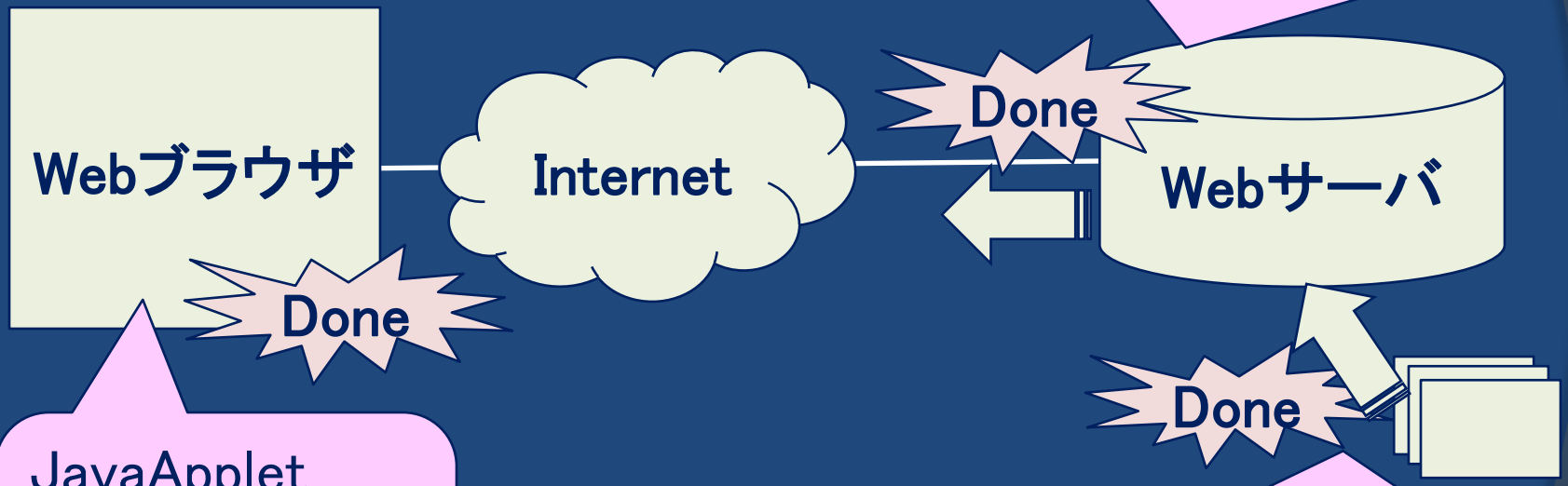
Webプログラミング２

7. JSPとServletによるWebプログラミング概要

（復習）Webアプリケーションの実現方式

★：授業で扱う範囲

SSI（Server Side Include）→ C-Shellなど
JSP（Java Server Pages）★
PHP　など



JavaApplet
JavaScript★
Active-X
Flash　など

CGI(Common Gateway Interface)
→ *Perl*、C言語、C-Shell　など
JavaServlet（★）

（1）Servlet/JSP概要

・サーバサイドプログラミングの一手法。

CGI、JavaServlet、JSPでできることはほぼ同じ。

JavaScriptとはできることが違う。

→ServletとJSPは似ているが別モノ：

どちらもJava言語で記述するが動作方法が違う。

→JavaServletは、本格的なアプリケーション開発に向いている：

厳密な記述が必要

コンパイルするので現場でのなりゆき修正ができない。

CGIに比べプロセスを何度もforkしないで良い分高速。

→ JSPはJavaServletと比べ、スクリプトベースである：

JSPはあいまいな記述が許容される、コンパイル不要、お手軽。

・記述スタイル。

JSPはHTMLの中にJavaプログラムを記述するイメージ。

JavaServletはJavaプログラムの中にHTMLを記述するイメージ。

・動作環境としてJavaServlet/JSP対応のWebサーバ上で実現される。

CGIを使える通常のWebサーバではない。

（2）Java概要

言語仕様（教科書参照⇒略）

参考サイト：
@IT
http://www.atmarkit.co.jp/fjava/index/index_jspservlet.html
Sun
http://sdc.sun.co.jp/java/javase/reference/index.html

1. main メソッド

```
public static void
main(String[ ] args)
{
      ・・・
}
```

2. 予約語

abstract
assert
boolean
break
byte
case
catch
char
class
const
contiune
default
do
double
else
enum
extends
final
finally
float
for
if
goto
implements
import
instanceof
int
interface
long
native
new
package
private
protected
public
return
short
static
strictfp
super
switch
synchronized
this
throw
throws
transient
try
void
volatile
while

3. class 表記方法

クラス修飾子(※1) class クラス名
{
  クラス定義
}
※1: public, final, abstract, 省略可

4. メソッド表記方法

メソッド修飾子(※2) 戻り値 メソッド名 引数
{
  メソッド定義
  return 戻り値;
}
※2: public, protected, private, final, abstract, static, synchronized, 省略可

