



USAID
FROM THE AMERICAN PEOPLE

MOBILE APPLICATION TO SECURE TENURE- BURKINA FASO

Development Environment Installation Setup
Guide (Version 1.0)

Abstract

This document provides detail of development environment setup, including the download and installation of various software/tools and source code.

Contents

1	INTRODUCTION	2
2	INSTALL AND CONFIGURE MAST DEVELOPMENT TOOLS	3
2.1	MAST Burkina Faso Web Infrastructure.....	3
2.1.1	Installing Java JDK 7	3
2.1.2	Setting Java and JRE Home	5
2.1.3	Installing PostgreSQL	6
2.1.4	Verifying the Installation.....	9
2.1.5	Installing PostGIS	10
2.1.6	Creating and populating database from script.....	14
2.1.7	Apache Tomcat Installation.....	15
2.1.8	Apache Tomcat Configuration.....	18
2.1.9	Eclipse Luna	22
2.2	MAST Mobile Application (Android)	23
2.2.1	Installing Java JDK 7	23
2.2.2	Installing the Stand-alone SDK Tools.....	23
2.2.3	Installing the Eclipse Plugin	24
3	DOWNLOAD AND DEPLOY SOURCE CODE.....	25
3.1	Download & Deploy Source Code (Web Infrastructure)	25
3.1.1	Configure and Build Source Code.....	25
3.1.2	Install Geoserver.....	26
3.2	Download & Deploy Source Code (MAST Mobile Application)	26
3.2.1	Download Source code.....	26
3.2.2	Configure and Build Source code.....	26

1 INTRODUCTION

This guide describes the setup of development environment of MAST-Burkin Faso web infrastructure and MAST Android mobile application customized for Burkina Faso on Windows environment. The objective of this document is to provide the detail of development environment setup, including the download and installation of various software and tools for end users so as to facilitate the recreation of MAST environment using the source code.

2 INSTALL AND CONFIGURE MAST DEVELOPMENT TOOLS

2.1 MAST Burkina Faso Web Infrastructure

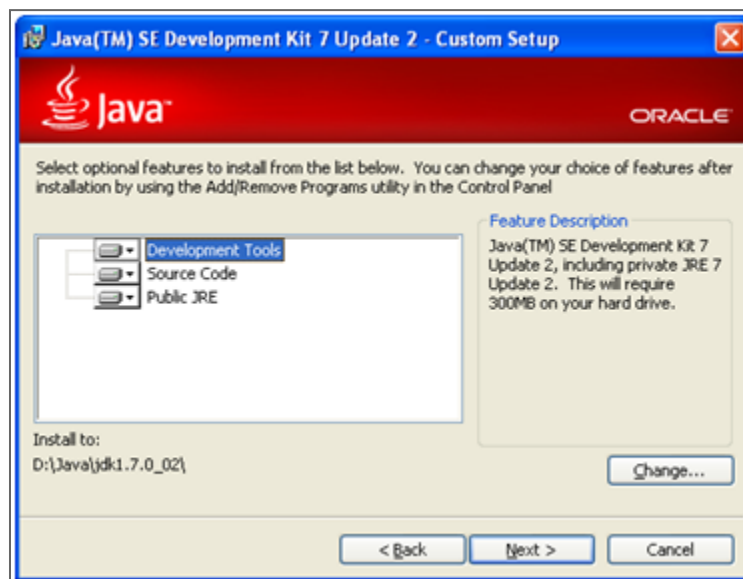
The core tools and software used to develop MAST web infrastructure are:

