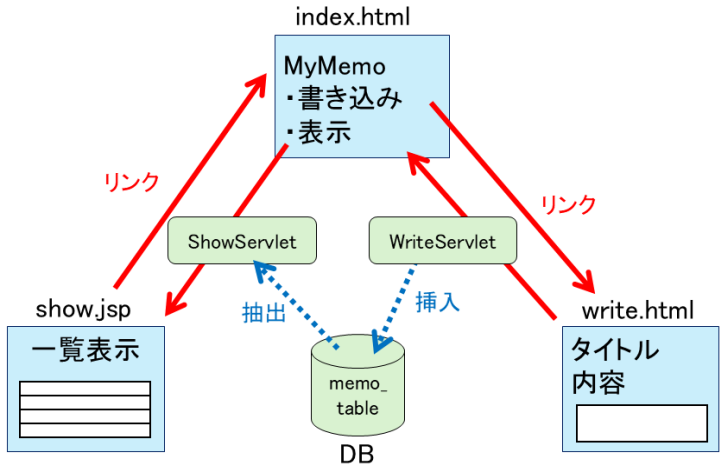
サーバーサイドJava

5. メモアプリケーション
（Servlet, JSP, DB）

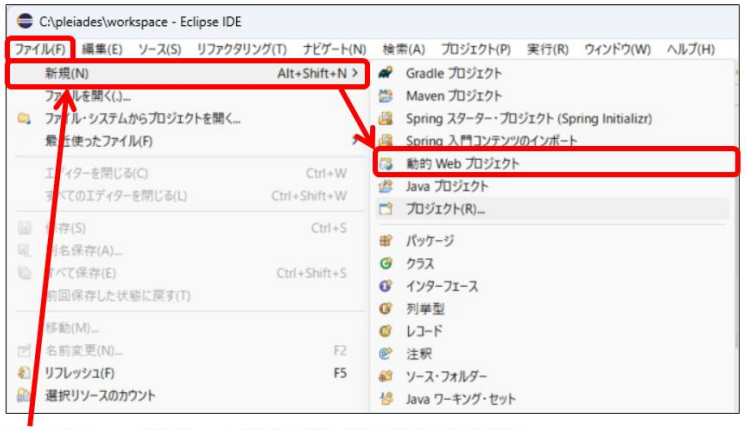
メモ・アプリケーションの作成

- Servlet、JSP、DBを連携させたメモ・アプリケーションの例を示す。
- このアプリでは次の3ページ作成する。
  - index.html トップページ
  - show.jsp　メモの一覧表示ページ
  - write.html メモの書き込みページ
- 処理を行なうために次の二つのServletを作成する。
  - ShowServlet DBにアクセスして一覧表示する
  - WriteServlet フォームデータをDBに挿入する
- メモデータはデータベースに格納する。
- アプリケーションの全体構成を次頁に示す。

メモ・アプリケーションの作成



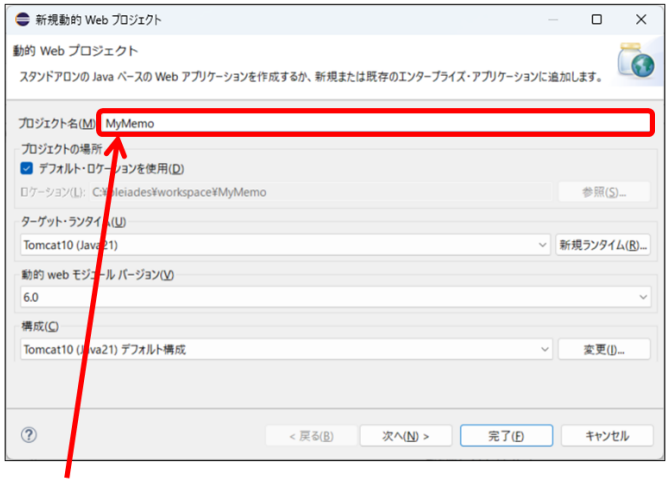
プロジェクトの作成



「ファイル → 新規 → 動的プロジェクト」を選択

注.動的Webプロジェクトが表示されないときは「プロジェクト」を選択する

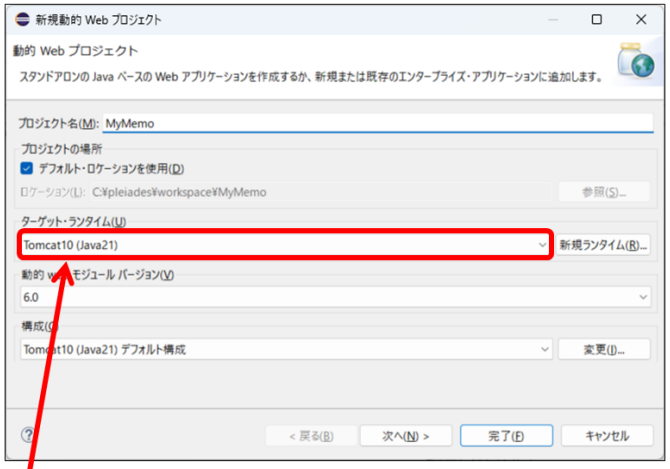
プロジェクトの作成



プロジェクト名に「MyMemo」と入力

5

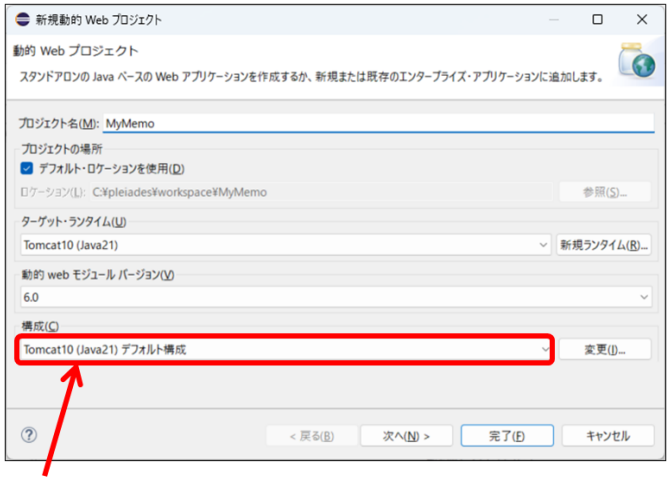
プロジェクトの作成



ターゲットランタイムで「Tomcat10(Java21)」を選択

6

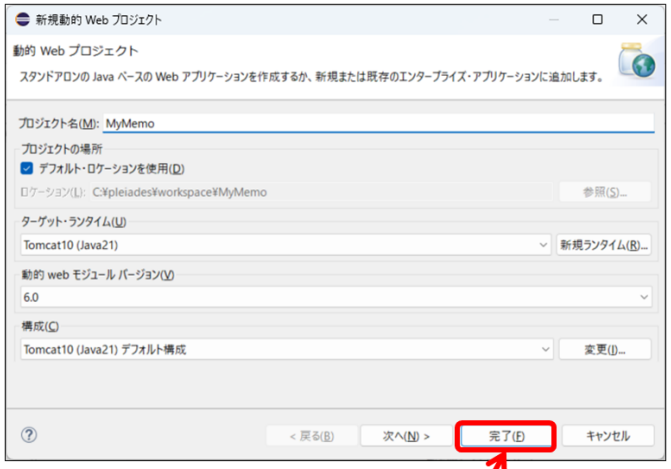
プロジェクトの作成



構成で「Tomcat10(Java21)デフォルト構成」を選択

7

プロジェクトの作成



「完了」をクリック

8

表の作成

- タイトルとメモ内容を記録する表をデータベース内にmemo_tableという名前で作成しておく。
- この表には番号、記録日時、タイトル、メモ内容を格納する4列を定義する。また、番号は主キーとする。

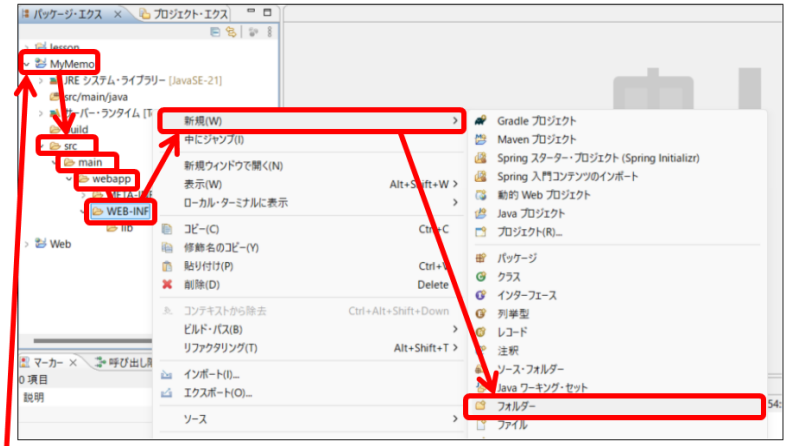
列名	id	tm	title	memo
データ型	INTEGER	TEXT	TEXT	TEXT
具体的な内容	番号	日時	タイトル	メモ内容

データベースと表の作成

- sqlite3コマンドで、プロジェクト内の「webapp/WEB-INF/data」ディレクトリに「memo.db」という名前でデータベースファイルを作成し、更に、次のコマンドでメモデータを格納するテーブルを作成しておく。
- テーブル作成のコマンドは次のようになる。

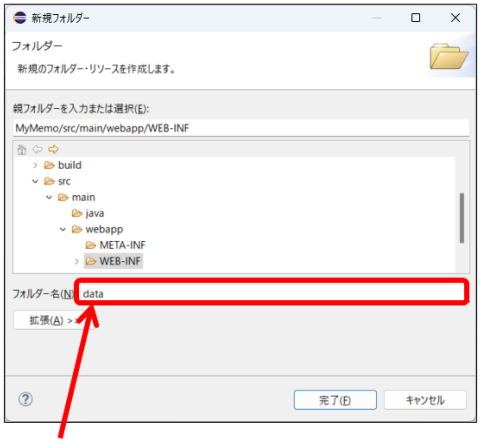
```
CREATE TABLE memo_table(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        tm TEXT,
        title TEXT,
        memo TEXT
);
```