5. コンストラクタ表記方法

コンストラクタ修飾子(※3) クラス名 引数
{
    コンストラクタ定義
}
※3: public, protected, private, 省略可

6. フィールド定義方法

フィールド修飾子(※4) データ型 フィールド名=初期値;
※4: public, protected, private, final, static

7.クラスの利用方法

newを使う

```
classname c =  new classname( arg );
```

8. メソッドの呼び出し / フィールドの呼び出し

```
c = classname( arg ).message( );
```

9. データ型

- 整数リテラル
  - byte
  - short
  - int
  - long
- 浮動小数点リテラル
  - float
  - double
- 文字リテラル
  - char
  - String
- 論理リテラル
  - boolean
- nullリテラル

10. 変数

- クラス変数/インスタンス変数
  - static　宣言
- ローカル変数
  - ループ内のみなど

11.演算子

```
□ . () ++ --
++ -- + - ~ !
(キャスト) new
* / %
+ -
<< >> >>>
< <= > >= instanceof
== !=
&
^
|
&&
||
?:
= += -= *= /= %= <<= >>= >>>=
$= |= ^=
```

12. 構文

- 条件分岐

```
if ( )
{
  ・・・;
}
```

```
if ( )
{
  ・・・;
}
else
{
  ・・・;
}
```

```
if ( )
{
  ・・・;
}
else if( )
{
  ・・・;
}
else
{
  ・・・;
}
```

```
switch( )
{
  case a:
   ・・・;
   break;
  case b:
    ・・・;
   break;
  default:
  ・・・;
}
```

・ループ

```
while( )
{
   ・・・;
}

do
{
   ・・・;
}
while( );

for( ; ; )
{
   ・・・;
}
```

```
for (配列:変数名)
{
   ・・・;
}
```

・ループ制御

```
{
  ..;
  break;
}
```

(ループを抜ける)

```
{
  ..;
  continue;
}
```

(先頭に戻る)