- Java JDK7
- PostgreSQL 9.3 /PostGIS 2.1
- Apache Tomcat 7
- Eclipse Luna IDE

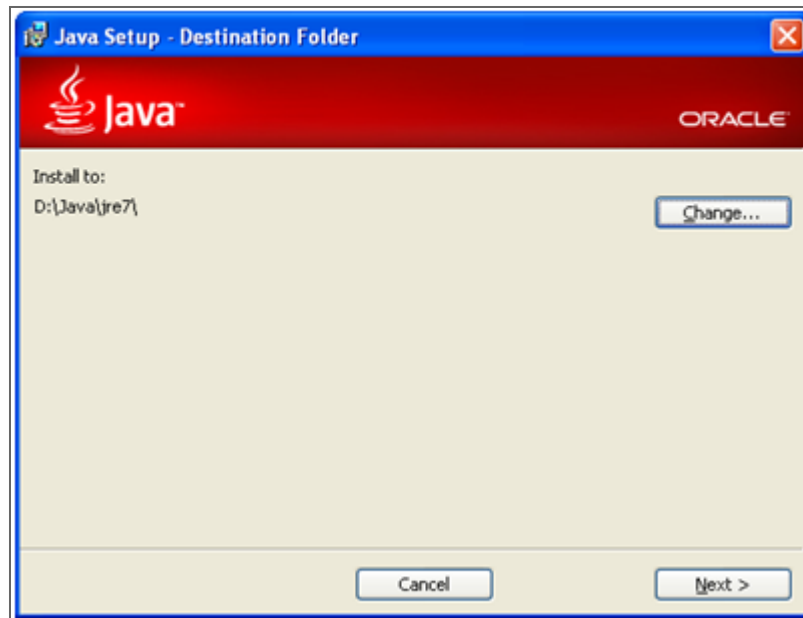
You also need to download source code for MAST web infrastructure from GitHub (<https://github.com/MASTUSAID-BurkinaFaso/MAST-BF-DM/>) and minimal postgresSQL scripts (<https://github.com/MASTUSAID-BurkinaFas/DB-SCRIPTS>) that are to be run on the database to schema and load mandatory master tables.

2.1.1 Installing Java JDK 7

- Download the Java 7 JDK update 56 or later from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Run the exe file and follow the instructions.



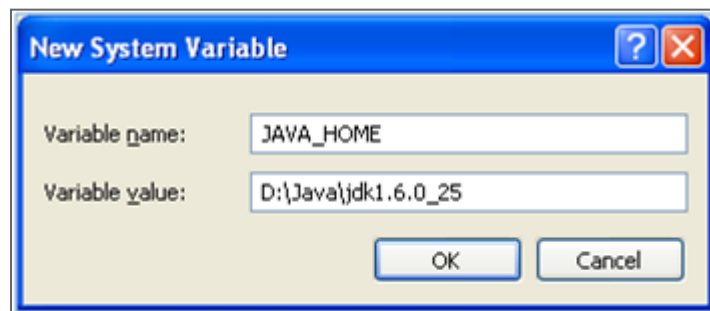
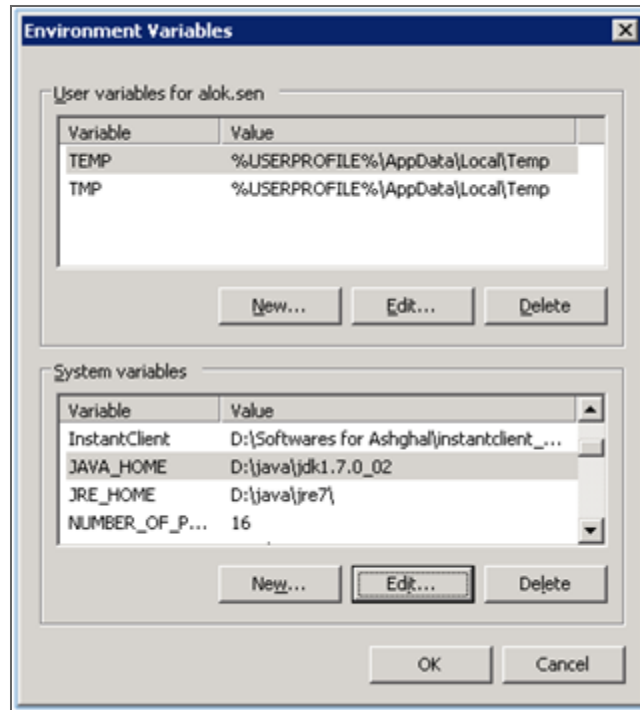
- Change the install path as required and click next.
- Click Next >



- Installation complete.

2.1.2 Setting Java and JRE Home

- From Control Panel open System Properties.
- Control Panel-> System Properties
- Click on Advanced Tab -> Environment Variables button.