データベースと表の作成



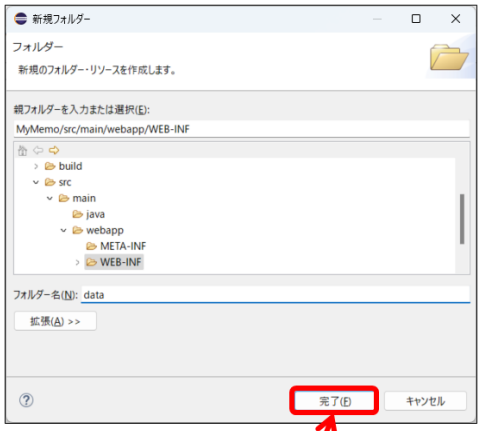
「src/main/webapp/WEB-INF」を右クリックして「新規 → フォルダー」を選択

データベースと表の作成



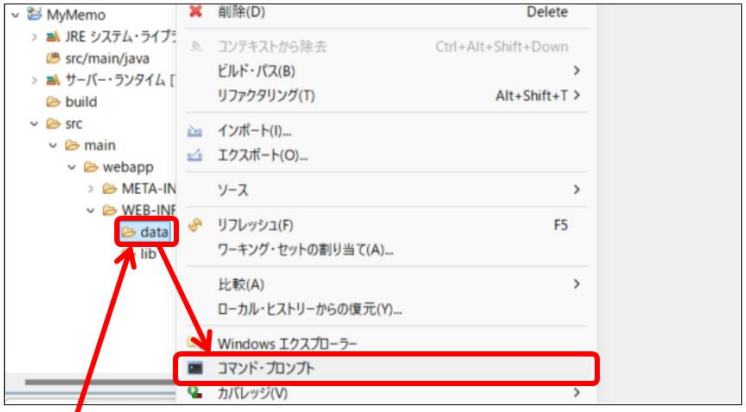
フォルダー名に「data」と入力

データベースと表の作成



「完了」をクリック

データベースと表の作成



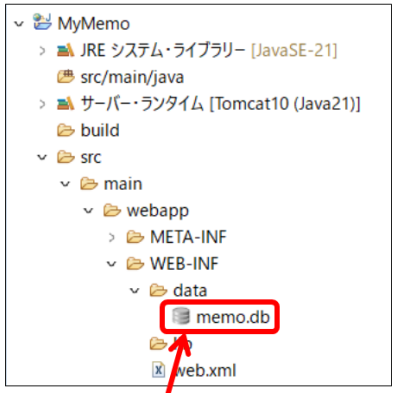
作成されたdataフォルダーを右クリックして「コマンド・プロンプト」を選択

データベースと表の作成

●コマンドプロンプト上で次を実行する。

```
…¥WEB-INF¥data> sqlite3 memo.db      ← DBに接続(新規作成)
SQLite version 3.46.0 2024-05-23 13:25:27 (UTF-16 console I/O)
Enter ".help" for usage hints.
sqlite> CREATE TABLE memo_table(     ← 表の作成
(x1...>     id INTEGER PRIMARY KEY AUTOINCREMENT,
(x1...>     tm TEXT,
(x1...>     title TEXT,
(x1...>     memo TEXT
(x1...> );
sqlite> .tables  ← 作成された表の確認
memo_table
sqlite> .quit  ← sqlite3コマンドの終了
```


データベースと表の作成

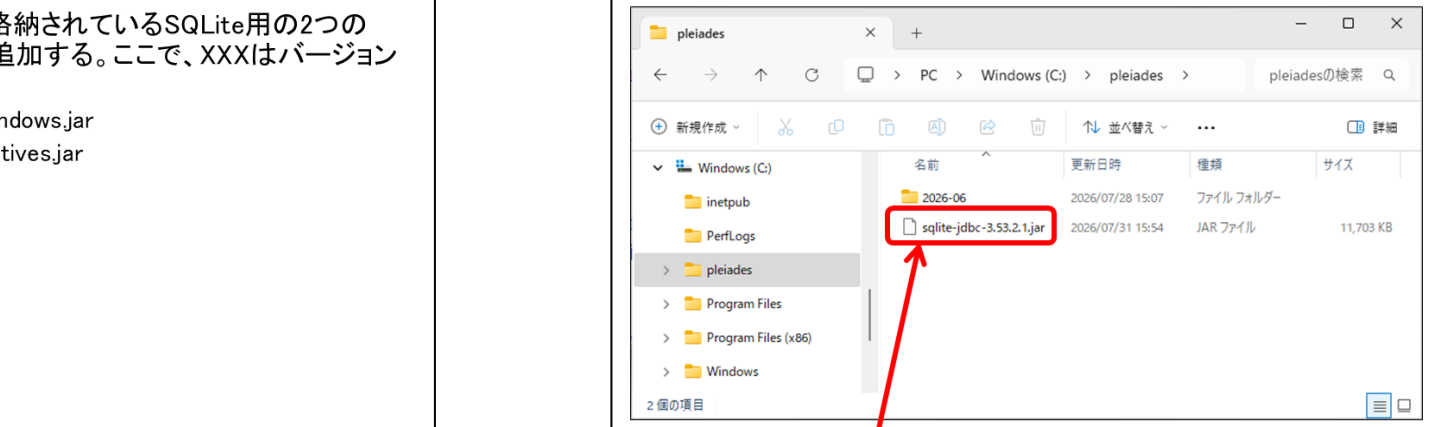


「src/main/webapp/WEB-INF/data」に「memo.db」が作成される

JARファイルの追加

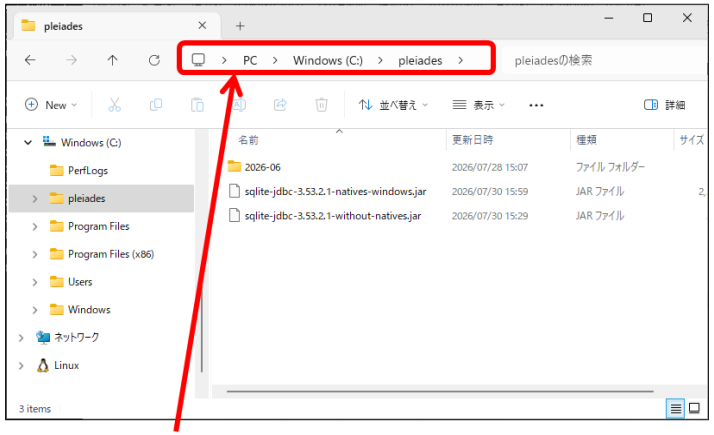
- SQLiteのJDBCドライバーが格納されているSQLite用の2つのJARファイルをプロジェクトに追加する。ここで、XXXはバージョン番号を表すものとする。
  - sqlite-jdbc-XXX-natives-windows.jar
  - sqlite-jdbc-XXX-without-natives.jar

JARファイルの追加



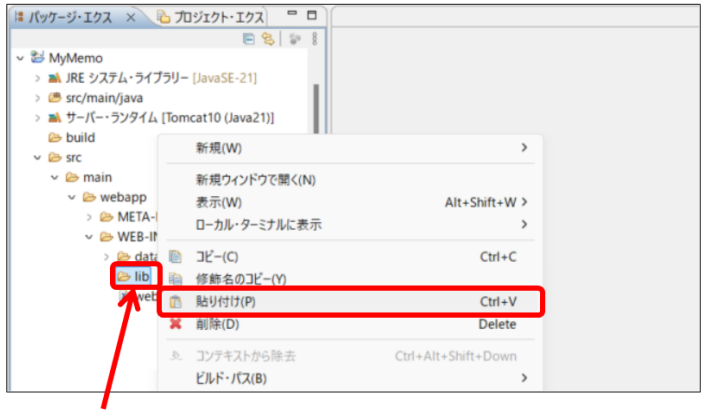
「sqlite-jdbc-XXX.jar」ファイルをコピー

JARファイルの追加



エクスプローラで「C:¥pleiades」を開く

JARファイルの追加



「src/main/webapp/WEB-INF/lib」に張り付け

JARファイルの追加

「src/main/webapp/WEB-INF/lib」に「sqlite-jdbc-XXX.jar」が配置される

index.htmlの作成

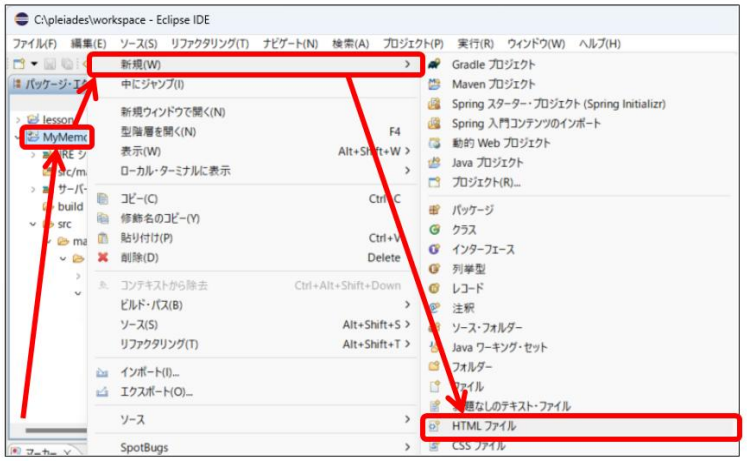
●トップページのindex.htmlでは、一覧表示ページを表示するShowServletへのリンクと、書き込みページwrite.htmlへのリンクを置いておく。