13. コンパイルと実行

・プログラム

```
class Sample
{
    main・・・

}
```

を Sample.java でセーブ

・コマンドラインでコンパイル

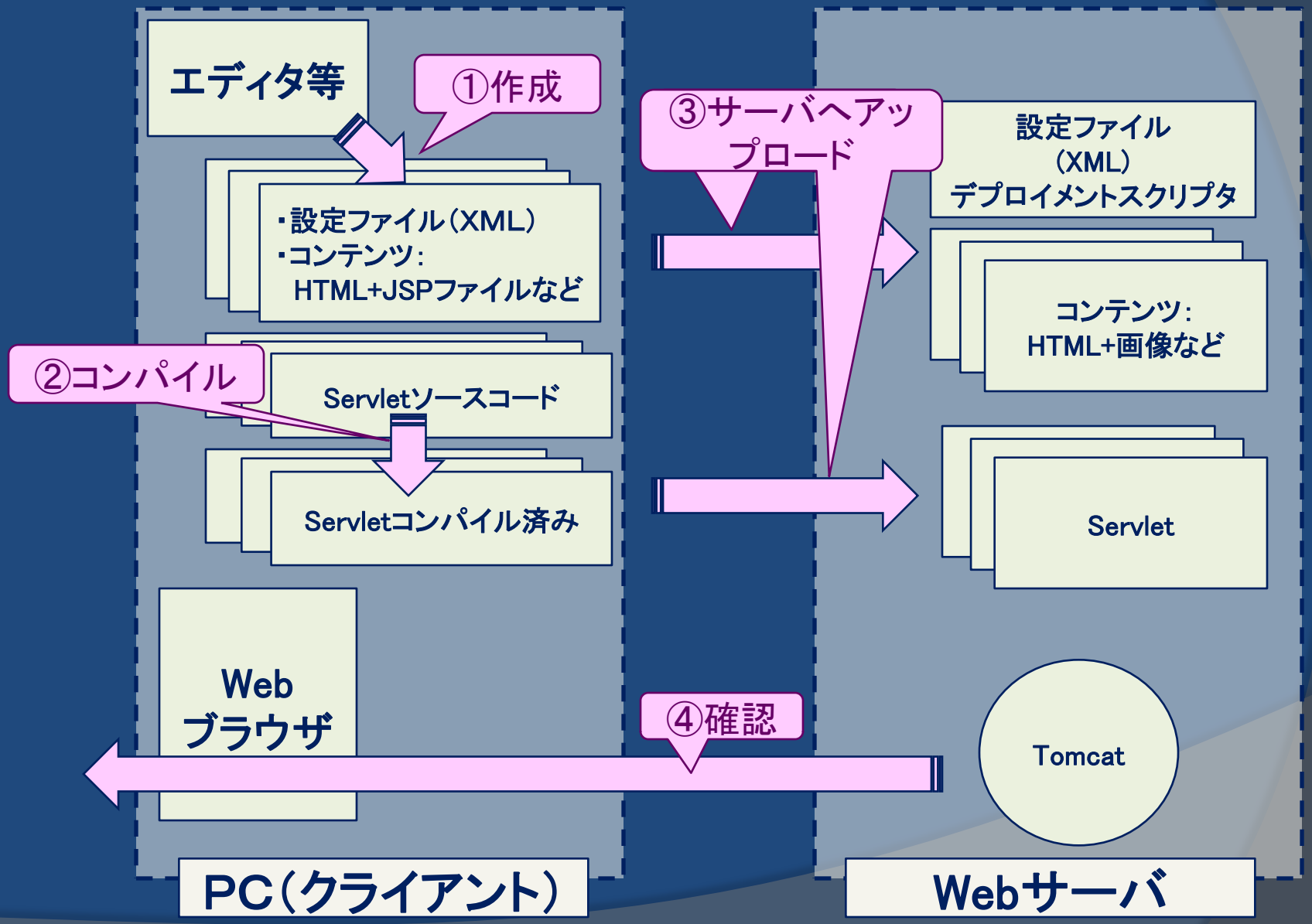
prompt% javac Sample.java

⇒ Sample.class が出力

・コマンドラインで実行

prompt% java Sample

（3）Servlet動作概要



（4）動作環境構築

・Java SE のインストール
　　http://www.oracle.com/technetwork/java/javase/downloads/
・（Webサーバ）Tomcatのインストール
　　http://tomcat.apache.org/
・環境変数の設定(システムのプロパティ):
　JAVA_HOME
　PATH
　 CLASSPATH
　 CATALINA_HOME

例

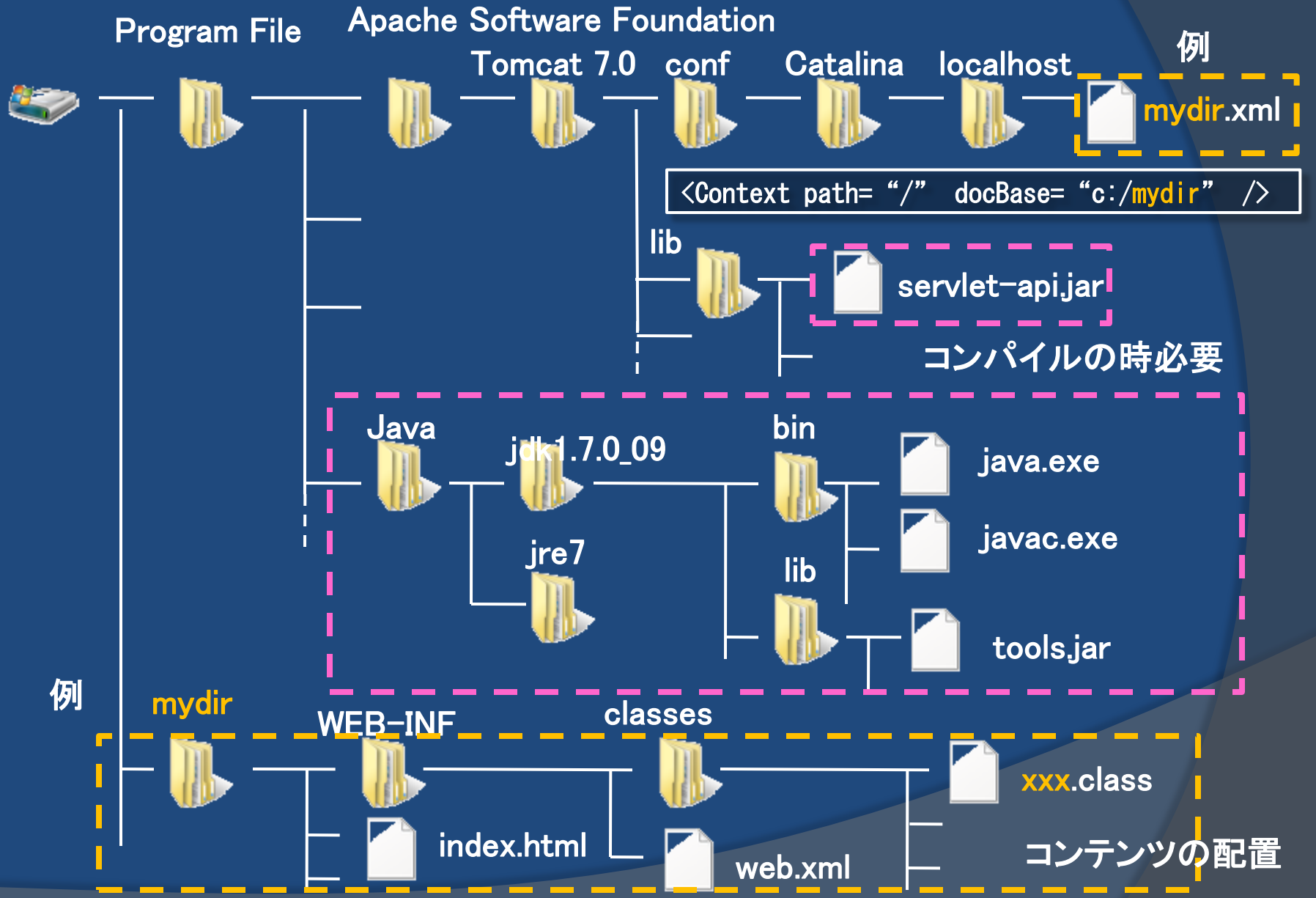
JAVA_HOME	C:¥Program Files¥Java¥jdk1.7.0_09
PATH	%JAVA_HOME%¥bin
CLASSPATH	%JAVA_HOME%¥lib¥tools.jar;%CATALINA_HOME¥lib¥servlet-api.jar
CATALINA_HOME	C:¥apache-tomcat-7.0.32