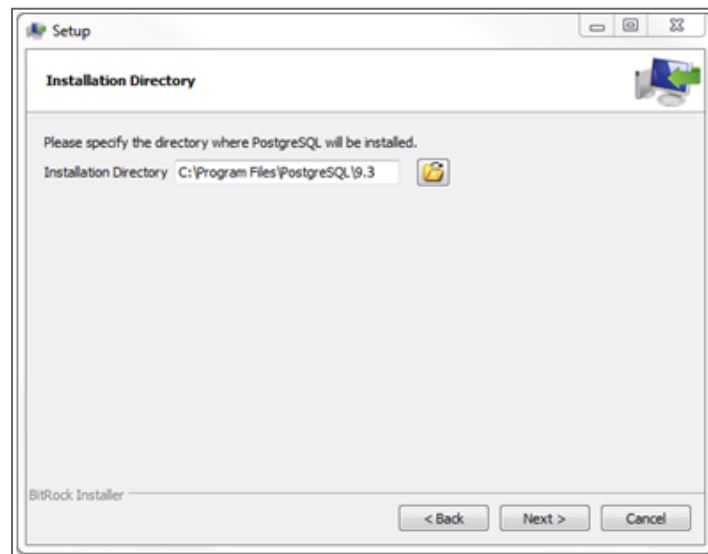


- In System variables click on new button.
- Create a new system variable JAVA_HOME specifying the JDK installation folder as variable value.
- In the same way create JRE_HOME variable with JRE installation folder (ex. D:\java\jre7\).

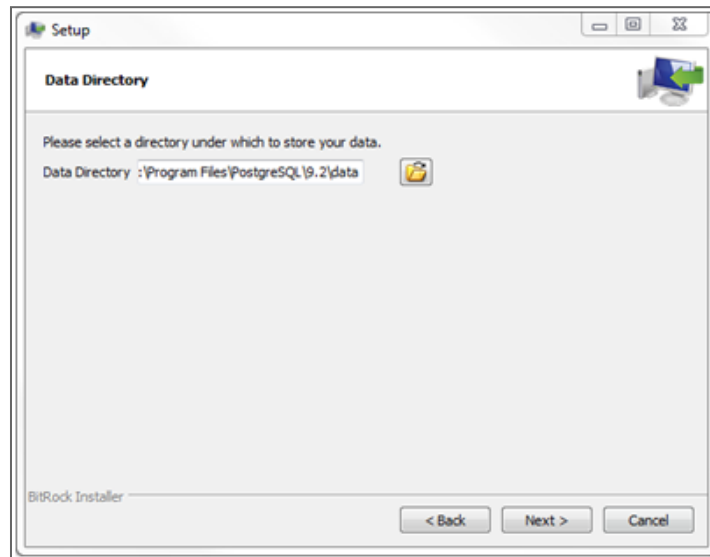
2.1.3 Installing PostgreSQL

The graphical installer for PostgreSQL includes the PostgreSQL server, pgAdmin III, a graphical tool for managing and developing databases, and StackBuilder, a package manager that is used to download and install PostGIS.

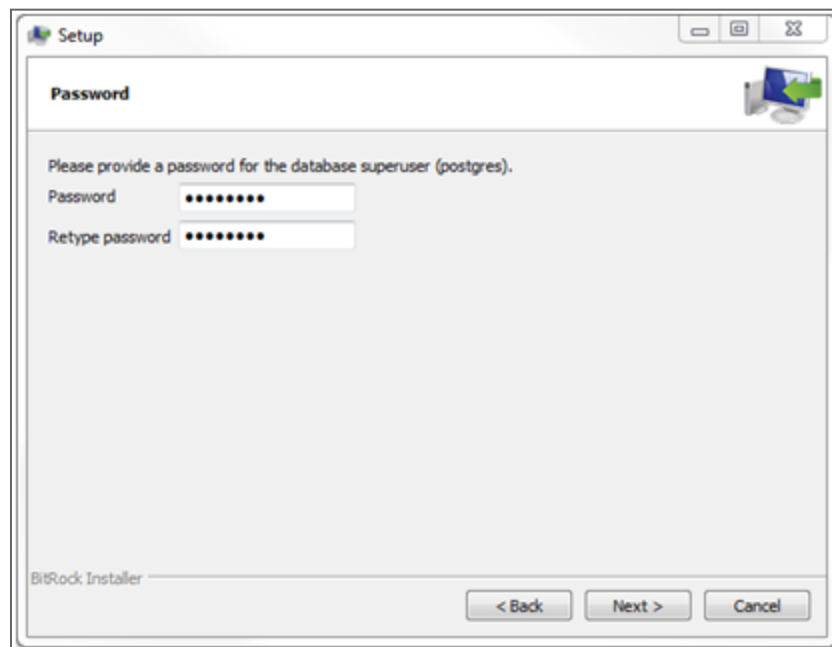
- Download PostgreSQL graphic installer from <http://www.enterprisedb.com/products-services-training/pgdownload#windows>
- Double click on the installer file.



- Specify installation directory.
- Specify the directory where you want to store the data in the database server.