index.htmlの作成

index.html

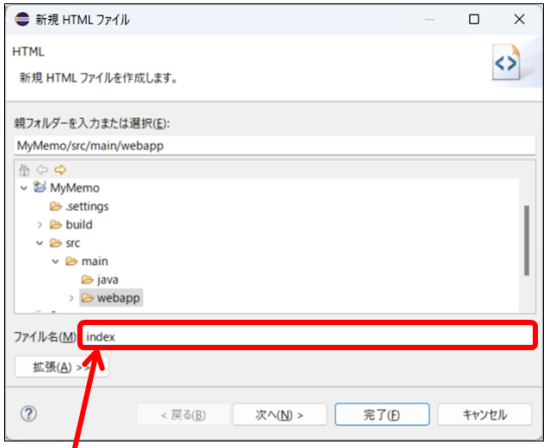
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>メモ帳</title>
</head>
<body>
    <h3>マイメモ</h3>
    <p><a href="/MyMemo/write.html">★書き込み</a></p>
    <p><a href="/MyMemo/ShowServlet">★表示</a></p>
</body>
</html>
```


index.htmlの作成



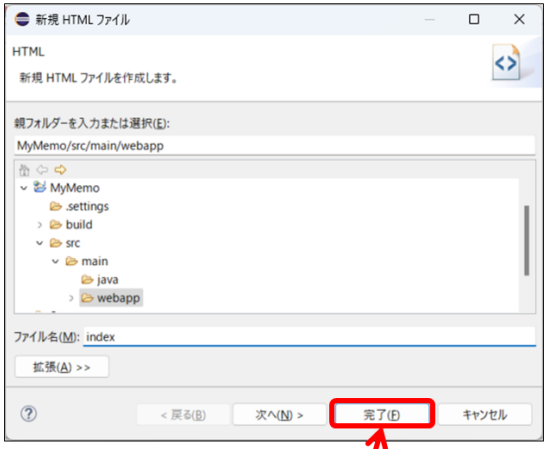
プロジェクト名MyMemoを右クリックして「新規 → HTMLファイル」を選択

index.htmlの作成



ファイル名に「index」と入力

index.htmlの作成



「完了」をクリック

index.htmlの作成

index.htmlを編集して保存

write.htmlの作成

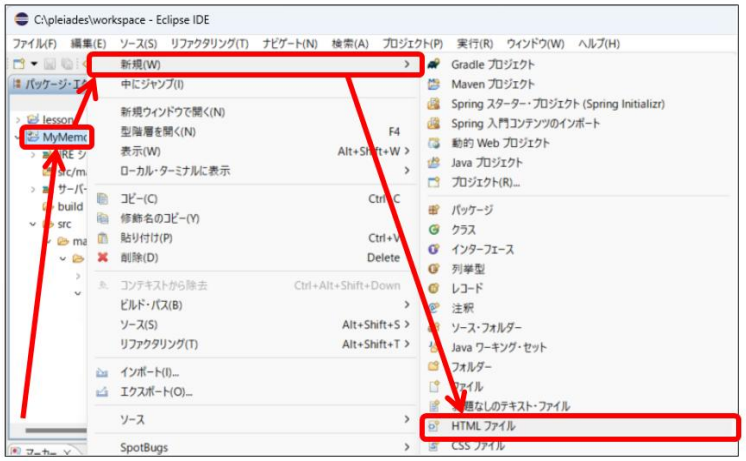
●write.htmlではメモを書き込むためのフォームを用意する。フォーム上にはタイトルとメモを書き込むコントロールを置いておく。

write.htmlの作成

write.html

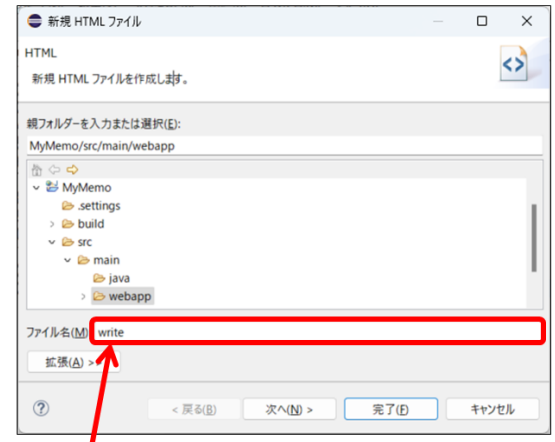
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>メモの書き込み</title>
</head>
<body>
    <form action="/MyMemo/WriteServlet" method="POST">
    <p>タイトル<br><input type="text" name="title"></p>
    <p>メ　モ<br>
      <textarea name="memo" cols="20" rows="5"></textarea></p>
    <p><input type="submit" value="記録"></p>
    </form>
</body>
</html>
```


write.htmlの作成



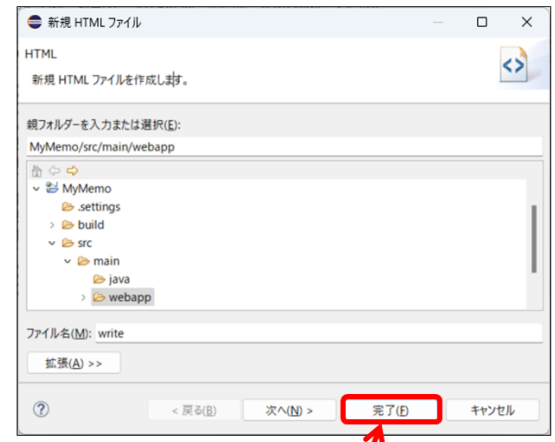
プロジェクト名MyMemoを右クリックして「新規 → HTMLファイル」を選択

write.htmlの作成



ファイル名に「write」と入力

write.htmlの作成



「完了」をクリック

write.htmlの作成

write.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>メモの書き込み</title>
</head>
<body style="font-family:sans-serif">
    <form action="/MyMemo/WriteServlet" method="POST">
    <p>タイトル<br><input type="text" name="title" size="30"></p>
    <p>メ モ<br>
      <textarea name="memo" cols="30" rows="5"></textarea></p>
    <p><input type="submit" value="記録"></p>
    </form>
</body>
</html>
```

write.htmlを編集して保存

Line.javaの作成

- データベースに格納された一行分のデータを入れるためのクラス(エンティティークラスともいう)Lineを、次頁の内容で作成する。
- このクラスは番号(主キー)、記録日時、タイトル、メモ内容を格納するフィールドと、その値を取得するメソッド(ゲッターメソッド)を用意する。また、オブジェクト生成時に値をセットできるようにコンストラクタを用意しておく。

Line.javaの作成

Line.java(1/2)

```
package com.example.app;

public class Line{
   private int id;       //番号
   private String tm;    //記録日時
   private String title; //タイトル
   private String memo;  //メモ内容

   public Line(int id, String tm, String title, String memo){
      this.id = id;
      this.tm = tm;
      this.title = title;
      this.memo = memo;
   }
```


Line.javaの作成

Line.java(2/2)

```
   public int getId() { return id; }
   public void setId(int id) { this.id = id; }

   public String getTm() { return tm; }
   public void setTm(String tm) { this.tm = tm; }

   public String getTitle() { return title; }
   public void setTitle(String title) { this.title = title; }

   public String getMemo() { return memo; }
   public void setMemo(String memo) { this.memo = memo; }
}
```

Line.javaの作成

プロジェクト名MyMemoを右クリックして「新規 → クラス」を選択

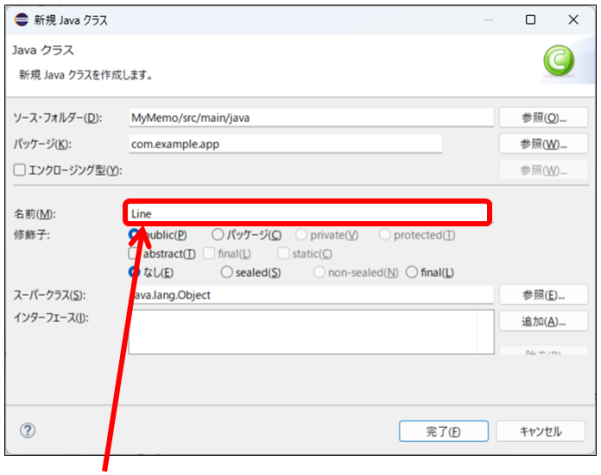
37

Line.javaの作成

パッケージに「com.example.app」と入力

38

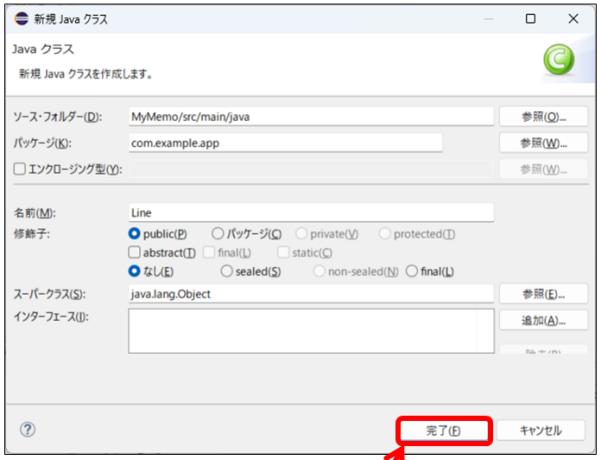
Line.javaの作成



クラス名に「Line」と入力

39

Line.javaの作成



「完了」をクリック

40

Line.javaの作成手順①

*Line.java

```
1  package com.example.app;
2
3  public class Line {
4      private int id;        //番号
5      private String tm;     //記録日時
6      private String title;  //タイトル
7      private String memo;   //メモ内容
8
9      public Line(int id, String tm, String title, String memo) {
10         this.id = id;
11         this.tm = tm;
12         this.title = title;
13         this.memo = memo;
14     }
15
16
17 }
18
```

プロパティ（フィールド）とコンストラクタを入力

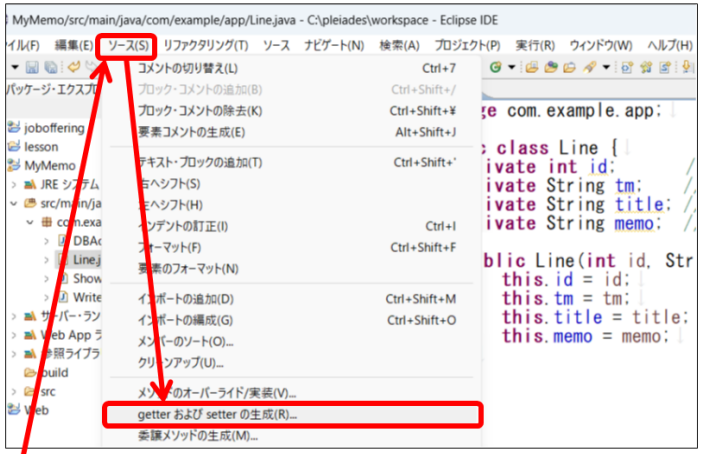
Line.javaの作成手順②

*Line.java