最初だけ

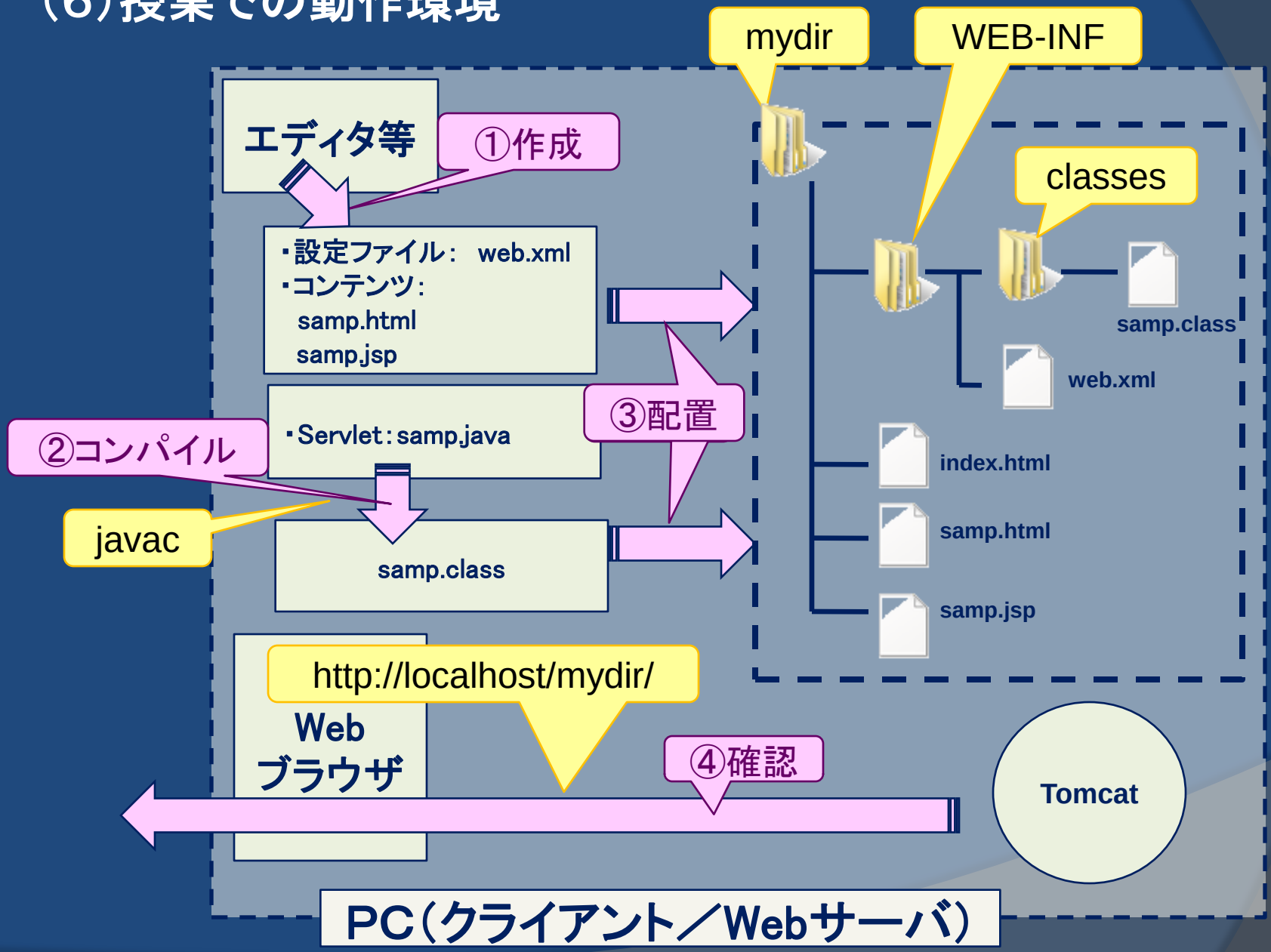
・Tomcatの起動
・設定ファイルでWebコンテンツのディレクトリを設定
・コーディング　⇒　コンパイル:　javac　⇒クラスファイルの配置
・設定ファイル（デプロイメントスクリプタ）への登録
・ブラウザからの動作確認（Tomcatの再起動を伴う場合あり）

