Javaサーブレット環境構築

各種Ver
JDK 7
Apache HTTP Server 2.2.x
Apache tomcat 6

DL先
JDK
https://jdk.java.net/java-se-ri/7
jdk_ri-7u75-b13-windows-i586-18_dec_2014.zip

Apache HTTP Server
https://archive.apache.org/dist/httpd/binaries/win32/
httpd-2.2.25-win32-x86-openssl-0.9.8y.msi

Apache tomcat
https://archive.apache.org/dist/tomcat/tomcat-6/
apache-tomcat-6.0.0.zip

md_jk
https://archive.apache.org/dist/tomcat/tomcat-connectors/jk/binaries/windows/
tomcat-connectors-1.2.40-windows-i386-httpd-2.2.x.zip

3. 互換性の問題

古いソフトウェア	対応 Java バージョン	サポート状況
Tomcat 6	Java 6, 7, 8	サポート終了
Tomcat 7	Java 7, 8	サポート終了
Tomcat 8	Java 8, 11	サポート終了
Tomcat 9	Java 11, 17	サポートあり
Apache 2.2	-	サポート終了
Apache 2.4	-	サポート中

インストール
JDK
jdk_ri-7u75-b13-windows-i586-18_dec_2014.zip
を解凍して、フォルダを「jdk7」に変更

JAVA_HOME に C:\jdk7 を設定
Path に %JAVA_HOME%\bin を追加

java -version を実行し、インストール確認

Apache HTTP Server

httpd-2.2.25-win32-x86-openssl-0.9.8y.msi
を実行しインストール
インストール先はC:¥
セキュリティ周りで管理者権限が必要

ServerNameやメアドは何がいい？

httpd -k start 管理者権限で！
httpd -k stop
httpd -k restart

http://localhost/ で It works!が表示される

もし独自のウェブサイトやアプリケーションを Apache に配置したい場合、以下の作業が必要です：

1. **ウェブサイトのファイルを配置する**: Apache のドキュメントルート（通常は `/var/www/html` または `C:\Program Files (x86)\Apache Group\Apache2\htdocs`）に、自分の HTML ファイルやアプリケーションファイルを配置します。
2. **Apache の設定を変更する**: 必要に応じて、Apache の設定ファイル（`httpd.conf` や `000-default.conf`）を編集して、ホスト名やポートの設定を変更します。

Apache tomcat
<起動>
apache-tomcat-6.0.0.zip
を解凍
bin/startup.bat実行

http://localhost:8080/ でtomcatのページが表示される。

＜自動起動設定＞
1.サービス登録
cmd 管理者権限で
cd C:¥apache-tomcat-6.0.0¥bin
service.bat install
⇒Apache Tomcatというサービス名で登録される。

2.自動起動設定
services.msc で「サービス」管理画面を開く
「スタートアップの種類」を 自動 に変更し、「適用」→「OK」

3.手動でのサービス起動コマンド
net start "Apache Tomcat"

設定
httpd.conf

```
188 #↵
187 # First, we configure the "default" to be a very restrictive set of ↵
189 # features.  ↵
189 #↵
190 <Directory />↵
191     Options FollowSymLinks↵
192     AllowOverride None↵
193     #Order deny,allow↵
194     #Deny from all↵
195 </Directory>↵
196 ↵
```

参考
https://weblabo.oscasierra.net/installing-apache22-windows-1/?utm_source=chatgpt.com

Servlet

classesフォルダ作成し、下記javaファイル作成。

C:\apache-tomcat-6.0.0\webapps\testapp\WEB-INF\classes\HelloServlet.java - sakura 2.4.1.2849

ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.getWriter().println("Hello, World!");
    }
}
```

[EOF]

下記でもイケルバズ。

```
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Hello from Java Servlet!</h1>");
        out.println("<p>現在の時間: " + new Date() + "</p>");
        out.println("</body></html>");
    }
```

C:¥apache-tomcat-6.0.0¥webapps¥testapp¥WEB-INF¥classes に移動

javac -cp "C:¥apache-tomcat-6.0.0¥lib¥servlet-api.jar" HelloServlet.java

web.xml作成

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                            http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">

    <!-- サーブレットの定義 -->
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>HelloServlet</servlet-class>
    </servlet>

    <!-- サーブレットマッピング -->
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

http://localhost:8080/testapp/hello

Apache-tomcat連携

＜Apache側＞

tomcat-connectors-1.2.40-windows-i386-httpd-2.2.x.zip

を解凍し、mod_jk.soをmodulesフォルダへ格納

httpd.conf編集