```
1  package com.example.app;
2
3  public class Line {
4      private int id;        //番号
5      private String tm;     //記録日時
6      private String title;  //タイトル
7      private String memo;   //メモ内容
8
9      public Line(int id, String tm, String title, String memo) {
10         this.id = id;
11         this.tm = tm;
12         this.title = title;
13         this.memo = memo;
14     }
15
16
17 }
18
```

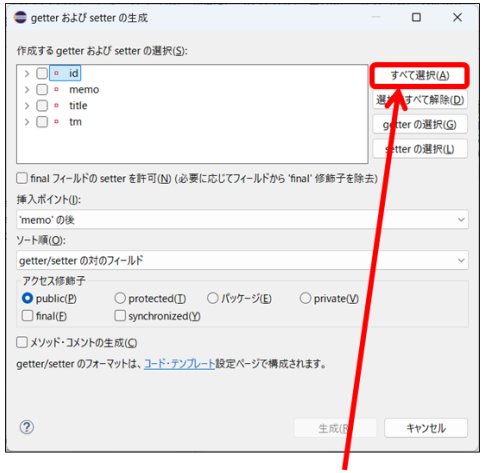
コンストラクタの下（クラスの閉じ括弧の上）をクリック

Line.javaの作成手順③



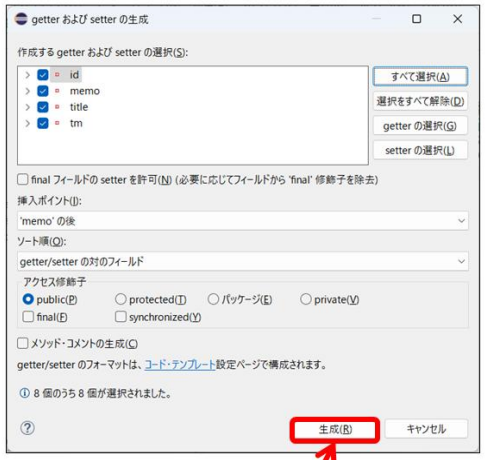
「ソース → getterおよびsetterの生成」を選択

Line.javaの作成手順④



「すべて選択」をクリック

Line.javaの作成手順⑤



「生成」をクリック

Line.javaの作成手順⑥

Line.java

```
1  package com.example.app;
2
3  public class Line {
4      private int id;      //番号
5      private String tm;   //記録日時
6      private String title; //タイトル
7      private String memo;  //メモ内容
8
9      public Line(int id, String tm, String title, String memo) {
10         this.id = id;
11         this.tm = tm;
12         this.title = title;
13         this.memo = memo;
14     }
15
16     public int getId() {
17         return id;
18     }
19
20     public void setId(int id) {
21         this.id = id;
22     }
```

getXXX()メソッドとsetXXX()メソッドが追加される
→ [Ctrl] + [s] で保存しておく

DBAccessクラス

- データベースにアクセスするための専用のクラス(DAOとも呼ばれる)を用意する。
- このクラスではwrite.htmlのフォームから送信されたメモ内容を記録するメソッド insert() と、メモ内容の一覧を取得するメソッド select() を定義している。

DBAccess.javaの作成

DBAccess. java (1/3)

```
package com.example.app;
import java.sql.*;
import java.util.*;
import jakarta.servlet.http.*;
public class DBAccess {
   //DB接続情報
   protected String connectionString = "";
   //コンストラクタ：memo.dbのパスを取得してフィールドにセット
   public DBAccess(HttpServletRequest request) {
      String dbPath = request.getServletContext()
                              .getRealPath("/WEB-INF/data/memo.db");
      connectionString = "jdbc:sqlite:" + dbPath;
   }
   //DB接続メソッド
   private Connection getConnection()
                          throws SQLException, ClassNotFoundException {
      Class.forName("org.sqlite.JDBC");
      return (DriverManager.getConnection(connectionString));
   }
```

DBAccess.javaの作成

DBAccess. java (2/3)　Dateがjava.utilとjava.sqlで被るので明示的に指定

```
   //DBへの書き込み
   public void insert(String title, String memo) {
      String tm = String.format("%tF %<tT", new java.util.Date());
      String sql =
             "INSERT INTO memo_table(tm,title,memo) VALUES(?,?,?)";
      try( Connection con = getConnection();
           PreparedStatement ps = con.prepareStatement(sql) ){
              ps.setString(1, tm);
              ps.setString(2, title);
              ps.setString(3, memo);
              ps.executeUpdate();
      }
      catch(SQLException | ClassNotFoundException e){
         e.printStackTrace();
      }
   }
```


DBAccess.javaの作成

DBAccess. java (3/3)

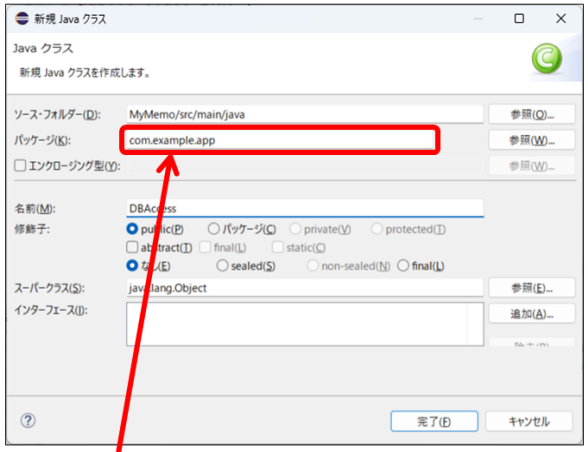
```
   //DBからデータの抽出
   public ArrayList<Line> select() {
      String sql = "SELECT * FROM memo_table";
      ArrayList<Line> list = new ArrayList<>();
      try( Connection con = getConnection();
           PreparedStatement ps = con.prepareStatement(sql);
           ResultSet rs = ps.executeQuery() ){
              while (rs.next()) {
                 Line line = new Line(rs.getInt(1), rs.getString(2),
                                      rs.getString(3), rs.getString(4));
                 list.add(line);
              }
      }
      catch (SQLException | ClassNotFoundException e) {
         e.printStackTrace();
      }
      return (list);
   }
}
```

注. 戻り値のArrayListは通常はListを使うが、文法の内容を簡単にするためArrayListにしておく！

DBAccess.javaの作成

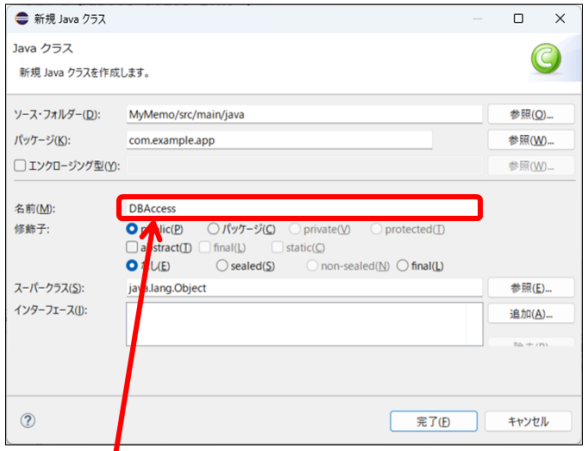
プロジェクト名MyMemoを右クリックして「新規 → クラス」を選択

DBAccess.javaの作成



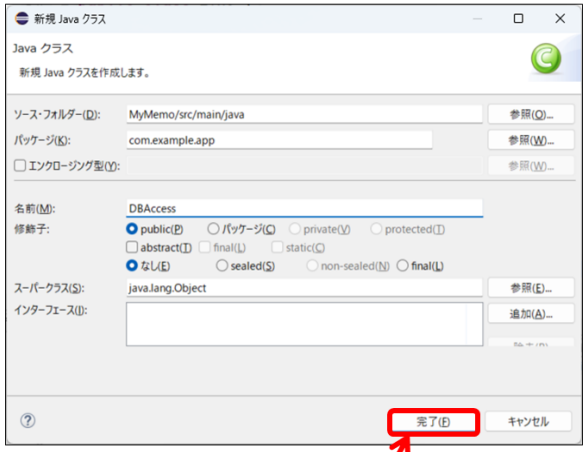
パッケージに「com.example.app」と入力

DBAccess.javaの作成



クラス名に「DBAccess」と入力

DBAccess.javaの作成



「完了」をクリック

DBAccess.javaの作成

```java
package com.example.app;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;

import jakarta.servlet.http.HttpServletRequest;

public class DBAccess {
    //DB接続情報
    protected String connectionString = "";

    //コンストラクター：memo.dbの実際のパスを取得してフィールドにセット
    public DBAccess(HttpServletRequest request) {
        String dbPath = request.getServletContext().getRealPath("/WEB-INF/data/memo.db");
        connectionString = "jdbc:sqlite:" + dbPath;
    }

    //DB接続
    private Connection getConnection() throws SQLException, ClassNotFoundException {
        Class.forName("org.sqlite.JDBC");
        return (DriverManager.getConnection(connectionString));
    }
```

DBAccess.javaを編集して保存 → ソースは別添1参照

WriteServletの作成

- サーブレットWriteServletでは、write.htmlからコールされて、次のことを行う。
  - getParameter()で入力値の取り込み
  - DBAccess.insert()でDBへ入力値の挿入
  - トップページへの遷移指定

WriteServletの作成

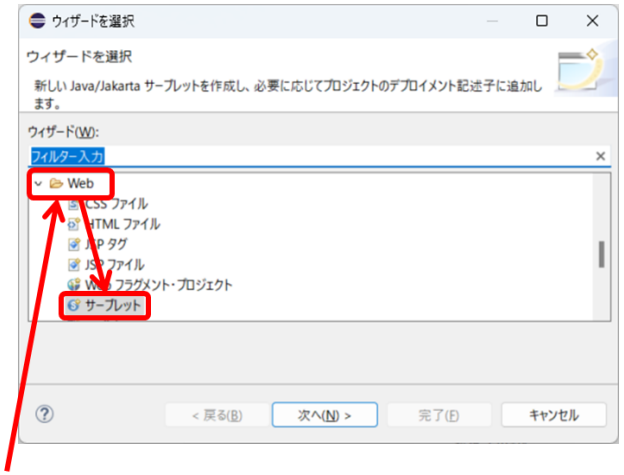
●WriteServletのdoPost() の処理は次のようにする。

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                       throws ServletException, IOException{
    //パラメータ取得
    String title = request.getParameter("title");
    String memo = request.getParameter("memo");
    //DBに挿入
    DBAccess dba = new DBAccess(request);
    dba.insert(title, memo);
    //トップページの呼び出し
    request.getRequestDispatcher("/index.html")
           .forward(request, response);
}
```