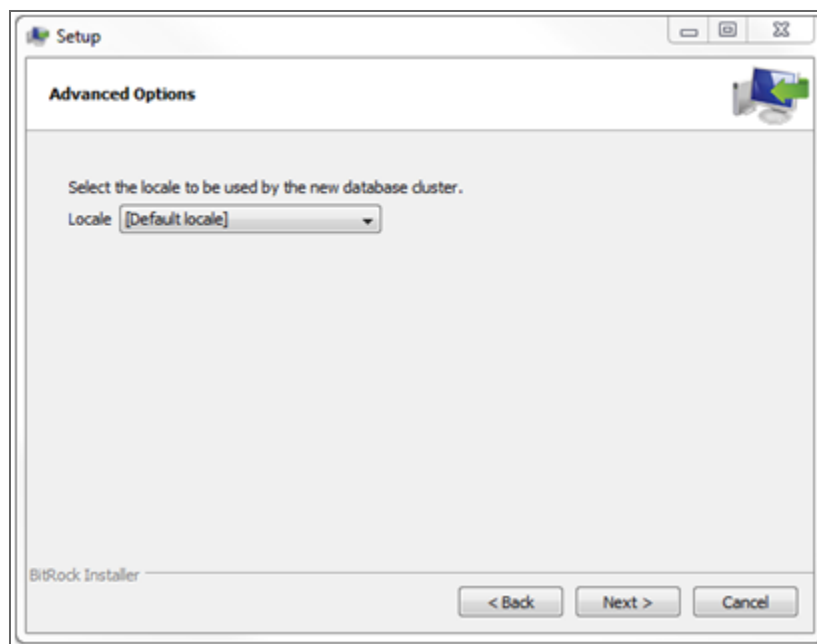
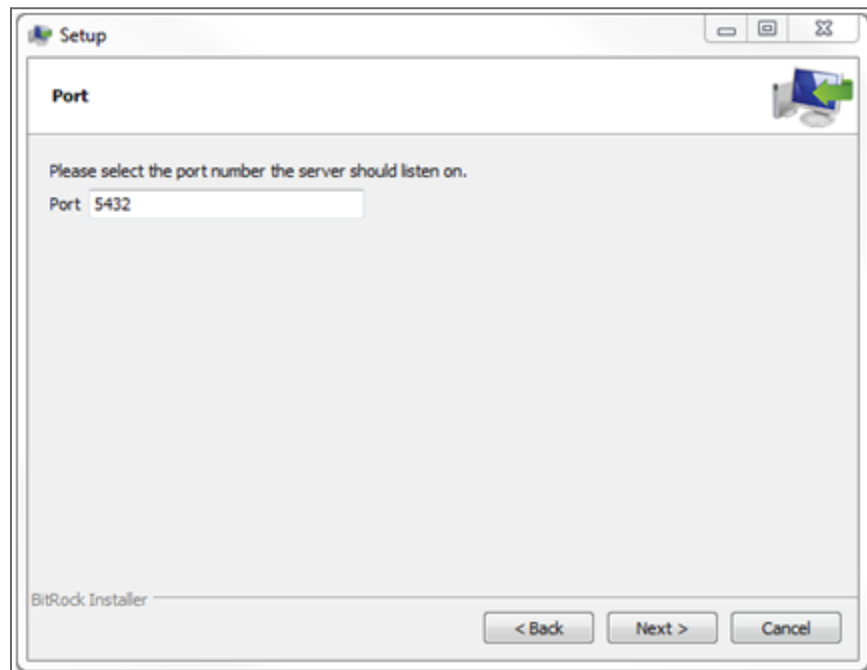


- Enter the password for the database superuser and service account. This password is



set for default Postgres user.

- Enter the port for PostgreSQL. Make sure that no other applications are using this port. Postgres by default uses port 5432.



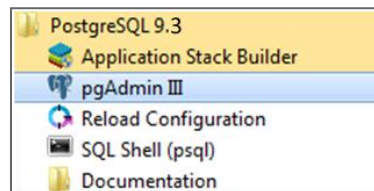
- Choose the default locale, this by default would select UTF-8 support for the database.

