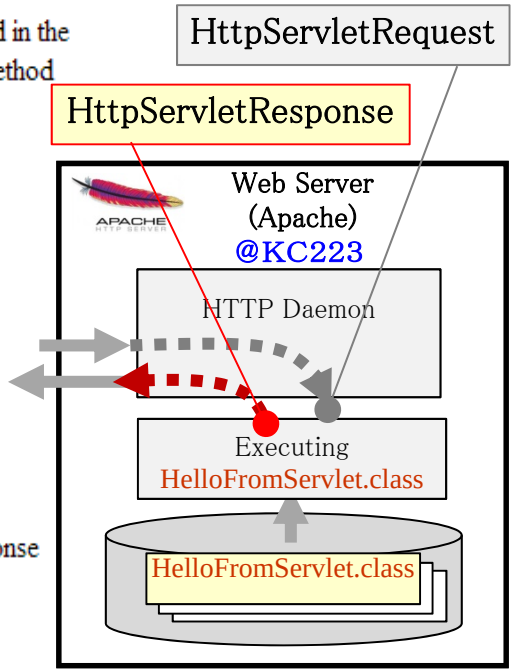
`java.io.UnsupportedEncodingException` - if the charset specified in `setContentType` cannot be used

`java.lang.IllegalStateException` - if the `getOutputStream` method has already been called for this response object

`java.io.IOException` - if an input or output exception occurred

See Also:

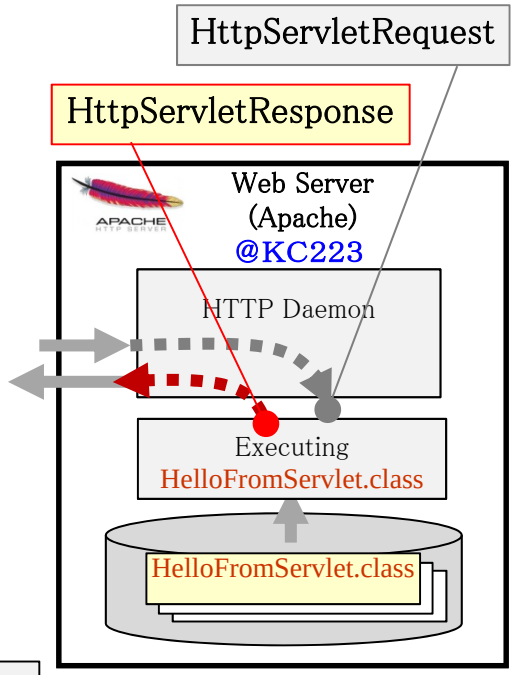
`getOutputStream()`, `setContentType(java.lang.String)`



3. HttpServletRequest &HttpServletResponse

Class PrintWriter, Methods (一部)

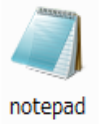
void	println() Terminates the current line by writing the line separator string.
void	println(boolean x) Prints a boolean value and then terminates the line.
void	println(char x) Prints a character and then terminates the line.
void	println(char[] x) Prints an array of characters and then terminates the line.
void	println(double x) Prints a double-precision floating-point number and then terminates the line.
void	println(float x) Prints a floating-point number and then terminates the line.
void	println(int x) Prints an integer and then terminates the line.
void	println(long x) Prints a long integer and then terminates the line.
void	println(Object x) Prints an Object and then terminates the line.
void	println(String x) Prints a String and then terminates the line.



例：HelloFromServlet.javaの部分

```
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>");
```

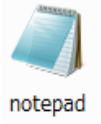
4. 演習

A. 

Webフォームの開発 (Notepad)

B. 

WebフォームをWebサーバへのUpload (FTP Client)

C. 

Servletの開発 (Notepad)

D. 

ServletをWebサーバへのUpload (FTP Client)

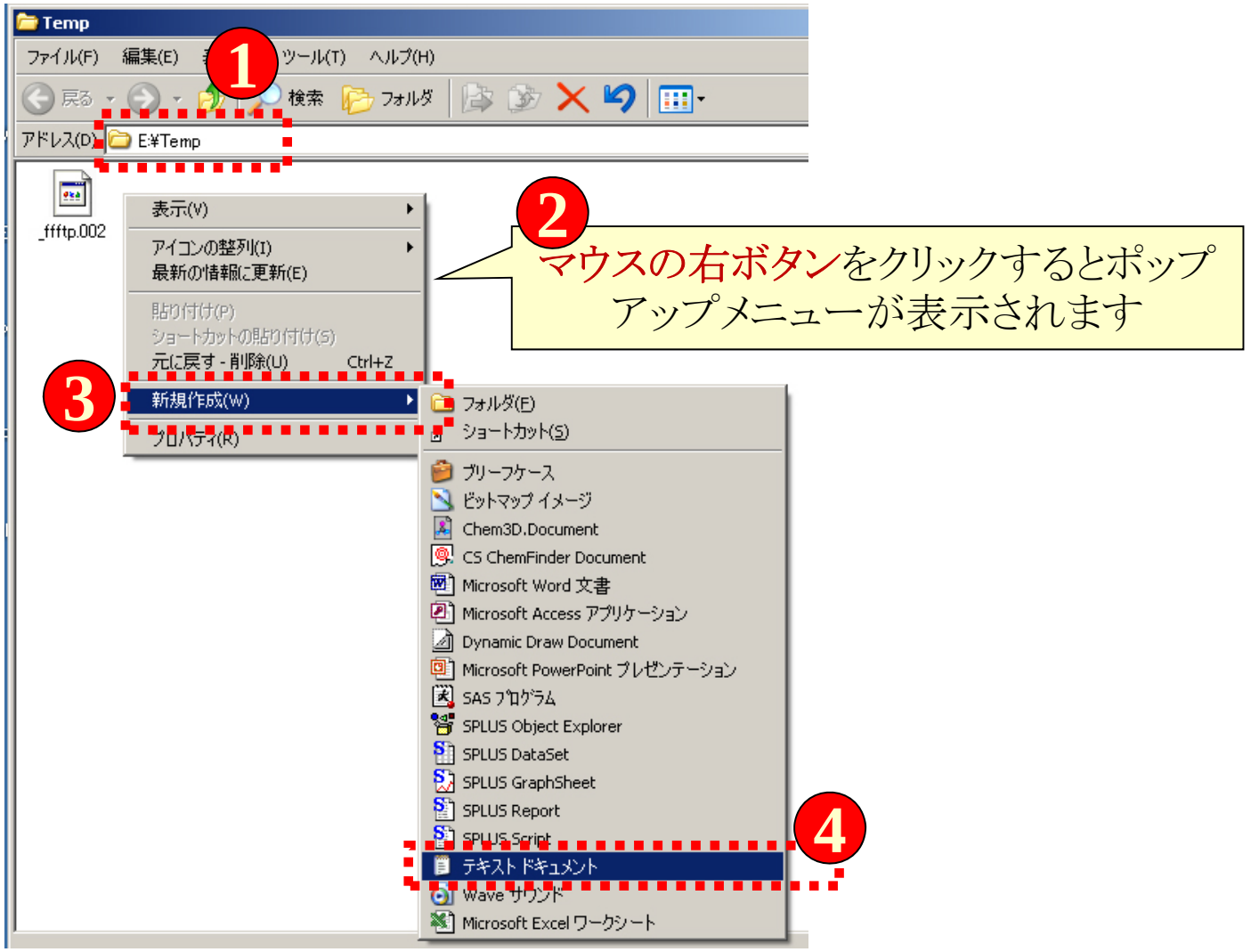
E. 