毎回
適宜

(5)一般的なディレクトリ構造



(6)授業での動作環境


mydir
WEB-INF
classes
エディタ等
①作成
・設定ファイル： web.xml
・コンテンツ：
samp.html
samp.jsp
samp.class
web.xml
③配置
②コンパイル
・Servlet：samp.java
index.html
javac
samp.html
samp.class
samp.jsp
http://localhost/mydir/
Web
ブラウザ
④確認
Tomcat
PC（クライアント／Webサーバ）

(7)事前確認

●mydirの下に　学籍番号.html　のファイルを作成
(ファイルの中身はHTMLで記述)

●ブラウザで　http://localhost:8080/mydir/学籍番号.html
　にアクセスし確認

※見れない場合は、設定ファイル
C:¥Tomcat 7.0¥conf¥Catalina¥localhost¥mydir.xml
　を確認

```
<Context path="/mydir"
 docBase="C:¥Users¥infommXXX¥Documents¥mydir"
 reloadable="false"/>
```

(8)動作確認:Servletプログラミング方法

●とりあえずやってみる(Servlet)

① javaプログラムを作成

sample.java

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

public class sample extends HttpServlet
{
    public void doGet(HttpServletRequest request,
HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        out.print("<HTML><TITLE>sample</TITLE><BODY>");
        out.print("Hello World!");
        out.print("</BODY></HTML>");

    }
}
```

④ web.xmlを作成(追記)

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>
    <servlet>
        <servlet-name>sample</servlet-name>
        <servlet-class>sample</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>sample</servlet-name>
        <url-pattern>/sample</url-pattern>
    </servlet-mapping>

</web-app>
```

② コンパイルする

javac sample.java

sample.class

③配置する

⑤ ブラウザから呼び出す:

http://localhost:8080/mydir/sample

（9）留意事項：Tomcatの再起動

●Servletを修正してみる

・javaプログラムを修正

・コンパイル

・classファイルを配置

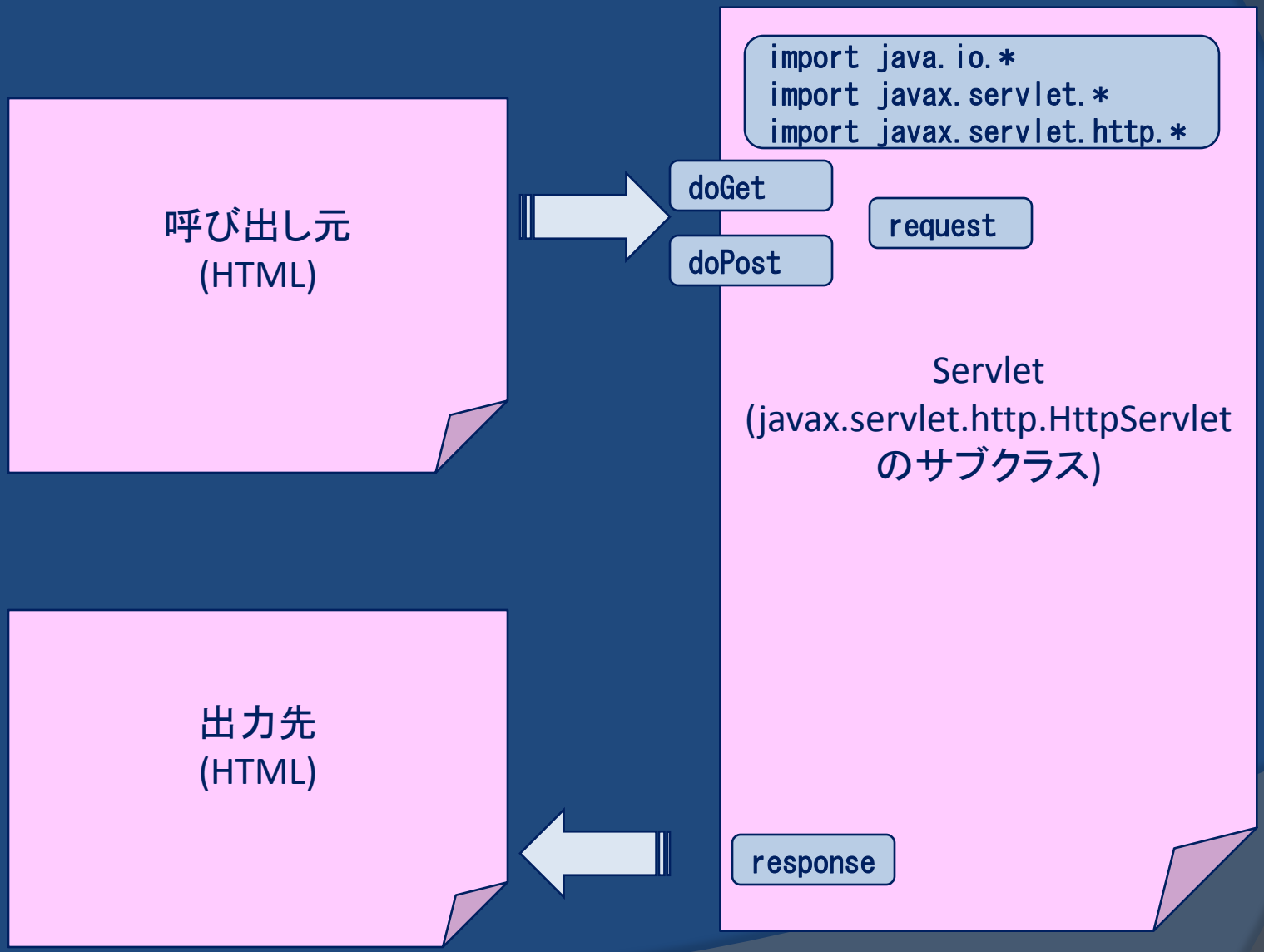
・ブラウザから確認⇒変更されていないことを確認

●Tomcatを再起動して動作を確認

方法1：設定⇒コントロールパネル ⇒管理ツール
⇒サービス⇒Tomcatの起動／再起動

方法2：プログラム⇒Tomcat⇒Tomcat Manager⇒該当部分を再ロード
授業では　ユーザ名：admin
パスワード：admin

（10）入力がある場合のServletプログラミングの基本構成

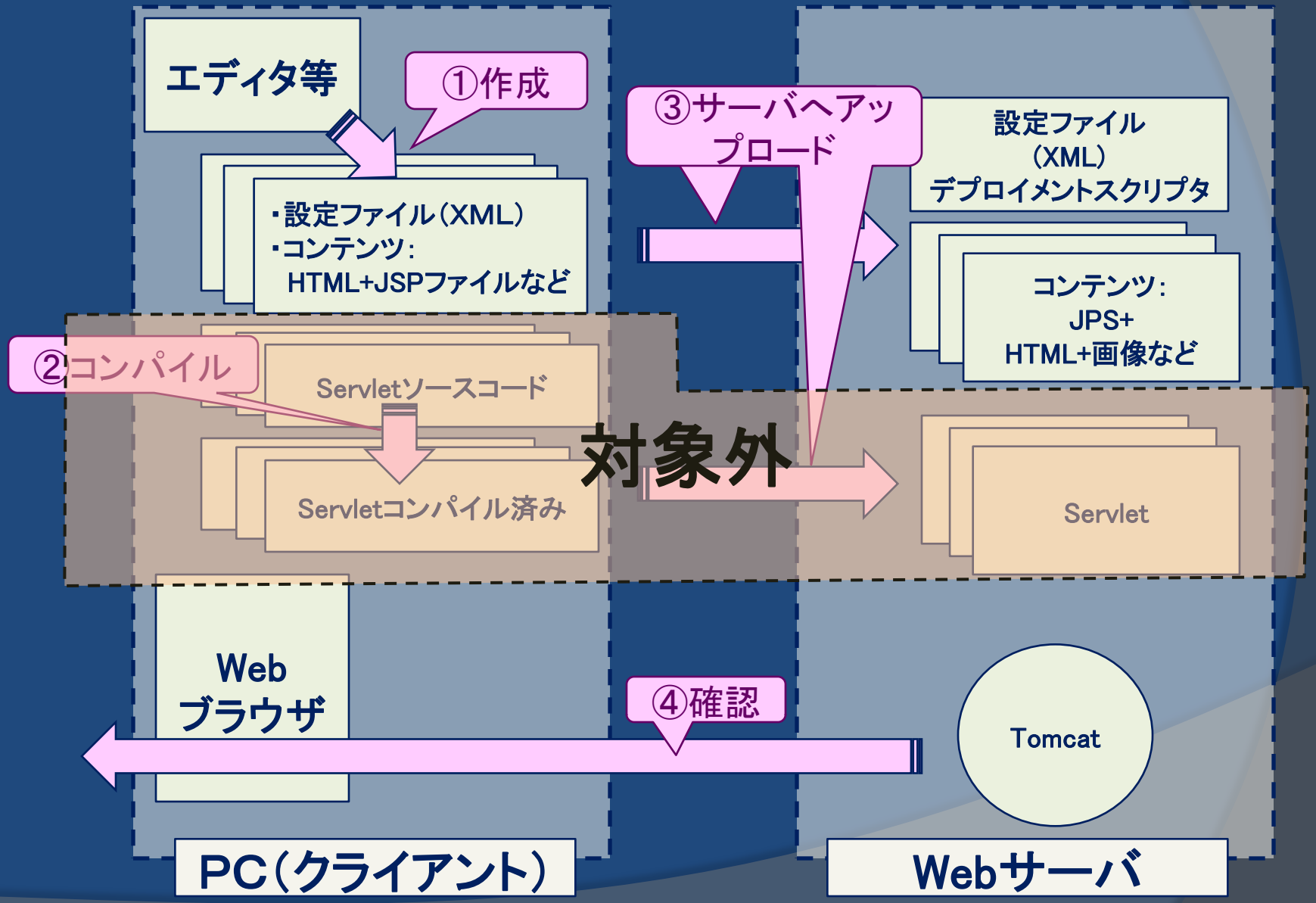