2.1.4 Verifying the Installation

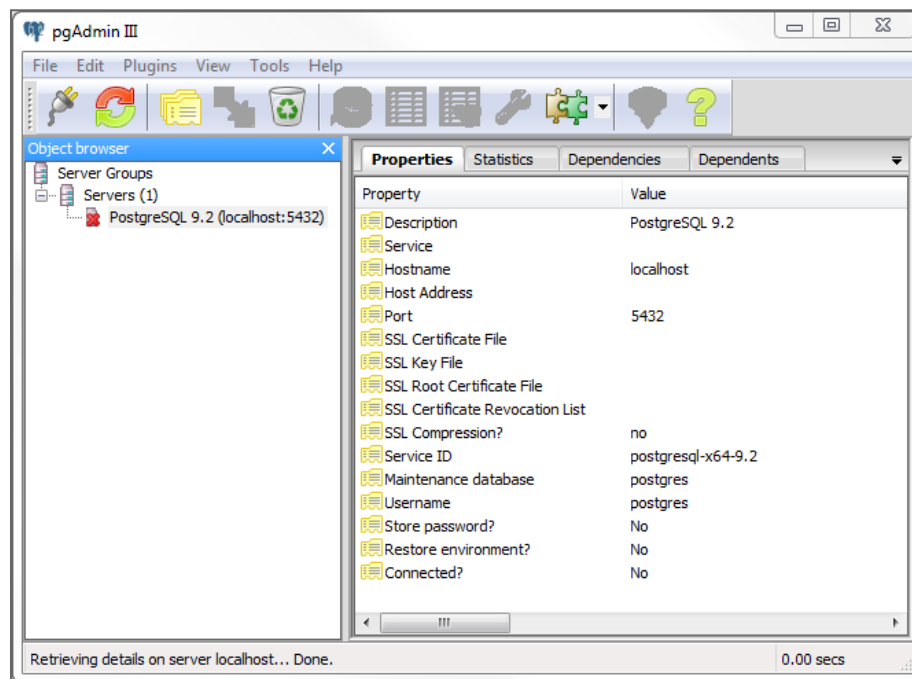
There are several ways to verify the installation. You can try to connect to the PostgreSQL database server from any client application e.g., psql, pgAdmin, etc.

Use pgAdmin III application installed with PostgreSQL database to check if database installed correctly and database service is running.

- Click on pgAdmin III to GUI.



- Double click on **PostgreSQL 9.3** on the object browser. It will ask you for admin password for Postgres user. Enter the password used in the installation step. Successful



login will display all the objects belonging to the server.

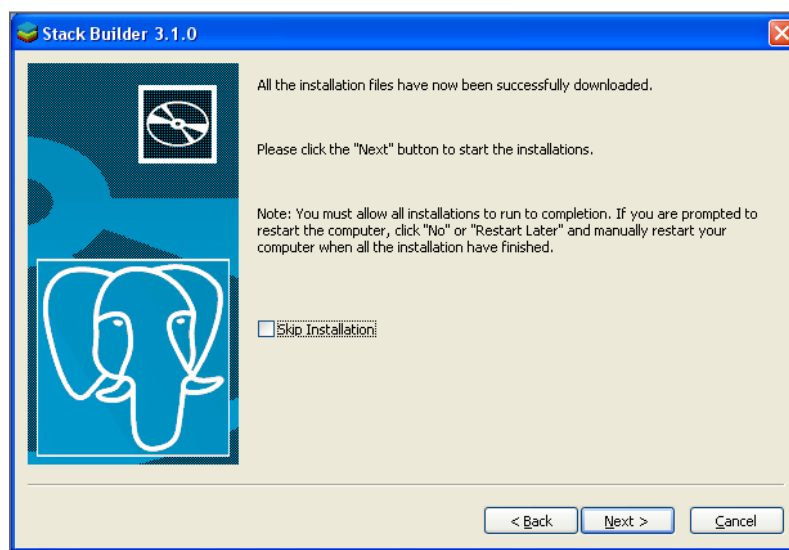
2.1.5 Installing PostGIS

After Postgres Installation is complete the installer will launch the Stack Builder application for installing Postgres extensions.