Servletのコンパイル (CompileServlet.jsp via Web Browser)

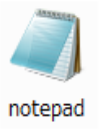
F. 

WebフォームのAccess (Web Browser)

A. Webフォームの開発

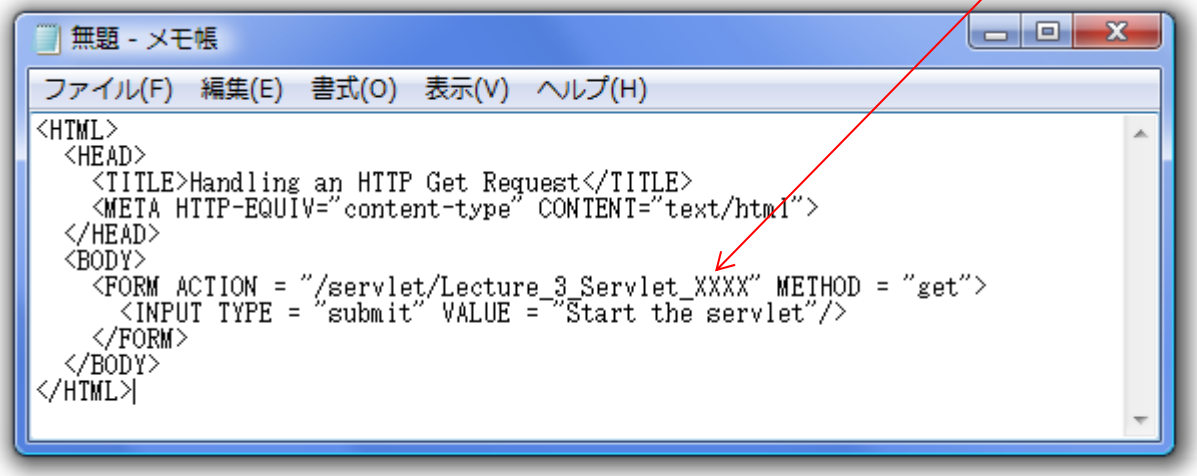


A. Webフォームの開発



下記の内容を入力:

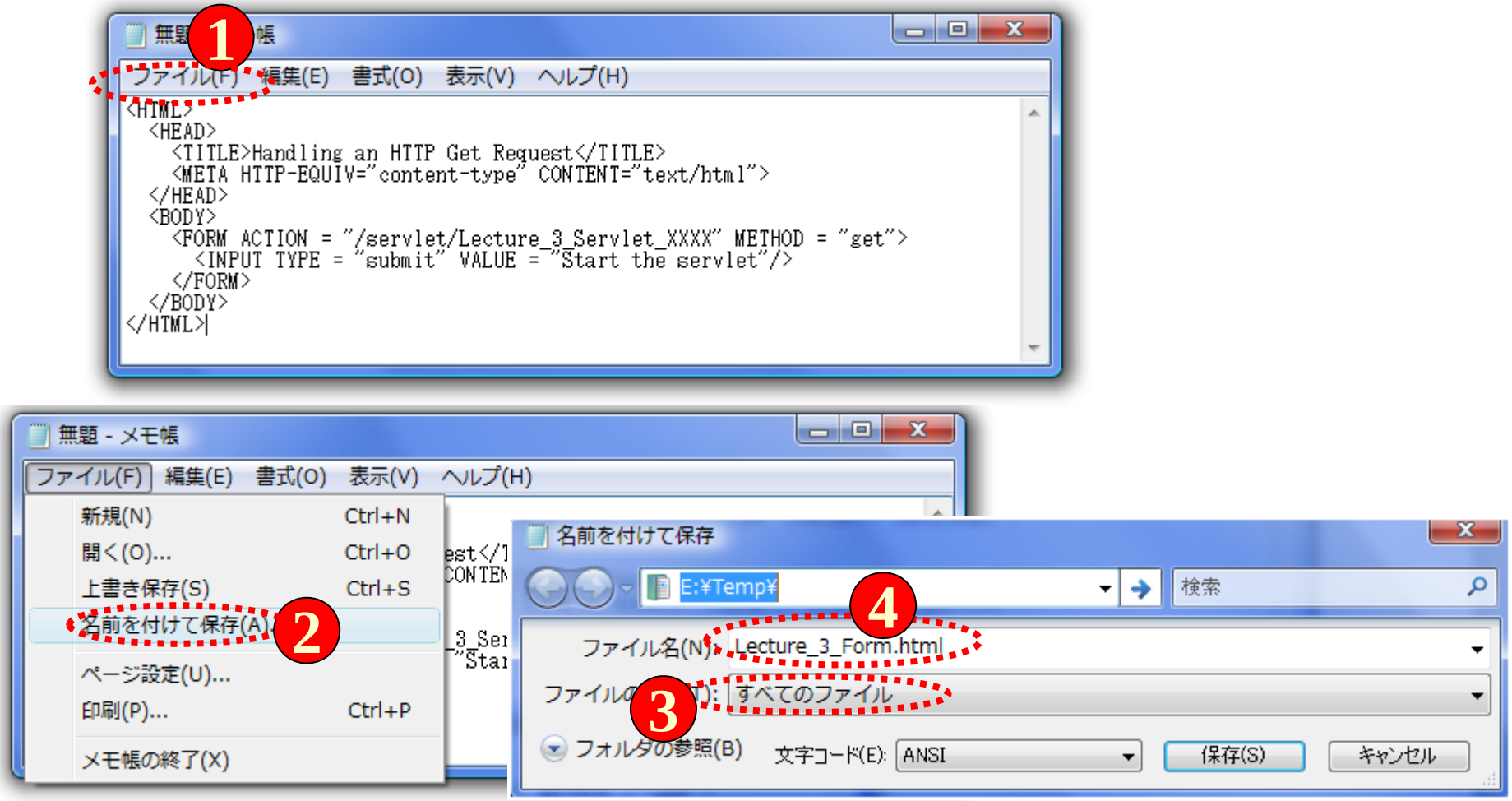
Lecture_3_Servlet_XXXX:
XXXX = 学籍番号
例: Lecture_3_Servlet_1116221001