WriteServletの作成

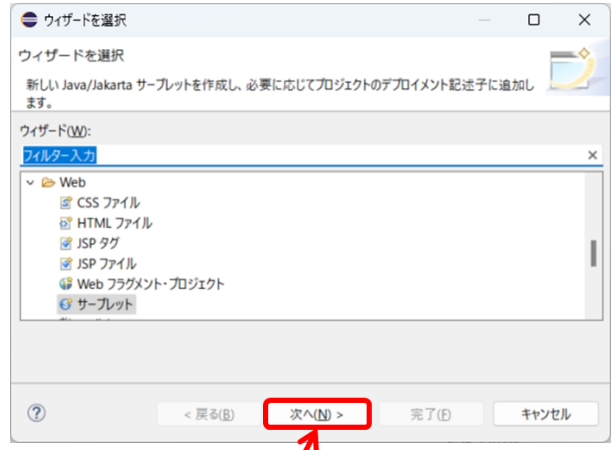
プロジェクト名MyMemoを右クリックして「新規 → その他」を選択

WriteServletの作成



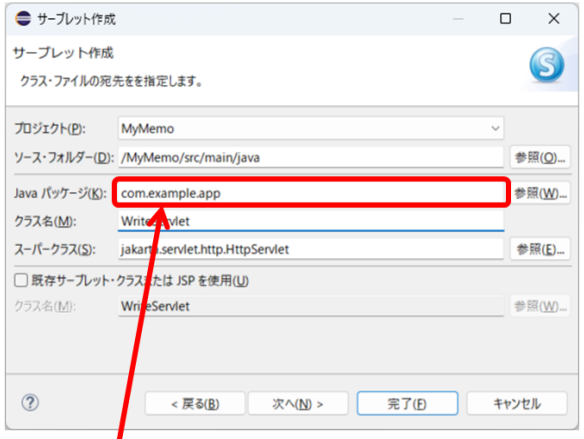
Webを展開して「サーブレット」を選択

WriteServletの作成



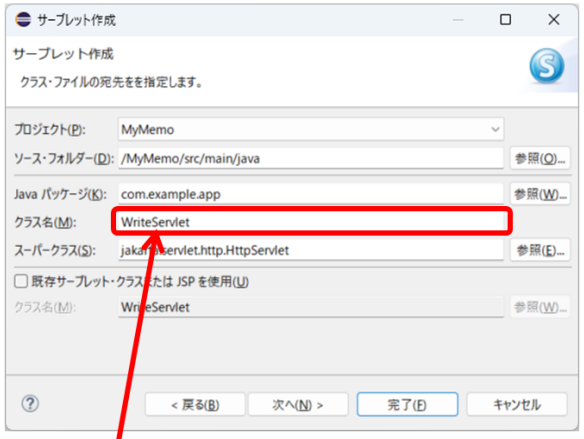
「次へ」をクリック

WriteServletの作成



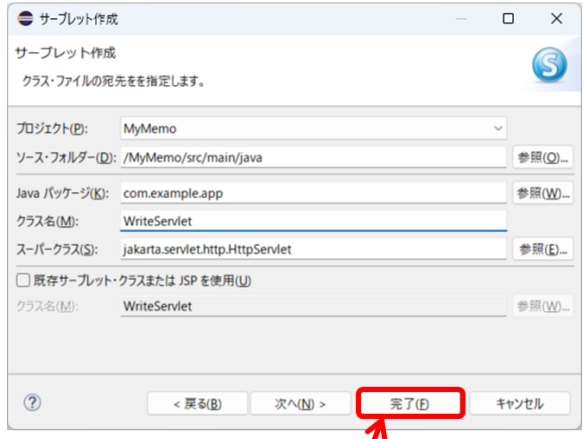
Javaパッケージに「com.example.app」と入力

WriteServletの作成



クラス名に「WriteServlet」と入力

WriteServletの作成



「完了」をクリック

WriteServletの作成

WriteServlet.java

```
1  package com.example.app;
2
3  import java.io.IOException;
4
5  import jakarta.servlet.ServletException;
6  import jakarta.servlet.http.HttpServlet;
7  import jakarta.servlet.http.HttpServletRequest;
8  import jakarta.servlet.http.HttpServletResponse;
9
10 public class WriteServlet extends HttpServlet {
11     private static final long serialVersionUID = 1L;
12
13     protected void doPost(HttpServletRequest request, HttpServletResponse
14             throws ServletException, IOException{
15
16
17
18         //パラメータ取得
19         String title = request.getParameter("title");
20         String memo = request.getParameter("memo");
21
```

WriteServlet.javaを編集して保存 → ソース全体は別添2参照

ShowServletの作成

- サーブレットShowServletはトップページから「表示」リンクをクリックしたときにコールされる。このサーブレットでは次のことを行なう。
  - ➢DBAccess.select()を使ってDBからデータの抽出
  - ➢リクエストオブジェクトにデータをセット
  - ➢/show.jspの呼び出し

ShowServletの作成

- ShowServletのdoGet()メソッド内は次のようになる。

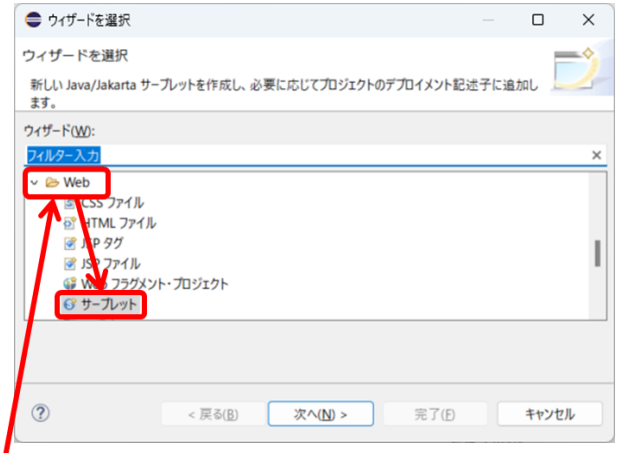
```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                       throws ServletException, IOException {

   //DBからデータの取得
   DBAccess dba = new DBAccess(request);
   ArrayList<Line> list = dba.select();

   //リクエストオブジェクトにデータをセット
   request.setAttribute("dbdata", list);

   //表示ページの呼び出し
   request.getRequestDispatcher("/show.jsp")
          .forward(request, response);
}
```

次のインポート文が必要：
import java.util.ArrayList;

ShowServletの作成

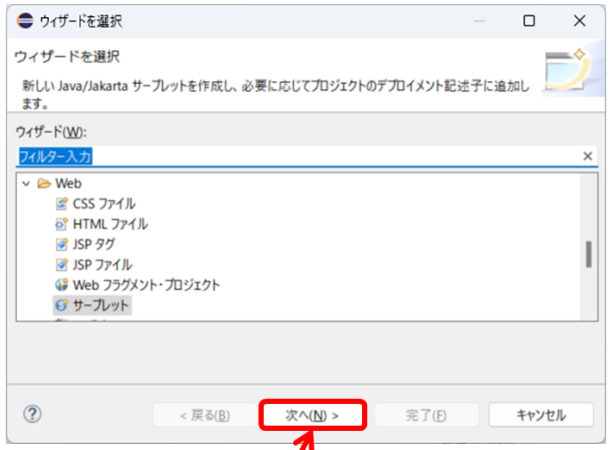
プロジェクト名MyMemoを右クリックして「新規 → その他」を選択

ShowServletの作成



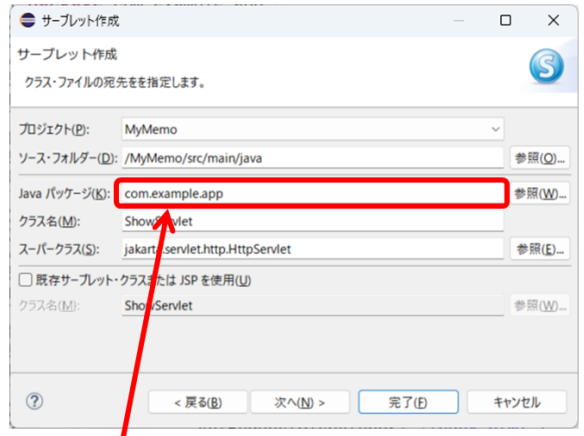
Webを展開して「サーブレット」を選択

ShowServletの作成



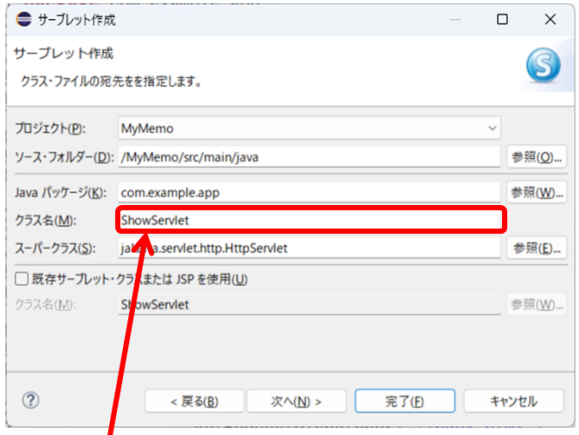
「次へ」をクリック

ShowServletの作成



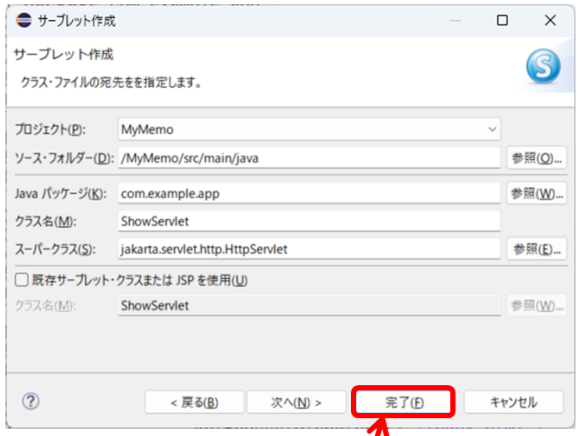
Javaパッケージに「com.example.app」と入力

ShowServletの作成



クラス名に「ShowServlet」と入力

ShowServletの作成



「完了」をクリック

ShowServletの作成

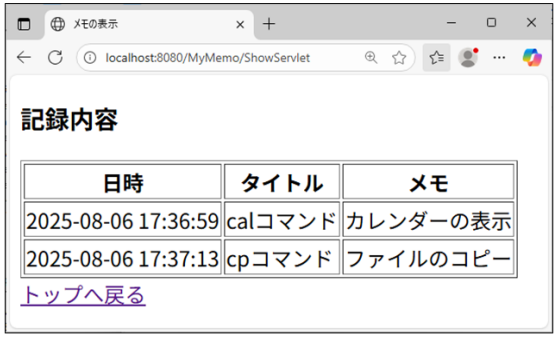
```java
package com.example.app;

import java.io.IOException;
import java.util.ArrayList;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class ShowServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

        //DBからデータの取得
        DBAccess dba = new DBAccess(request);
        ArrayList<Line> list = dba.select();

        //リクエストオブジェクトにデータをセット
        request.setAttribute("dbdata", list);

        //表示ページの呼び出し
        request.getRequestDispatcher("/show.jsp").forward(request, response);
    }
}
```