（11）練習1：HTMLファイルから入力されたデータの受取

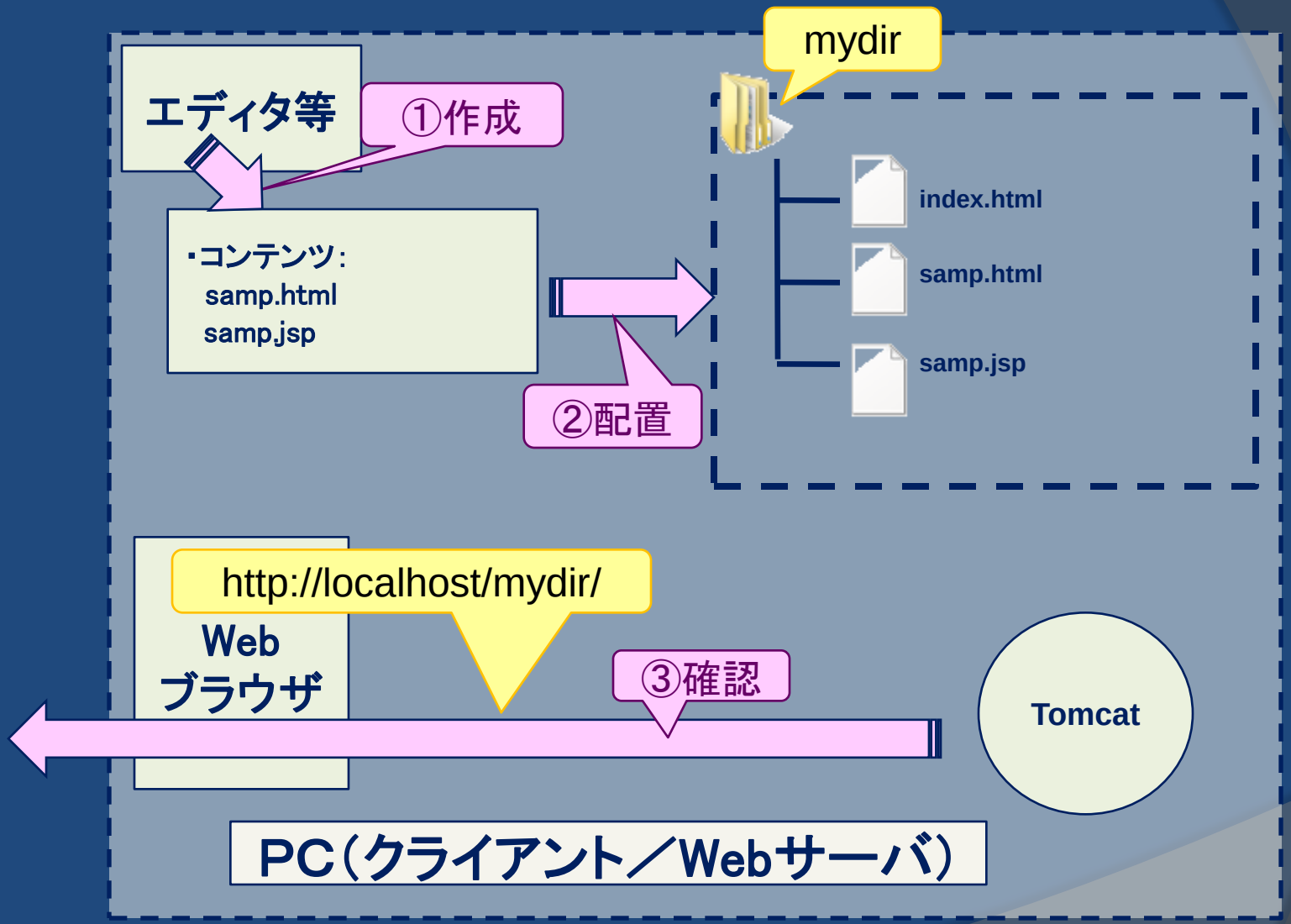
```
<HTML>
<TITLE>samp</TITLE>
<BODY>
<HR>
<FORM METHOD="POST"
ACTION="http://localhost/mydir/sample">
入力してください。<INPUT TYPE="text" NAME="sono1">
<BR>
入力してください。<INPUT TYPE="text" NAME="sono2">
<BR>
<INPUT TYPE="submit" VALUE="submit">
</FORM>
<HR>
</FORM>
</BODY>
</HTML>
```