A. Webフォームの開発

notepad

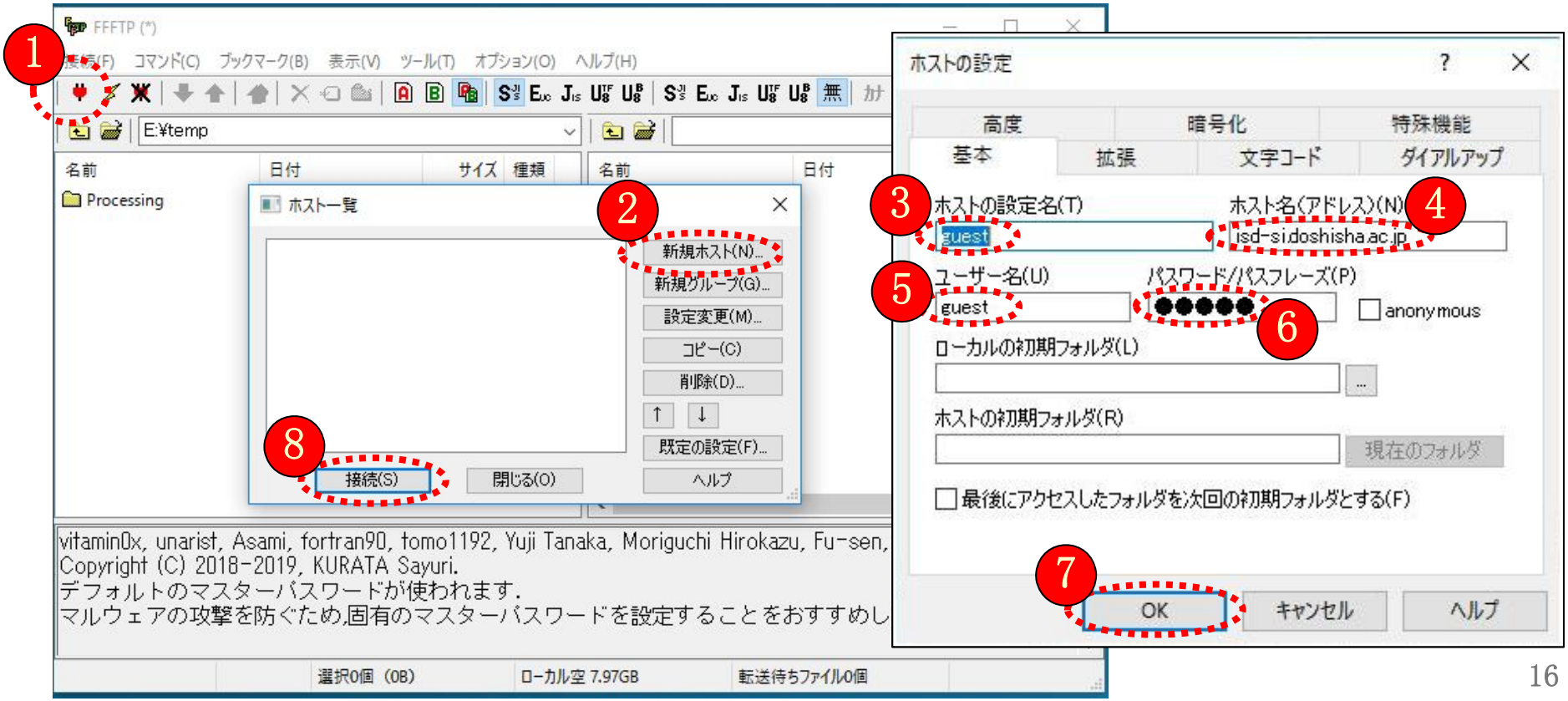
入力された内容を保存する： ファイル名はE:\Temp\Lecture_3_Form.html