[Ctrl] + [Shift] + [o] で
インポート文を追加する！

ShowServlet.javaを編集して保存 → ソースは別添3参照

show.jspの作成

●結果を表示するshow.jspではrequestにセットされたListオブジェクトを、getAttribute()メソッドを使って取り出し、繰り返しを使って表形式で表示する。



show.jspの作成

show.jsp(1/2)

```jsp
<%@ page language="java"
         contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
<%@ page import="com.example.app.*, java.util.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>メモの表示</title>
</head>
<body>
    <%
      ArrayList<Line> list =
                  (ArrayList<Line>)request.getAttribute("dbdata");
    %>
```

import文が必要！

show.jspの作成

show.jsp(2/2)

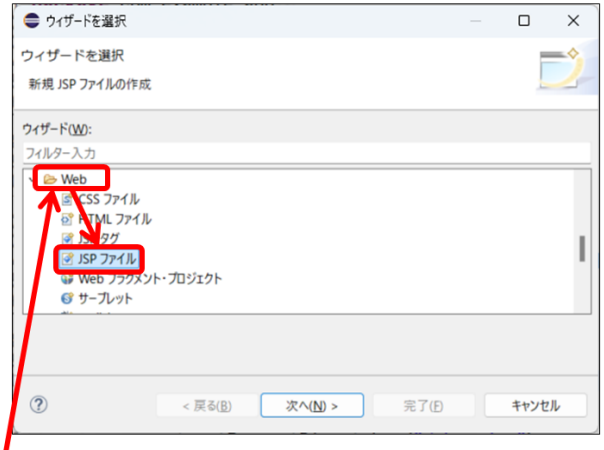
```jsp
    <h3>記録内容</h3>
    <table border="1">
    <tr><th>日時</th><th>タイトル</th><th>メモ</th></tr>
    <% for(Line d : list){ %>
        <tr>
            <td><%= d.getTm() %></td>
            <td><%= d.getTitle() %></td>
            <td><%= d.getMemo() %></td>
        </tr>
    <% } %>
    </table>
    <a href="/MyMemo/index.html">トップへ戻る</a>
</body>
</html>
```