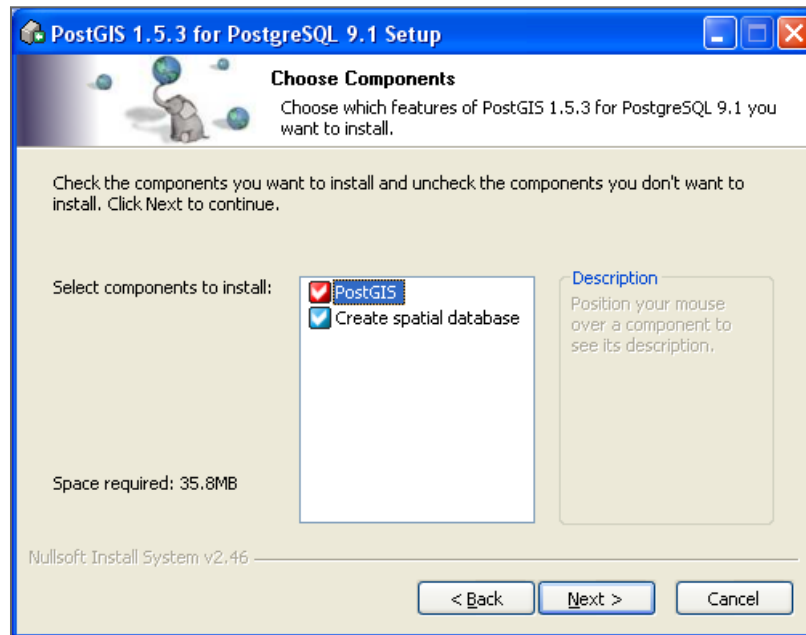
- Select the Postgres database that has been installed, from the dropdown.



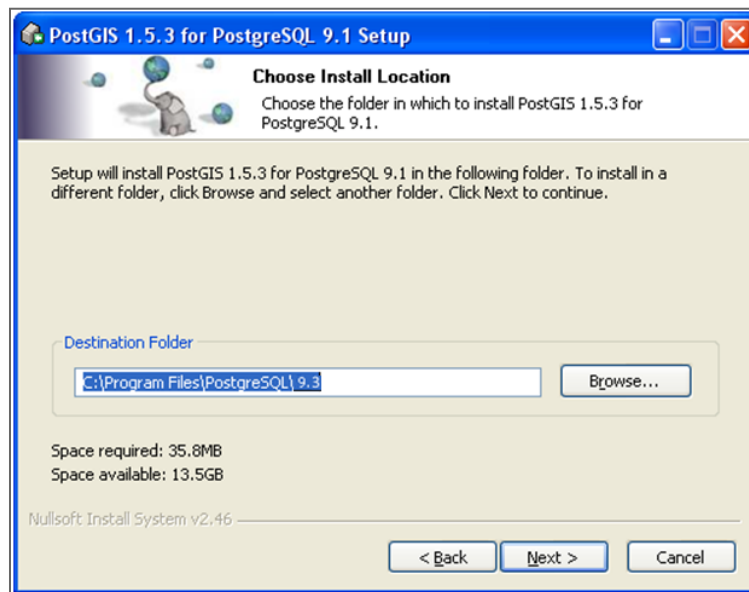
- Expand the spatial extension node and select the PostGIS option. Click Next.
- Click on the next few option to download the PostGIS extension.
- Click on Next button to start PostGIS installation.



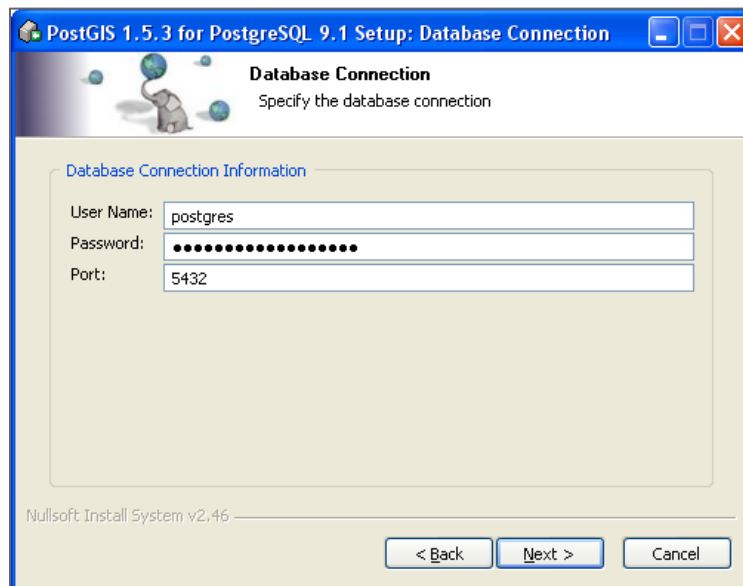
- Use default options to install PostGIS as well as create a Spatial Database.