B. WebフォームをWebサーバへのUpload

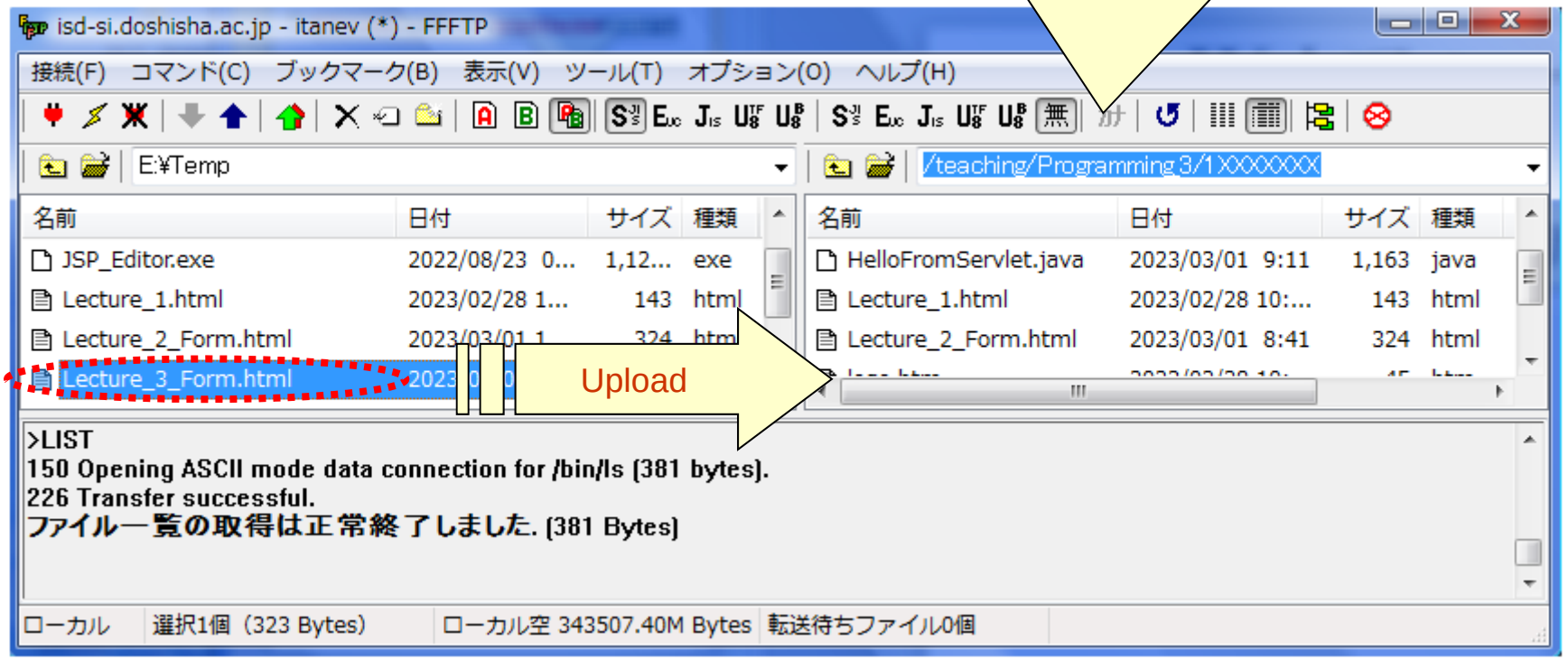
FTP Client (FFFTP)を実行し、「ホストの設定」「基本」のページでFTPサーバーのパラメターを記入。