```
<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>

# mod_jk モジュールのロード
LoadModule jk_module modules/mod_jk.so

# workers.properties
JkWorkersFile conf/workers.properties

# ログ設定
JkLogFile logs/mod_jk.log
JkLogLevel info

# Tomcatへのリクエスト転送設定
JkMount /testapp/* worker1
JkMount /*.jsp worker1
```

[EOF]

workers.properties編集

C:\Apache2.2\conf\workers.properties - sakura 2.4.1.2849

ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ

```
# 基本設定
workers.tomcat_home=C:/apache-tomcat-6.0.0
workers.java_home=C:/jdk7
ps=¥

# ワーカー設定
worker.list=worker1
worker.worker1.port=8009
worker.worker1.host=localhost
worker.worker1.type=ajp13
```

[EOF]

＜tomcat側＞

conf/server.xml確認

workers.propertiesとあっていること。

```
    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

http://localhost/testapp/hello　　⇒Apache経由でtomcat

Apache経由の利点：

Eclipse

＜公式＞

新Ver　　設定回り

https://www.eclipse.org/downloads/packages/　　https://www.kkaneko.jp/tools/ubuntu/linux_eclipse.html

旧Ver

https://archive.eclipse.org/eclipse/downloads/

⇒ここからは、ChromeではDLがブロックされるかも。下記で対処。

・公式であること

・ウィルススキャンすること

・shaで改善されていないこと

CertUtil -hashfile eclipse-SDK-4.12-win32-x86_64.zip SHA256

Tomcat 6 サポートのある Eclipse バージョン

Eclipse 3.6（Helios） ～ Eclipse 3.8（Juno） が、Tomcat 6 のサポートが確認されています。

これらのバージョンは、Eclipse IDE としてServlet 2.5（Tomcat 6） とServlet 3.0（Tomcat 7 以降） のサポートを提供します。

Eclipse本体もJDK7（JRE7）で動作するため、これがよい。

推奨されるバージョン

Eclipse 3.6（Helios）

Eclipse 3.7（Indigo）

Eclipse 3.8（Juno）

推奨されるEclipse3.6~3.8は、以下のプラグインが同梱されておらず、Servlet開発ができない。

別途ダウンロードが必要だが、tomcat6（servlet2.5）対応の古いダウンロード先が見つからず。

WTP (Web Tools Platform)　　⇒Eclipse 上での Web (J2EE) アプリケーション開発を行うためのツール群を集めた plug-in 。

日本語化のプラグインも見つからず。

pleiades版なら、WTPも同梱済であり、日本語化もされているため、pleiades版を使用するものとする。

＜Pleades All in One＞

https://willbrains.jp/

Eclipse 4.4（Luna）　　⇒Eclipse本体がJDK8。

Eclipse 4.3（Kepler）　　⇒採用。Eclipse本体がJDK7の最終Ver。

Eclipse 4.2（Juno）

Eclipse 3.8（Juno）

Eclipse 3.7（Indigo）

Eclipse 3.6（Helios）

JDKやtomcatは本番環境のVerに合わせたいため、StanderdEditonを使用する。

EnterpriseEditonでは、JDKやtomcatなど不要なものはいっているため、管理が煩雑となるため、StanderdEditonでOK。

Tomcatのプロジェクト作成参考：

https://ameblo.jp/kurabeat/entry-10551325515.html　　⇒「Tomcat をインストール済みの環境であっても、

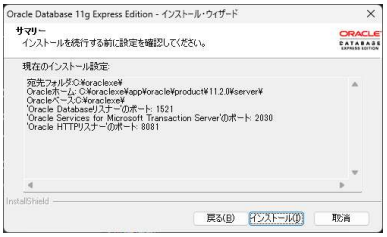
https://ameblo.jp/kurabeat/entry-10552221943.html　　Eclipse 専用の Tomcat を別途インストールする必要があります。」とあるが、