show.jspの作成

プロジェクト名MyMemoを右クリックして「新規 → その他」を選択

77

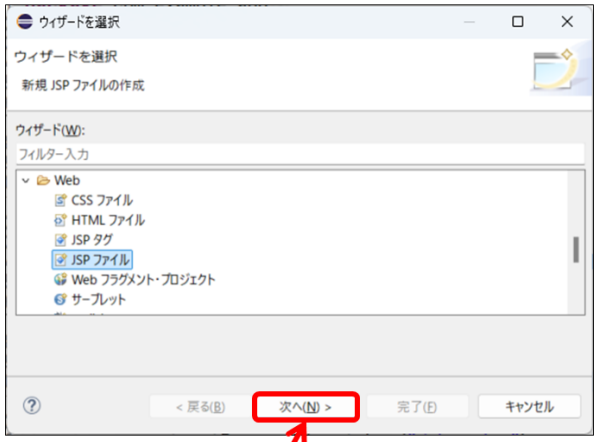
show.jspの作成



Webを展開して「JSPファイル」を選択

78

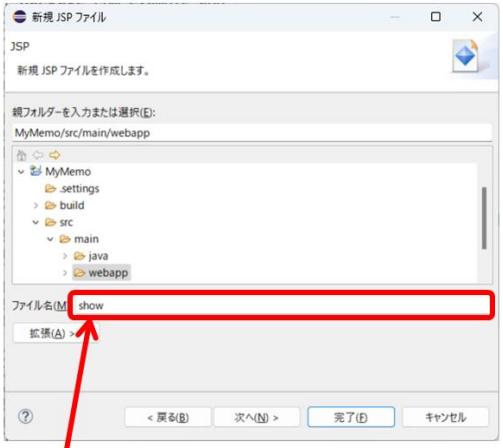
show.jspの作成



「次へ」をクリック

79

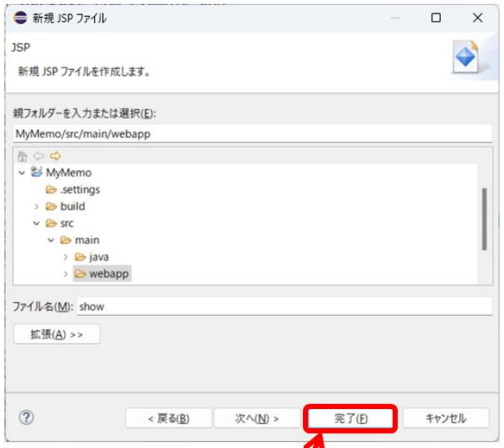
show.jspの作成



ファイル名に「show」と入力

80

show.jspの作成



「完了」をクリック

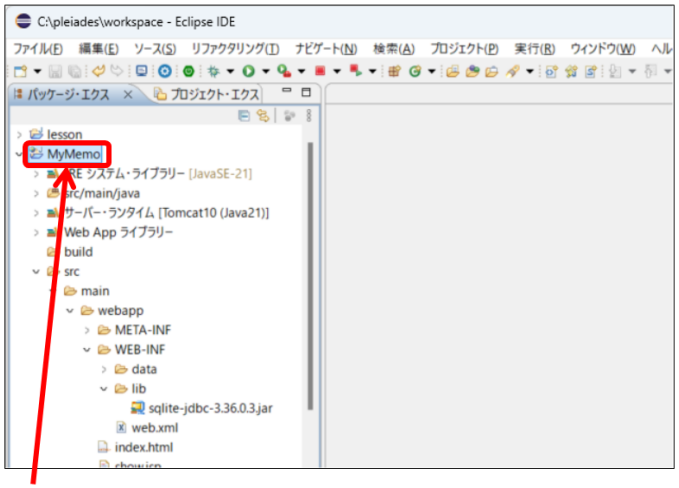
show.jspの作成

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <%@ page import="com.example.app.*, java.util.*" %>
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title>メモの表示</title>
9  </head>
10 <body>
11     <%
12         ArrayList<Line> list = (ArrayList<Line>)request.getAttribute("dbdata");
13     %>
14     <h3>記録内容</h3>
15     <table border="1">
16     <tr><th>日時</th><th>タイトル</th><th>メモ</th></tr>
17     <% for(Line d : list){ %>
18             <tr>
19                 <td><%= d.getTm() %></td>
20                 <td><%= d.getTitle() %></td>
21                 <td><%= d.getMemo() %></td>
22             </tr>
23     <% } %>
24     </table>
```

import文が必要！

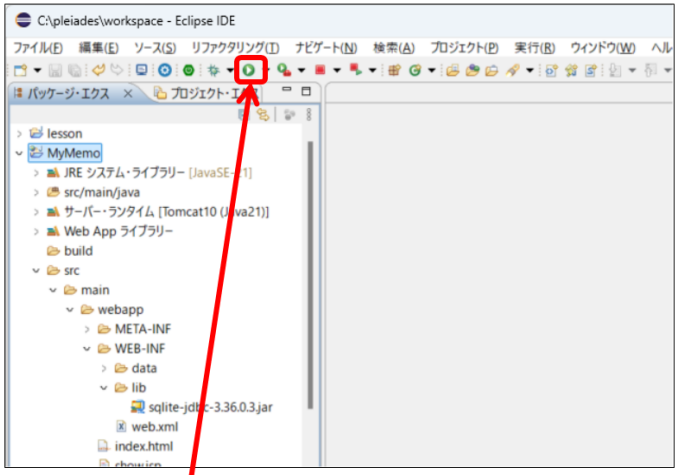
show.jspを編集して保存 → ソースは別添4参照

実行



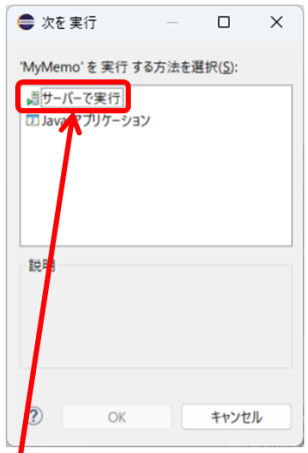
プロジェクト名の「MyMemo」をクリック

実行



実行ボタン「 」をクリック

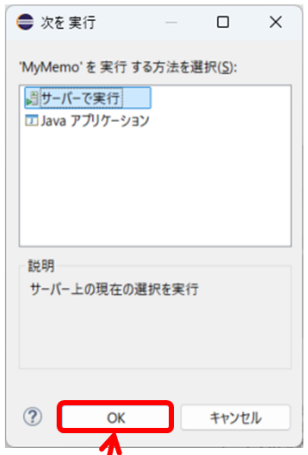
実行



「サーバーで実行」をクリック

85

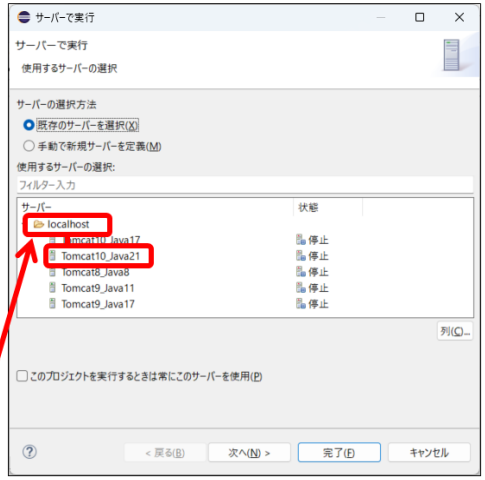
実行



「OK」をクリック

86

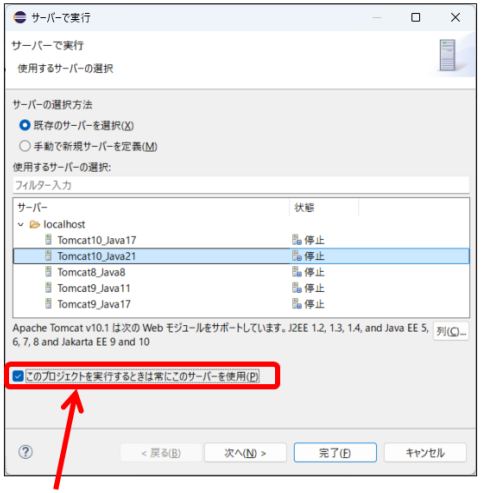
実行



「localhost → Tomcat10_Java21」を選択

87

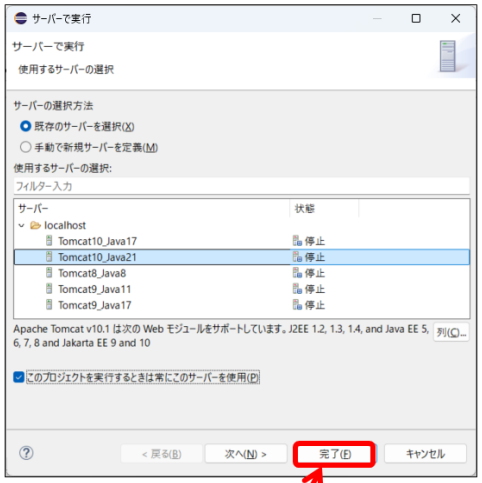
実行



「このプロジェクトを実行するときは…」にチェック

88

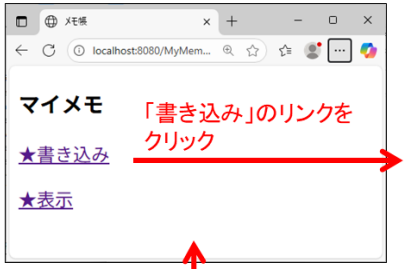
実行



「完了」をクリック

実行

index.html



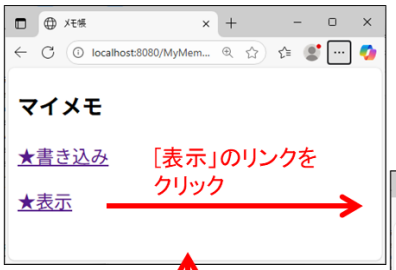
「書き込み」のリンクを
クリック

write.html

タイトルとメモを入力して「記録」をクリック
→ DBに書き込んでトップページに戻る

実行

index.html



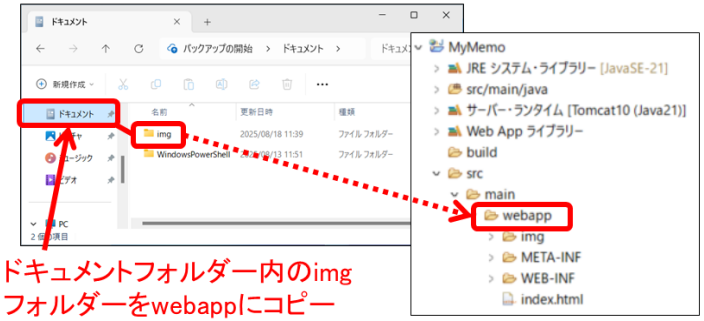
「表示」のリンクを
クリック

show.jsp

DBに保存されているメモ内容を表示

CSSの利用

- JSPでもCSSと同様にスタイルを指定できる。ここでは簡単に次のような内容でスタイルを設定する。
- 画像ファイルは「src/main/webapp」に配置する。



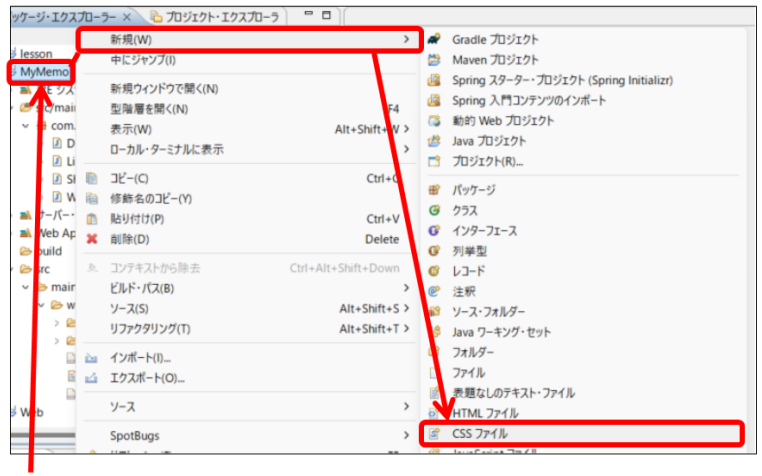
ドキュメントフォルダー内のimg
フォルダーをwebappにコピー

- 今回のCSSは次頁のようにする。

```css
@charset "UTF-8";
body, textarea{ /* bodyとtextareaの設定 */
   font-family: sans-serif; /* フォント */
}
body{ /* bodyの設定 */
   background-color:#adff2f; /* 背景色 */
   background-image:url("img/bpic.jpg"); /* 背景画像 */
}
table{ /* テーブルの設定 */
   background-color: #ffffffc0; /* 背景色を透かした白にする */
}
td, th { /* テーブルセルの設定 */
   padding:3px; /* セルの内容と線の間隔 */
}
a{ /* リンクの設定 */
   text-decoration: none; /* アンダーラインを表示しない */
}
.pre{ /* class="pre"の設定 */
   white-space: pre;  /* ソースの改行を表示でも有効にする */
}
```

93

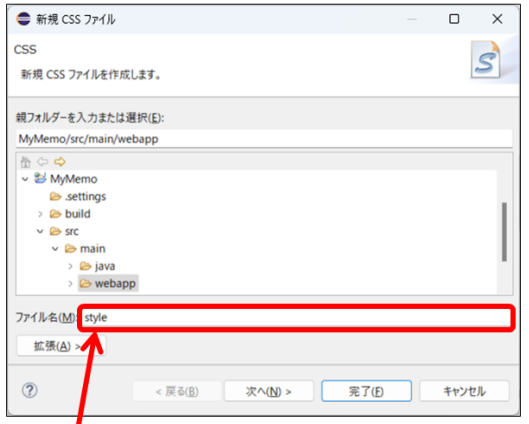
CSSの利用



プロジェクト名MyMemoを右クリックして「新規 → CSSファイル」を選択

94

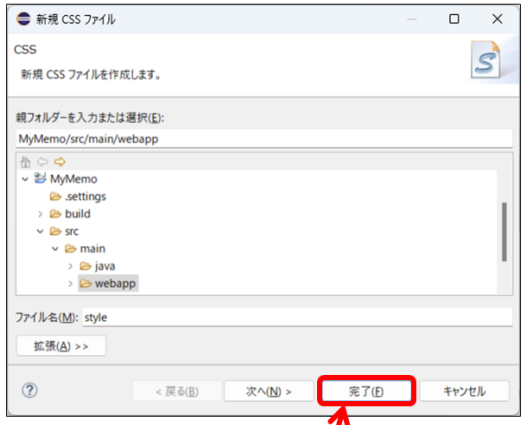
CSSの利用



ファイル名に「style」と入力

95

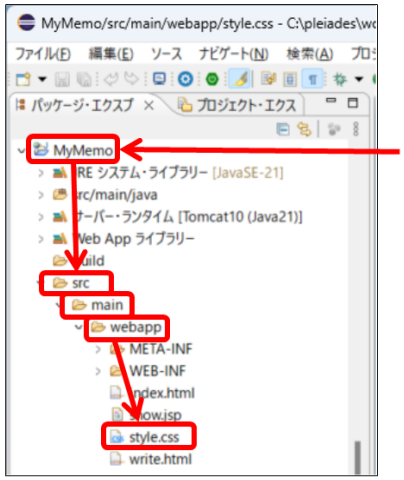
CSSの利用



「完了」をクリック

96

CSSの利用



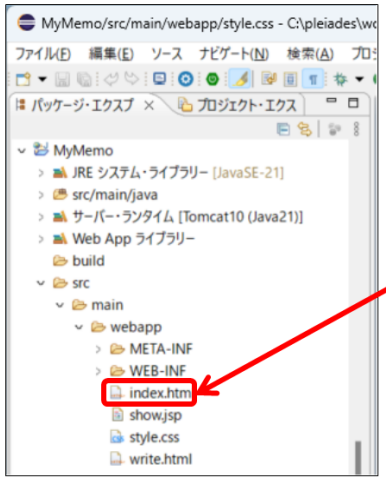
「MyMemo → src
→ main → webapp」に
style.cssが作成される

CSSの利用

style.cssを編集して保存 → ソースは別添5参照

```
@charset "UTF-8";
body, textarea{
    font-family: sans-serif; /* フォント */
}
body{
    background-color: cornflowerblue; /* 背景色 */
    background-image:url("img/bpic.jpg"); /* 背景画像 */
}
table{
    background-color: #ffffffc0; /* 背景色を透かした白にする */
}
td, th {
    padding:3px; /* セルの内容と線と間隔 */
}
a{
    text-decoration: none; /* リンクのアンダーラインを表示しない */
}
.pre{
    white-space: pre;  /* 改行を有効にする */
}
```


CSSの利用



index.htmlをダブル
クリックして開く

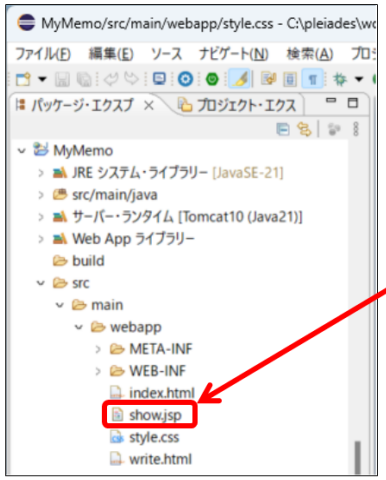
CSSの利用

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>メモ帳</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
    <h3>マイメモ</h3>
    <p><a href="/MyMemo/write.html">★書き込み</a></p>
    <p><a href="/MyMemo/ShowServlet">★表示</a></p>
</body>
</html>
```