ホスト名④は「isd-si.doshisha.ac.jp」で、⑤ユーザ名と⑥パスワードは「guest」です。

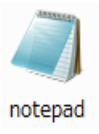


B. WebフォームをWebサーバへのUpload

自分のフォルダーを選択してから、Lecture_3_Form.htmlを**Upload**します



C. Servletの開発



1. Notepadで、下記のServletを入力する
2. Servletは E:¥Temp¥Lecture_3_Servlet_XXXX.javaに保存する

無題 - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

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

public class Lecture_3_Servlet_XXXX 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>"
                +"<META HTTP-EQUIV='Content-Type' Content='text/html;
charset=SHIFT_JIS'></META>"
                +"</HEAD>"
                +"<BODY BGCOLOR=¥"#FFFFFF¥">"
                +" <CENTER>"
                +"  <H2>Hello! Your first servlet is working</H2>"
                +"  <H3>[ local time is <font color=blue>"
                + new java.util.Date()
                +"</font> ]</H3>"
                +" </CENTER>"
                +"</BODY>"
                +"</HTML>");
        }
}
```

注意1: XXXX = 学籍番号,
例: Lecture_3_Servlet_1116221001
注意2: class名＝ファイル名 (.java)
例: class名: Lecture_3_Servlet_1116221001
ファイル名: Lecture_3_Servlet_1116221001.java

D. ServletをWebサーバへのUpload

1 FFFTPを実行する.

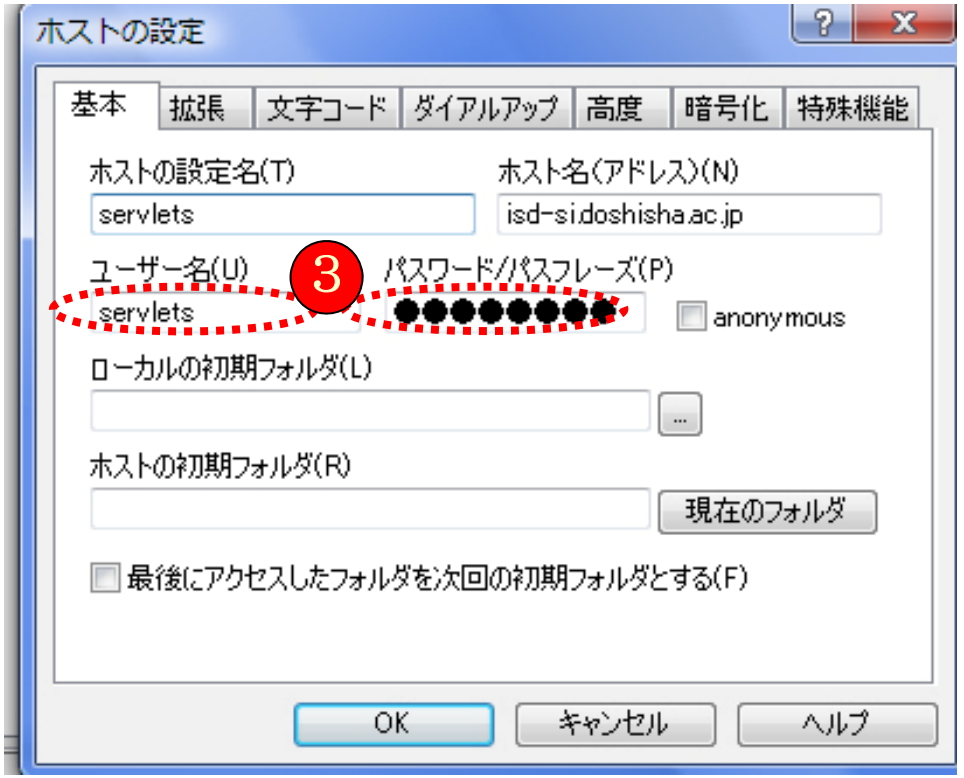
2 新規ホスト(N)... ボタンを押し、3 ホストを設定し、

接続します. ホストの設定:

ホスト名: isd-si.doshisha.ac.jp

ユーザ名: servlets

パスワード: servlets

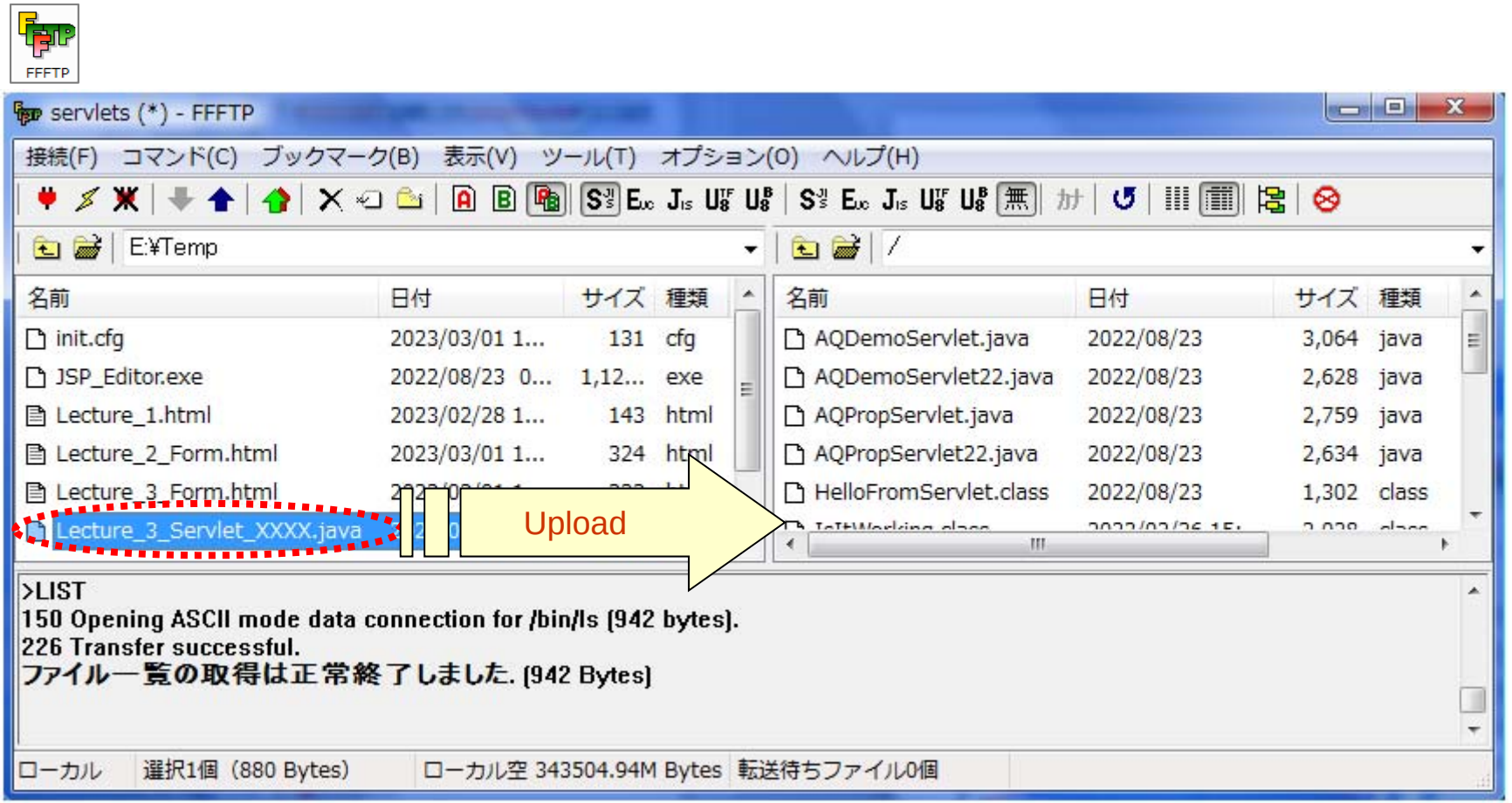


注意:

ServletのUploadの時に
ユーザ名: servlets
パスワード: servlets
です。

Web FormのUpload時に
ユーザ名: guest
パスワード: guest
でした。

D. ServletをWebサーバへのUpload



E. Servletのコンパイル

1. Web Browser: http://isd-si.doshisha.ac.jp/teaching/Programming_3/1XXXXXXX/
   (1XXXXXXX = 学籍番号)
2. Compile_Servlet.jspにAccessする:

E. Servletのコンパイル

Q: リモートコンパイルの実現ほうほうは？

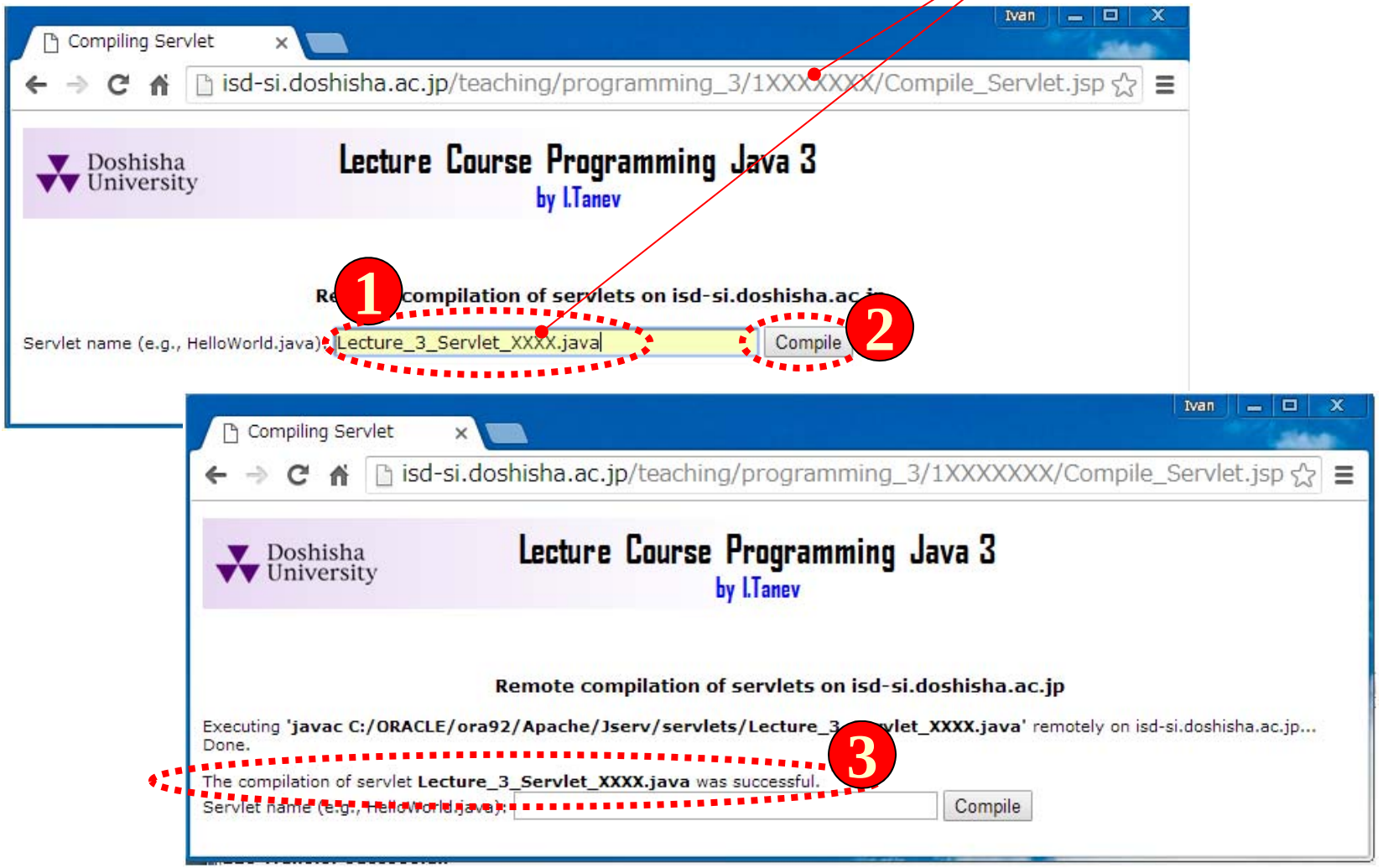
A: Compile_Servlet.jspの内容（一部）:

```
<% String
  SName = request.getParameter("SName_"),
  InfoStr="",
  cmd[]=new String[2];
  cmd[0]="javac";
  cmd[1]="C:/ORACLE/ora92/Apache/Jserv/servlets/"+SName;

if ((SName!=null) && (SName.length()>0 ))  {
try  {
    Runtime rt = Runtime.getRuntime();
    out.println("Executing <B>'" + cmd[0] + " " + cmd[1]
                +"'</B> remotely on isd-si.doshisha.ac.jp...");
Process proc = rt.exec(cmd);
…
%>
<FORM METHOD="POST" ACTION="Compile_Servlet.jsp">
   Servlet name (e.g., HelloWorld.java):
  <INPUT STYLE="font-size:9.0pt;font-family:Verdana; width:300px"
             TYPE="text" NAME="SName_" VALUE = "">
  <INPUT TYPE = submit VALUE = "Compile">
</FORM>
```

E. Servletのコンパイル

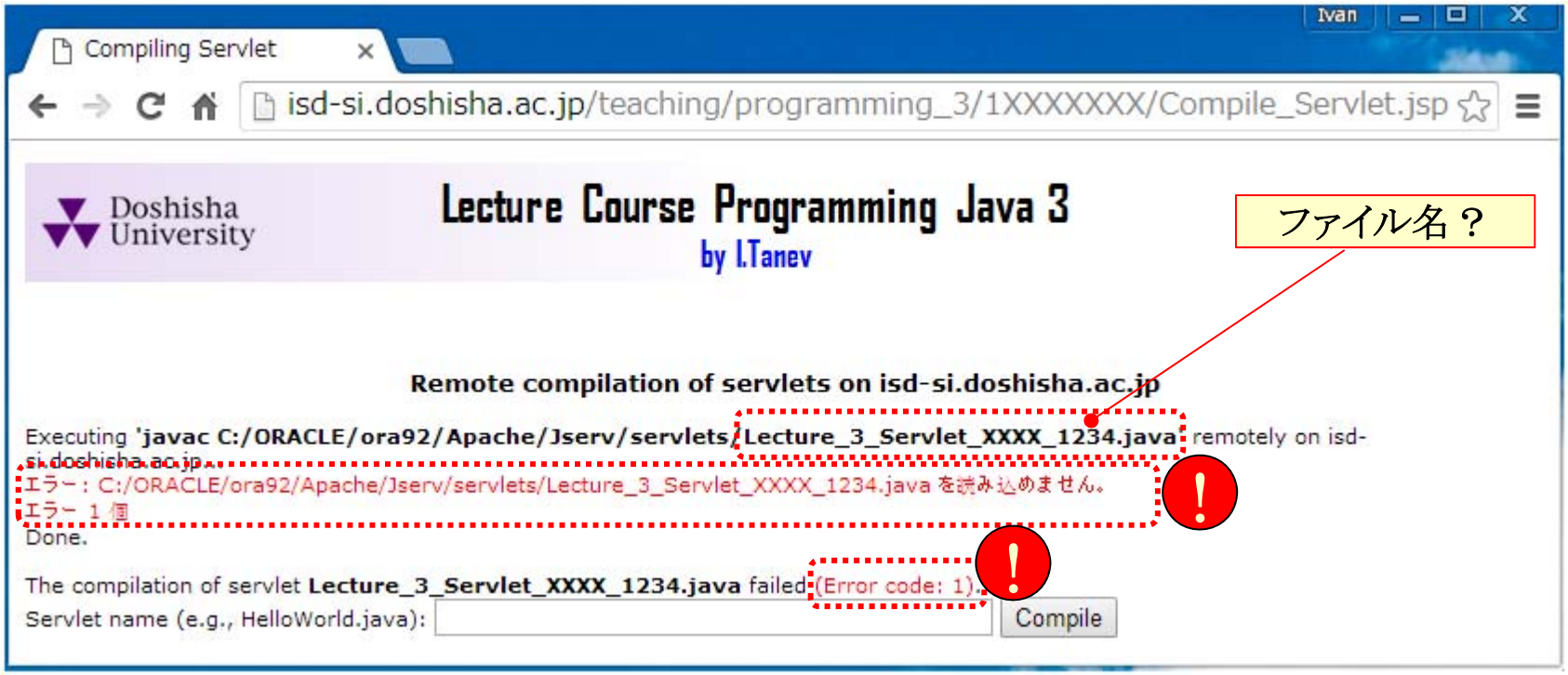
1. Servletのネームの入力
2. Compile ボタンの押す
3. コンパイル結果(エラーなし？)の確認