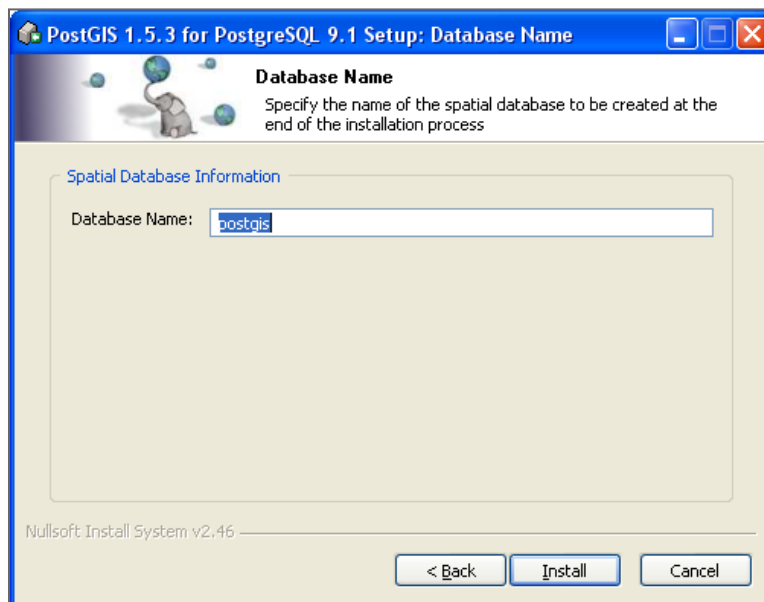
- Choose Location of PostGIS extension installation.



- Provide the credentials of the Postgresql database over which PostGIS is to be installed.

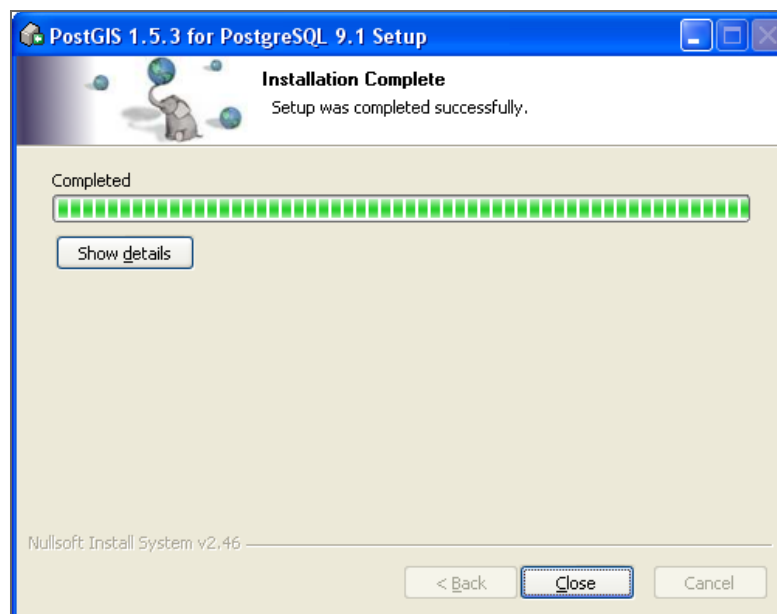
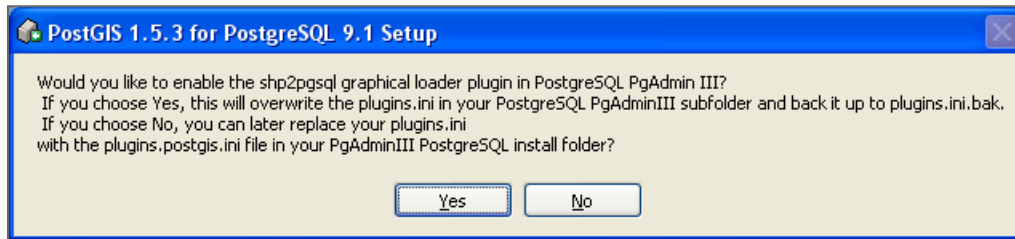


This would be the same as provided while installing Postgres database.



- Keep the spatial database Name as PostGIS (default).

- Click on Yes to install shapefile loader.