https://ameblo.jp/kurabeat/entry-10553212913.html　　開発環境ならインストール済のものを参照すればOK。

oracle11g

Oracle 11g (11g の代替として 11g Express Edition)

https://www.oracle.com/database/technologies/xe-prior-release-downloads.html

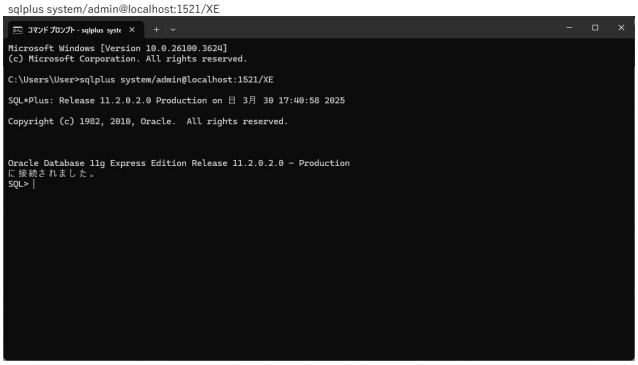


pass
admin

sqlplus system/パスワード@localhost:1521/XE
sqlplus system/admin@localhost:1521/XE → エラーが出る場合：

lsnrctl status を実行し、リスナーが動作しているか確認

lsnrctl start で手動起動



エラーが **一時的なファイルの作成ミス** や **レジストリの警告** なら無視しても OK ですが、**重要なコンポーネントのエラー** だと問題になることがあります。

無視してよいエラー例

xxxxxKEY_XE.reg をインスタンス化できませんでした。

Windows の互換性の警告

一時ファイルのアクセス権限エラー（動作に問題がなければ無視可）

無視しないほうがよいエラー例

OracleServiceXE が開始できませんでした。

リスナーが起動しませんでした。

SQL*Plus で接続できない。

jdbc6
https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc6

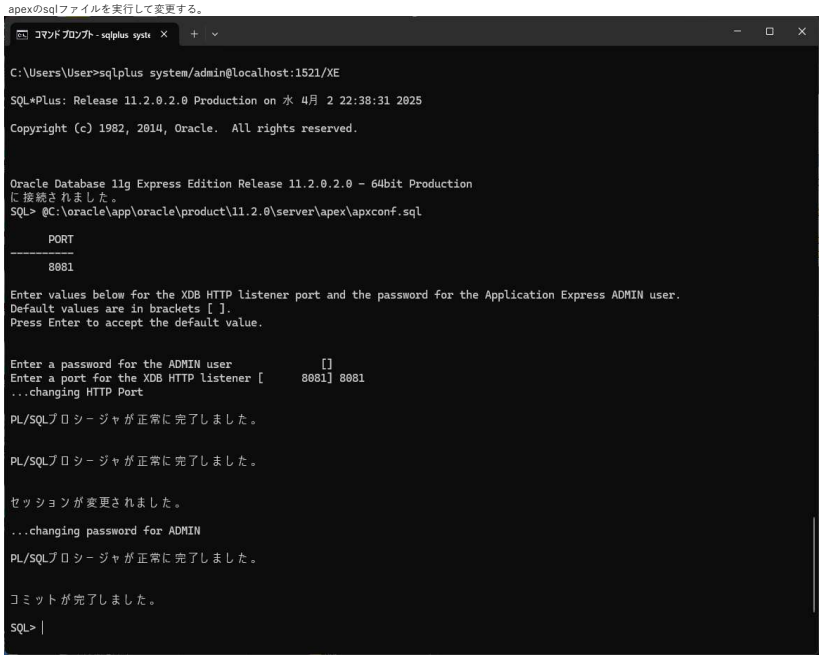
C:¥apache-tomcat-6.0.0¥webapps¥testapp¥WEB-INF¥classes に移動
javac -cp "C:¥apache-tomcat-6.0.0¥lib¥servlet-api.jar;C:¥apache-tomcat-6.0.0¥lib¥ojdbc6-11.2.0.4.jar" DbTestServlet.java
javac -cp "C:¥apache-tomcat-6.0.0¥lib¥servlet-api.jar" HelloServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:XE", "system", "password");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT sysdate FROM dual");

            out.println("<h1>Database Time:</h1>");
            while (rs.next()) {
                out.println("<p>" + rs.getString(1) + "</p>");
            }