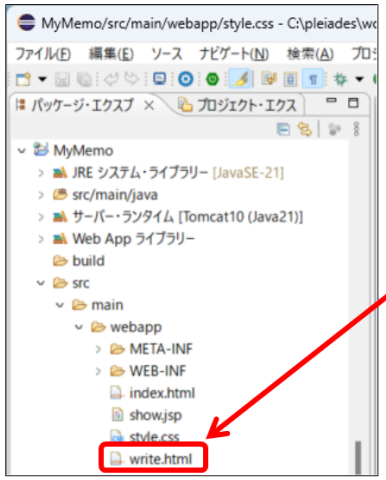
head内に
<link rel="stylesheet" href="style.css" />
を追加

CSSの利用



show.jspをダブルクリックして開く

CSSの利用



write.htmlをダブルクリックして開く

CSSの利用

```
5  <!DOCTYPE html>
6  <html>
7  <head>
8  <meta charset="UTF-8">
9  <title>メモの表示</title>
10 <link rel="stylesheet" href="style.css">
11 </head>
```

head内に <link rel="stylesheet" href="style.css" /> を追加

テーブル表示の一番下の<td>に class="pre" を追加

```
20    <% for(Line d : list){ %>
21        <tr>
22            <td><%= d.getTm() %></td>
23            <td><%= d.getTitle() %></td>
24            <td class="pre"><%= d.getMemo() %></td>
25        </tr>
26    <% } %>
```

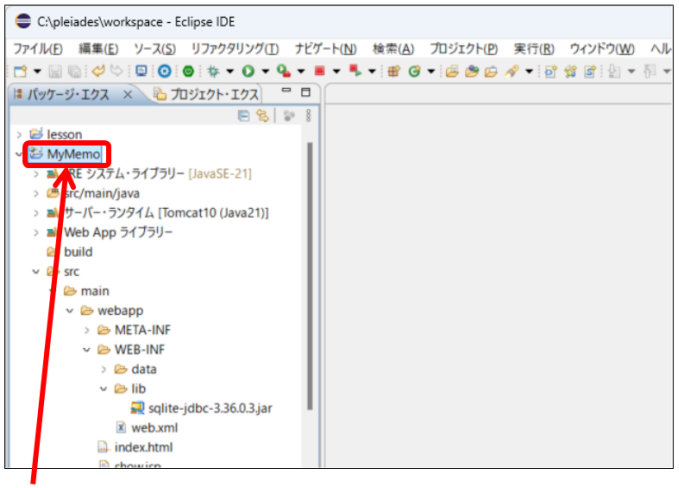

CSSの利用

write.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8">
5  <title>メモの書き込み</title>
6  <link rel="stylesheet" type="text/css" href="style.css" />
7  </head>
8  <body style="font-family:sans-serif">
9      <form action="/MyMemo/WriteServlet" method="POST">
10     <p>タイトル<br><input type="text" name="title" size="30"></p>
11     <p>メ モ<br>
12         <textarea name="memo" cols="30" rows="5"></textarea></p>
13     <p><input type="submit" value="記録"></p>
14     </form>
15 </body>
16 </html>
```

head内に
<link rel="stylesheet" type="text/css" href="style.css" />
を追加

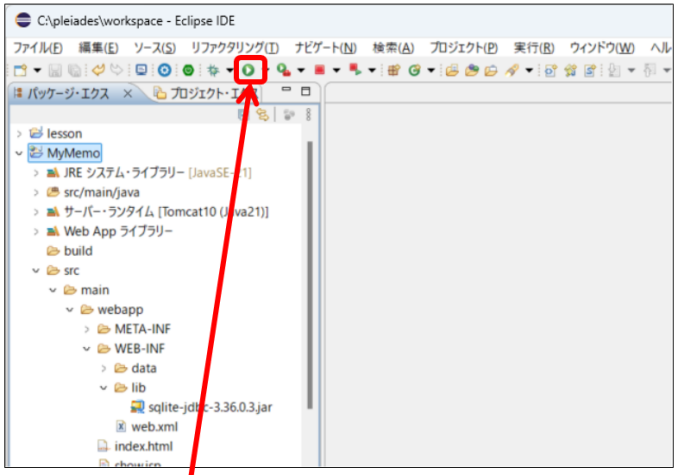
実行



プロジェクト名の「MyMemo」をクリック

105

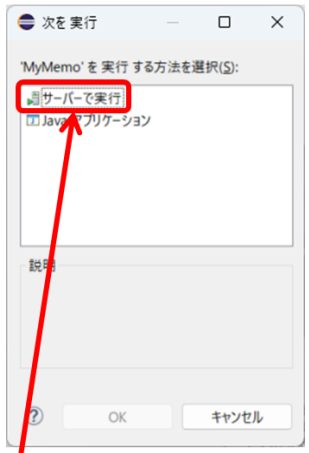
実行



実行ボタン「 」をクリック

106

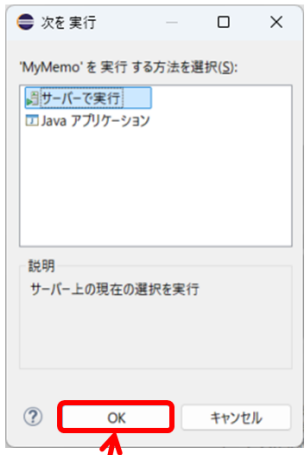
実行



「サーバーで実行」をクリック

107

実行

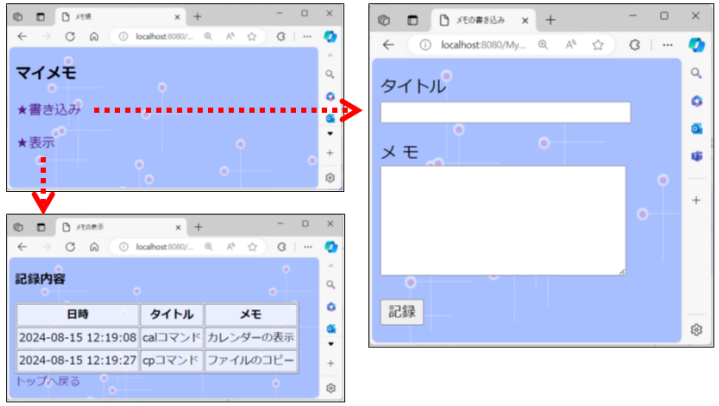


「OK」をクリック

108

実行

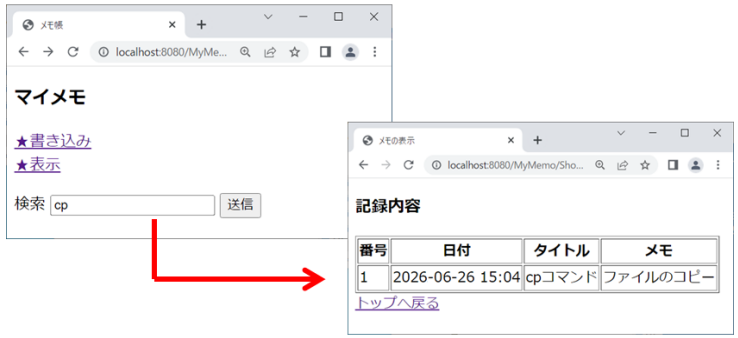
●実行するとスタイルが設定されている。



問題

① 例題と同じプログラムをプロジェクト名をMyAppとして作成せよ。ここで、データベース名はmyapp.dbとし、テーブル名はmyapp_tableとすること。

② 例題にタイトルによる検索機能を追加せよ。



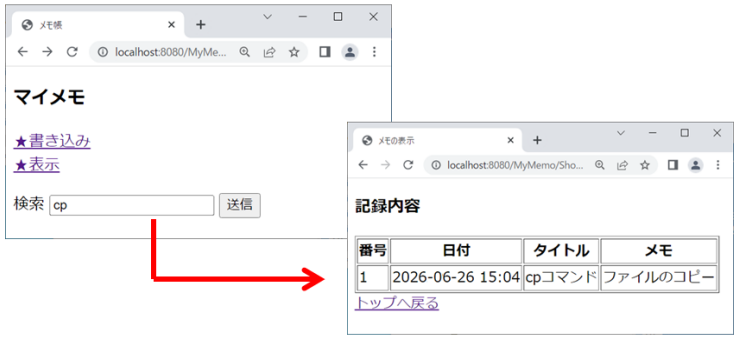
サーバーサイドJava

6. Servlet, JSP, DBの連携
解答例

問題

① 例題と同じプログラムをプロジェクト名をMyAppとして作成せよ。ここで、データベース名はmyapp.dbとし、のテーブル名はmyapp_tableとすること。→ 略

② 例題にタイトルによる検索機能を追加せよ。



問題②

index.html(変更)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>メモ帳</title>
</head>
<body>
   <h3>マイメモ</h3>
   <p> <a href="/MyMemo/write.html">★書き込み</a><br>
       <a href="/MyMemo/ShowServlet">★表示</a></p>
       <form action="/MyMemo/SearchServlet">
          検索
          <input type="text" name="key">
          <input type="submit">
       </form>
</body>
</html>
```

追加

問題②

DBAccess.java(メソッドを追加)

```
   public ArrayList<Line> search(String key) {
      String sql = "SELECT * FROM memo_table WHERE title LIKE ?";
      ArrayList<Line> list = new ArrayList<>();
      try (Connection con = getConnection();
            PreparedStatement ps = con.prepareStatement(sql) ){
            ps.setString(1, "%" + key + "%"); //曖昧検索
            ResultSet rs = ps.executeQuery();
         while (rs.next()) {
            Line line = new Line(rs.getInt(1), rs.getString(2),
                  rs.getString(3), rs.getString(4));
            list.add(line);
         }
      } catch (SQLException | ClassNotFoundException e) {
         e.printStackTrace();
      }
      return (list);
   }
```


問題②

SearchServlet.java(新規に作成:ShowServletと殆ど同じ)

```
/* import等略 */
public class SearchServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;

   protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                              throws ServletException, IOException {
      String key = request.getParameter("key");

      DBAccess dba = new DBAccess(request);
      ArrayList<Line> list = dba.search(key);

      request.setAttribute("dbdata", list);

      request.getRequestDispatcher("/show.jsp")
            .forward(request, response);
   }
}
```