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

public class sample extends HttpServlet
{
    public void doPost(HttpServletRequest request,
HttpServletResponse response)
    throws IOException, ServletException
    {
        String input;
//      request.setCharacterEncoding("Shift_JIS");

        response.setContentType("text/html; charaset=Shift_JIS");
        PrintWriter out = response.getWriter();
        out.print("<HTML><TITLE>sample</TITLE><BODY>");
        out.print("Hello World Servlet<BR>");
        input=request.getParameter("sono1");
        out.print( input );
        out.print( "<BR>" );
        input=request.getParameter("sono2");
        out.print( input );
        out.print("</BODY></HTML>");
    }
}
```

（12）JSP動作概要


エディタ等
①作成
・設定ファイル（XML）
・コンテンツ：
HTML+JSPファイルなど
③サーバへアップロード
設定ファイル
（XML）
デプロイメントスクリプタ
コンテンツ：
JPS+
HTML+画像など
②コンパイル
Servletソースコード
Servletコンパイル済み
対象外
Servlet
Web
ブラウザ
④確認
Tomcat
PC（クライアント）
Webサーバ

（13）動作環境（JSP）


mydir
エディタ等
①作成
・コンテンツ：
samp.html
samp.jsp
②配置
index.html
samp.html
samp.jsp
http://localhost/mydir/
Web
ブラウザ
③確認
Tomcat
PC（クライアント／Webサーバ）

（14）JSPの書式

●JSP 書式 ：

ディレクティブ	<%@	%>	JSPから生成されるServletを制御
宣言部	<%!	%>	定数、メンバ変数、ユーザ定義メソッドを定義
スクリプトレット	<%	%>	スクリプトの記述
式	<%=	%>	out.println( ); の短縮系
コメント	<%--	--%>	コメントアウト

例

```
<%@ page contentType="text/html;charset=Shift_JIS" %>
<HTML><TITLE>samp</TITLE><BODY>
<%-- これは練習です。 --%>
<%!
 int flag=100;
%>
<%
 out.print("Hello World!");
 out.print("<BR>");
 out.print(new java.util.Date());
 out.print( flag+100 );
%>
<BR>
<%= "Hello World"  %>
</BODY></HTML>
```

（15）事前確認 （servletの時と同じ）

●mydirの下に　学籍番号.html　のファイルを作成
（ファイルの中身はHTMLで記述）

●ブラウザで　http://localhost:8080/mydir/学籍番号.html
　にアクセスし確認

※見れない場合は、設定ファイル
C:¥Tomcat 7.0¥conf¥Catalina¥localhost¥mydir.xml
　を確認

```
<Context path="/mydir"
 docBase="C:¥Users¥infommXXX¥Documents¥mydir"
 reloadable="false"/>
```

(16)JSPプログラミング方法

●とりあえずやってみる(HTML＋JSP)

・動作確認:mydir　配下に以下を配置

samp.html

```
<HTML>
<TITLE>sample</TITLE>
<BODY>
Hello World
<BR>
</BODY>
</HTML>
```

samp.jsp

```
<%@ page contentType="text/html;charset=Shift_JIS" %>
<HTML>
<TITLE>sample</TITLE>
<BODY>
<%= "Hello World" %>
<BR>
</BODY>
</HTML>
```

・ブラウザから以下のURLで確認

http://localhost:8080/mydir/samp.html

または

http://127.0.0.1:8080/mydir/samp.html

http://localhost:8080/mydir/samp.jsp

または

http://127.0.0.1:8080/mydir/samp.jsp

（17）練習2：HTMLファイルから入力されたデータの受取

```
<HTML>
<TITLE>samp</TITLE>
<BODY>
<HR>
<FORM METHOD="POST"
ACTION="http://localhost/mydir/samp3.jsp">
入力してください。<INPUT TYPE="text"
NAME="sono1"><BR>
入力してください。<INPUT TYPE="text"
NAME="sono2"><BR>
<INPUT TYPE="submit" VALUE="submit">
</FORM>
<HR>
</FORM>
</BODY>
</HTML>
```

```
<%@ page contentType="text/html;charset=Shift_JIS" %>
<%!
String input;
%>
<HTML>
<TITLE>samp</TITLE>
<BODY>
<HR>
<%
request.setCharacterEncoding("Shift_JIS");
input=request.getParameter("sono1");
out.print( input );
%>
<BR>
<%
input=request.getParameter("sono2");
out.print( input );
%>
<HR>
</BODY>
</HTML>
```

（18）練習3：Webサーバとしての動作確認

・以下のようなJSPを使ったHTMLファイルを作成
（自分のIPアドレスを調べる）

samp1.jsp

```
<%@ page contentType="text/html;charset=Shift_JIS" %>
<HTML>
<TITLE>sample</TITLE>
<BODY>
<%
java.net.InetAddress in = java.net.InetAddress.getLocalHost();
String localAddress = in.getHostAddress();
out.println( localAddress );
%>
</BODY>
</HTML>
```

・JSPを使ったHTMLファイルを自由に記述
（他の人が見て面白い内容にする）

・ブラウザを使って他の人のWebページを参照する

http://XXX周りの誰かのIPアドレスXXX:8080/mydir/samp.jsp