            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
            out.println("<p>Error: " + e.getMessage() + "</p>");
        }
    }
}
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        // ★★★ ここを編集 ★★★
        String url = "jdbc:oracle:thin:@localhost:1521:XE";  // DB接続URL
        String user = "system";  // DBユーザー名
        String password = "password";  // DBパスワード

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver"); // JDBCドライバのロード
            Connection conn = DriverManager.getConnection(url, user, password);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT 'Hello, Oracle!' FROM dual");

            while (rs.next()) {
                out.println("<h1>" + rs.getString(1) + "</h1>");
            }

            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception e) {
            out.println("<h1>Database Connection Error</h1>");
            out.println("<p>" + e.getMessage() + "</p>");
        } finally {
            out.close();
        }
    }
}
```

oracleのWebUIのポート番号変更

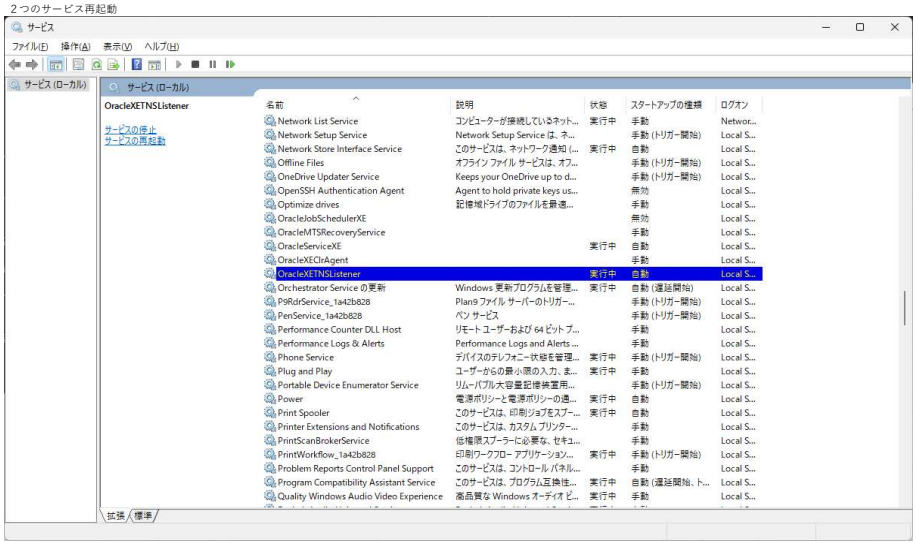
apexのsqlファイルを実行して変更する。



⇒sql発行。デフォのインストールフォルダは「oraclexe」

⇒管理者ユーザ名（system）のパスワード変更も対象となる模様。
⇒競合しないポート番号を指定

２つのサービス再起動



Oracle SQL Developer ダウンロード

https://www.oracle.com/technetwork/jp/developer-tools/sql-developer/downloads/index.html

Apache-Tomcat連携モジュール

✅ mod_proxy_ajp vs mod_proxy_http vs mod_jk の違い

この3つはすべて Apache HTTP Server から Tomcat へリクエストを転送 するための方法ですが、それぞれ使うプロトコルや特徴が異なります。

項目	mod_proxy_ajp	mod_proxy_http	mod_jk
プロトコル	AJP (Apache JServ Protocol)	HTTP (通常のウェブ通信)	AJP (独自実装)
ポート	8009	8080	8009
通信方式	バイナリ転送（高速）	テキスト転送（標準）	バイナリ転送（最適化）
パフォーマンス	高速	遅め（HTTP のオーバーヘッドあり）	mod_proxy_ajp より高速な場合あり
セキュリティ	AJP は 脆弱性がある ⚠	HTTPS/TLS 利用可（安全）	AJP は 脆弱性がある ⚠
ロードバランス	一応可能（mod_proxy_balancer）	可能（mod_proxy_balancer併用）	可能（workerで詳細設定）
設定の簡単さ	普通（Tomcat で AJP 有効化が必要）	簡単（Tomcat のデフォルト HTTP を使うだけ）	難しい（worker 設定が必要）
Tomcat 負荷	低い（Tomcat に負担をかけない）	高い（Tomcat 側で処理が増える）	低い（Tomcat に最適化）
Windows での安定性	普通	安定	安定（ただし設定が複雑）

🚀 どれを選ぶべき？

ケース	おすすめのモジュール
シンプルな構成で簡単に動かしたい	✅ mod_proxy_http
HTTPS (SSL) を使いたい	✅ mod_proxy_http
最大限のパフォーマンスを出したい	✅ mod_proxy_ajp
Tomcat のリソースを節約したい	✅ mod_proxy_ajp
セキュリティ重視（AJP を避けたい）	✅ mod_proxy_http
負荷分散や詳細な制御が必要	✅ mod_jk
Windows で安定して運用したい	✅ mod_proxy_http or mod_jk

💡 結論

- 手軽 & 安全な選択肢 → mod_proxy_http
- パフォーマンス重視 → mod_proxy_ajp
- 複雑なロードバランシングが必要 → mod_jk

1 mod_proxy_ajp（AJP を使う）

📌 Apache 設定（httpd.conf）