注意:学籍番号



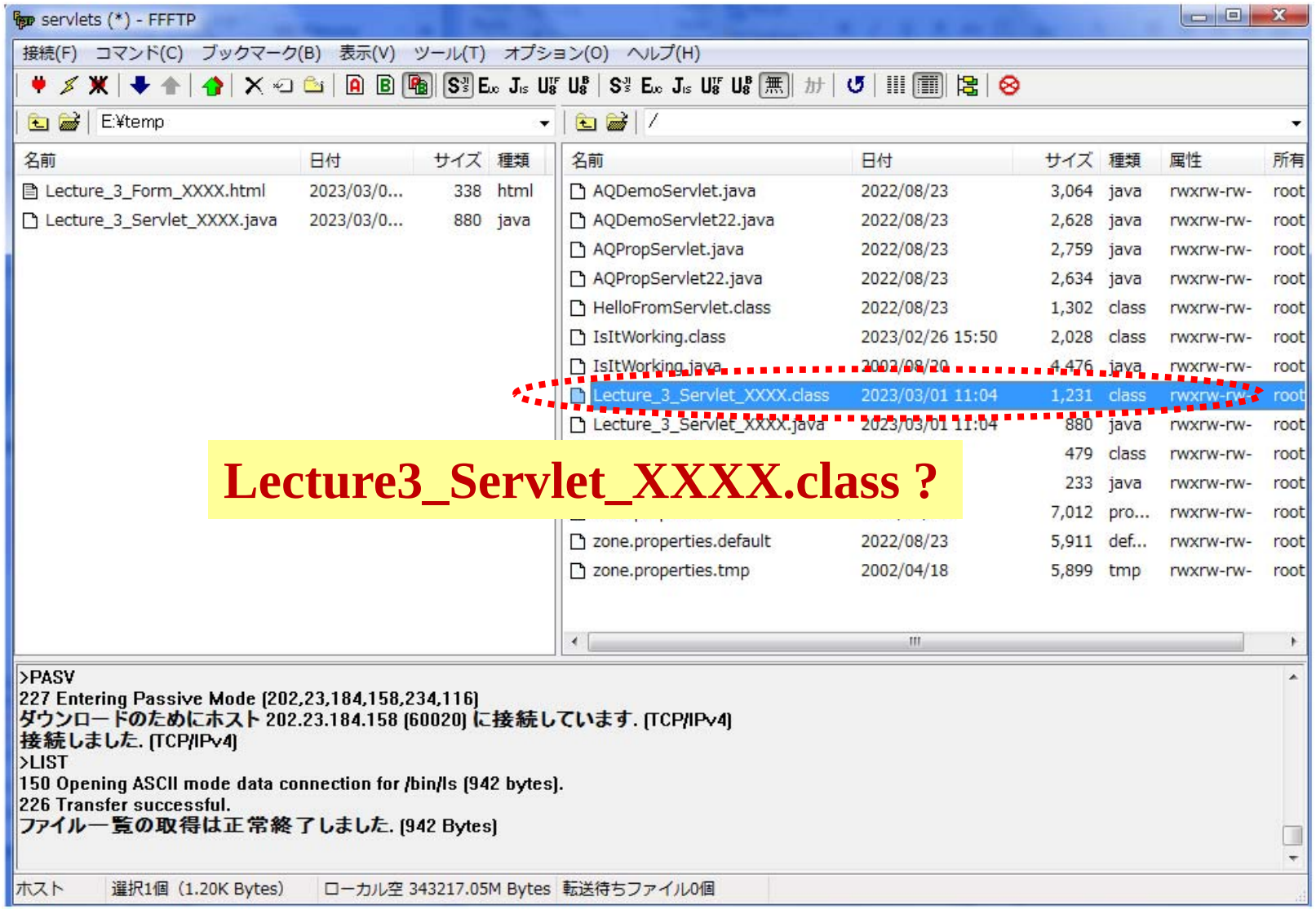
E. Servletのコンパイル

or コンパイル結果: エラーの例(エラー・メッセージは赤で表示されています)



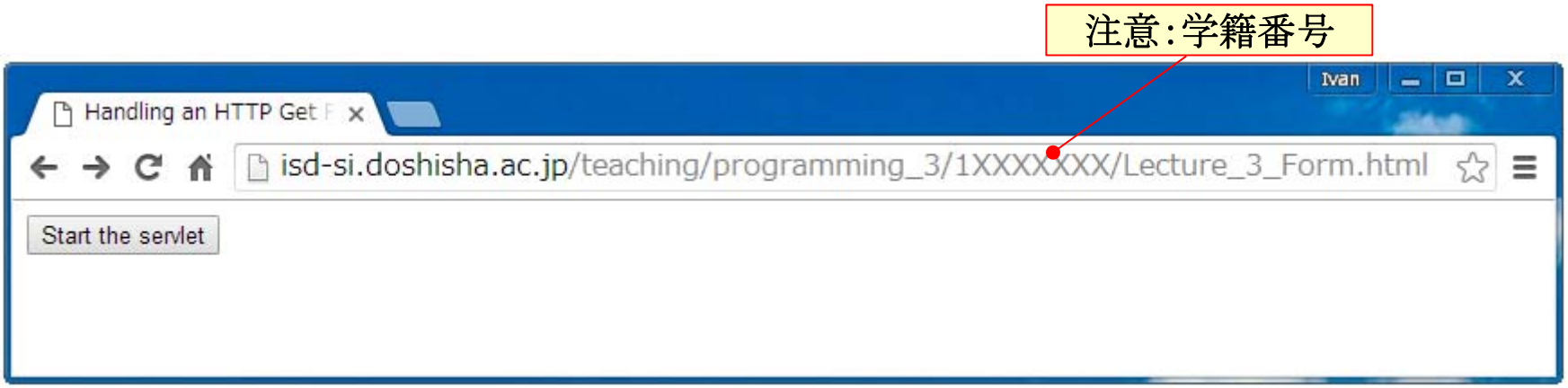
E. Servletのコンパイル

Servletはコンパイルが出来ましたか？
FFFTP ClientでLecture_3_Servlet_XXXX.classファイルの確認

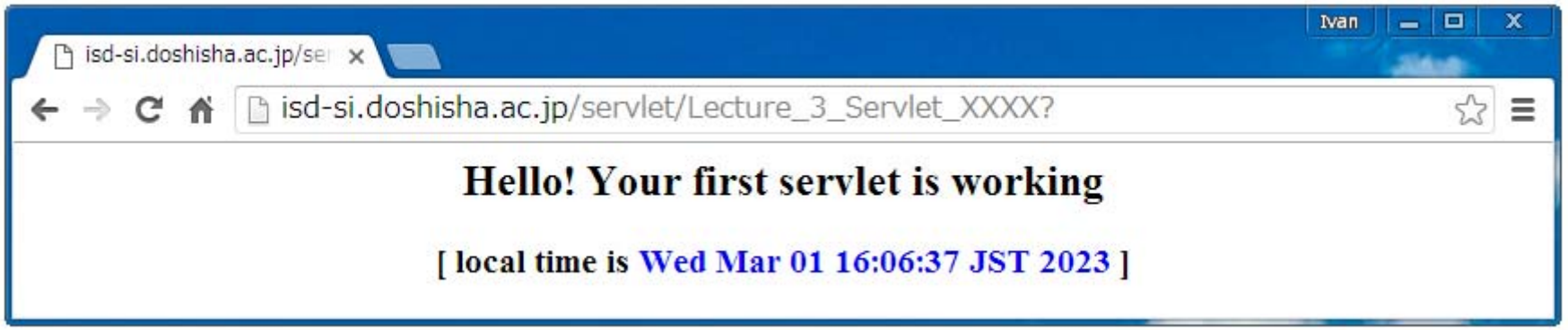


F. WebフォームのAccess

Web Browser: http://isd-si.doshisha.ac.jp/teaching/Programming_3/**1XXXXXXX**/Lecture_3_Form.html



Lecture_3_Form1.htmlの画面:サーブレットLecture_3_Servlet_XXXX呼び出す前



サーブレットLecture3_Servlet_XXXX呼び出す後

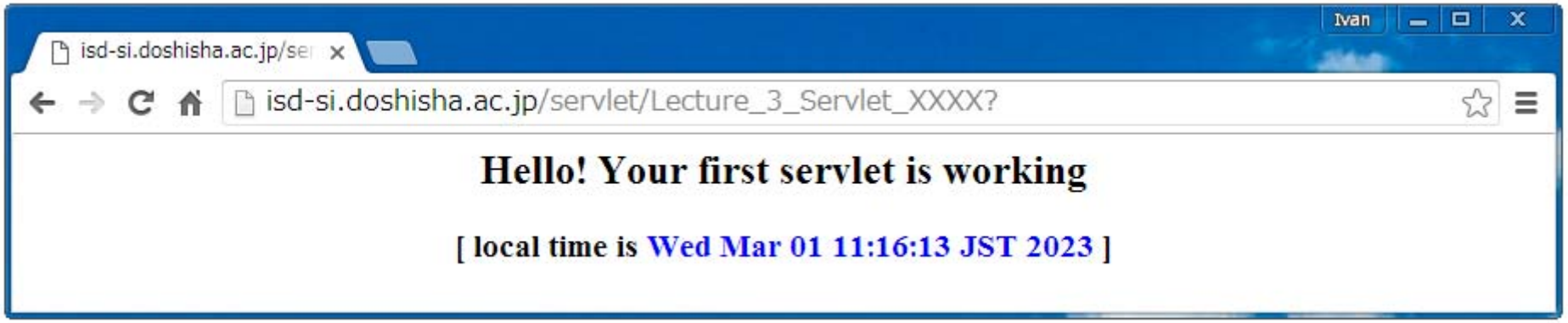
課題

Lecture_3_Form.htmlの開発

Lecture_3_Servlet_XXXX.javaの開発

Lecture_3_Servlet_XXXX.classのコンパイル

Lecture_3_Form.htmlのAccess結果:



締め切り:5月7日(木曜日)、23:59

本日の授業(第3回)の課題は

評価されています

評価されている課題のリスト
(各回:8ポイント、合計40ポイント)

- 第3回,
- 第4回,
- 第5回,
- 第6回,
- 第14回

The End