```
apache
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so

<VirtualHost *:80>
    ServerName example.com

    ProxyRequests Off
    ProxyPass / ajp://localhost:8009/
    ProxyPassReverse / ajp://localhost:8009/
</VirtualHost>
```

📌 Tomcat 設定（server.xml）

```
xml
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

✅ メリット ✓ AJP のバイナリ転送で 高速
✓ Apache 側でキャッシュできるので Tomcat の負荷が低い
❌ デメリット ❌ AJP の脆弱性（外部公開すると攻撃されるリスクあり）
❌ Apache 2.2 では バグがあった（2.4 なら OK）

2 mod_proxy_http（通常の HTTP を使う）

📌 Apache 設定（httpd.conf）

```
apache
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

<VirtualHost *:80>
    ServerName example.com

    ProxyRequests Off
    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/
</VirtualHost>
```

📌 Tomcat 設定（特に不要） Tomcat はデフォルトで 8080 ポートが開いているので、そのまま使える。

✅ メリット ✓ 設定が簡単（Tomcat のデフォルト HTTP を使うだけ）
✓ HTTPS が使える（セキュリティが強い）
❌ デメリット ❌ HTTP のオーバーヘッドがあるため、パフォーマンスが低い
❌ Tomcat に負荷がかかる（Apache 側で処理を減らせない）

3 mod_jk（AJP を最適化した専用モジュール）

📌 Apache 設定（httpd.conf）

```
apache
LoadModule jk_module modules/mod_jk.so

JkWorkersFile "C:/Apache24/conf/workers.properties"
JkMount /servlet/* ajp13_worker
```

📌 workers.properties（Tomcat との接続設定）

```
properties
worker.list=ajp13_worker

worker.ajp13_worker.type=ajp13
worker.ajp13_worker.host=localhost
worker.ajp13_worker.port=8009
```

📌 Tomcat 設定（server.xml）

```
xml
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

✅ メリット ✓ パフォーマンスが最も高い（最適化された AJP 転送）
✓ 負荷分散（ロードバランシング）を柔軟に設定可能
✓ リクエストログを細かく制御できる
❌ デメリット ❌ 設定が複雑（worker 設定が必要）
❌ mod_jk.so を別途インストールする必要がある
❌ Windows では セットアップが面倒

jspアプリ

2. シンプルなJSPアプリの作成

2-1. JSPファイルの作成

1. `C:\tomcat\webapps\myapp\` に `index.jsp` を作成
2. 以下のコードを記述

jsp　コピーする　編集する

```jsp
<html>
<head><title>JSP Sample</title></head>
<body>
    <h1>こんにちは、Tomcat！</h1>
    <p>現在の時間: <%= new java.util.Date() %></p>
</body>
</html>
```

3. Tomcatを再起動し、`http://localhost:8080/myapp/index.jsp` にアクセス

3. 動作確認

- `http://localhost/myapp/` にアクセスし、JSPのページが表示されるか確認
- Apache HTTP Server経由（`http://localhost/myapp/`）でも動作するか確認

---

4. 自動起動の設定（オプション）

TomcatをWindowsサービスとして登録

1. コマンドプロンプトを管理者権限で開く
2. `cd C:\tomcat\bin`
3. `service.bat install` を実行
4. Windowsの「サービス」からTomcatをスタート/ストップ可能