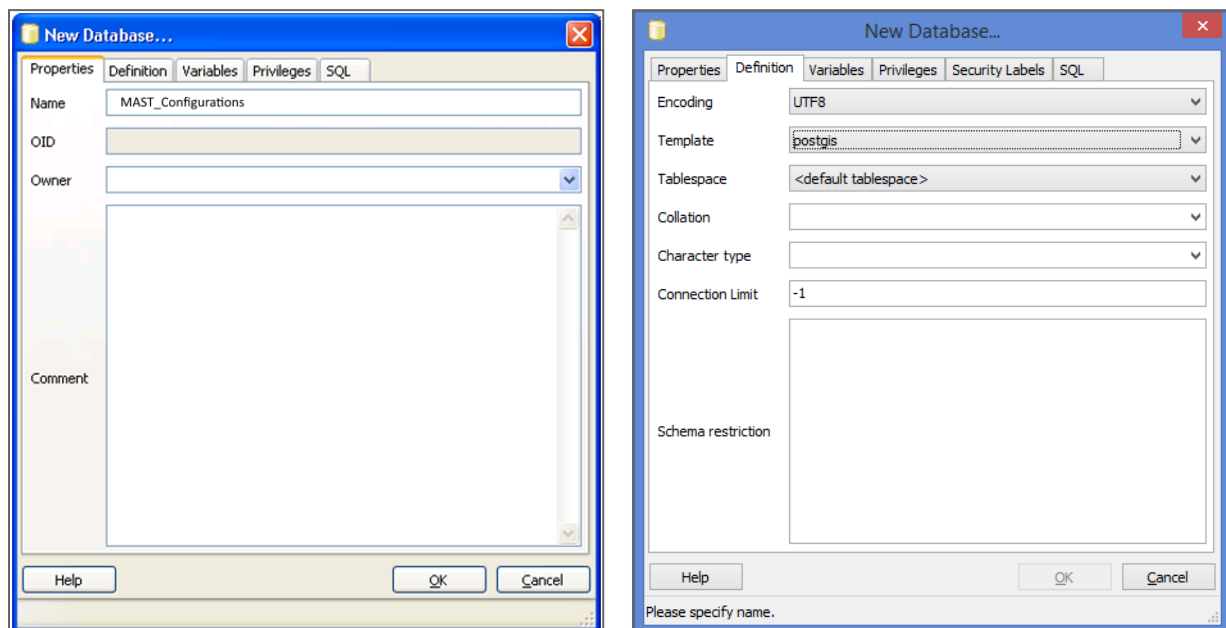


- PostGIS installation complete.

2.1.6 Creating and populating database from script

Below steps defines the steps involved in restoring the database backup file downloaded from GitHub.

- Download the postgresql database scripts from GitHub (<https://github.com/MASTUSAID-BurkinaFaso/DB-SCRIPTS>)
- Copy the database scripts file in the local system/server.
- Open pgAdminIII.
- Connect to Postgres database using the authentication specified while installing the database.
- Right click on Database node and Click on NEW DATABASE option. Provide a database name (MAST_Configurations) it should match with the database bak file that you are



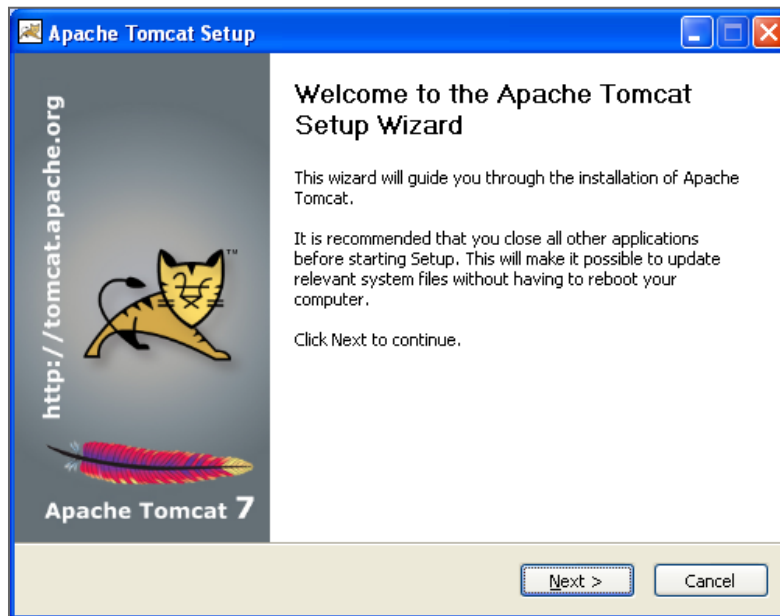
going to restore.

- Goto definition tab and select template as PostGIS from the dropdown. This would enable spatial support to the newly created database.
- Click OK. New database would be created.
- Run the script [mast_configurations_schema.sql](#) to create database schema.
- Once database schema is created run the script [mast_configurations_data.sql](#) to populate master tables.
- Validate whether all the tables have been created and master data populated in this new database.

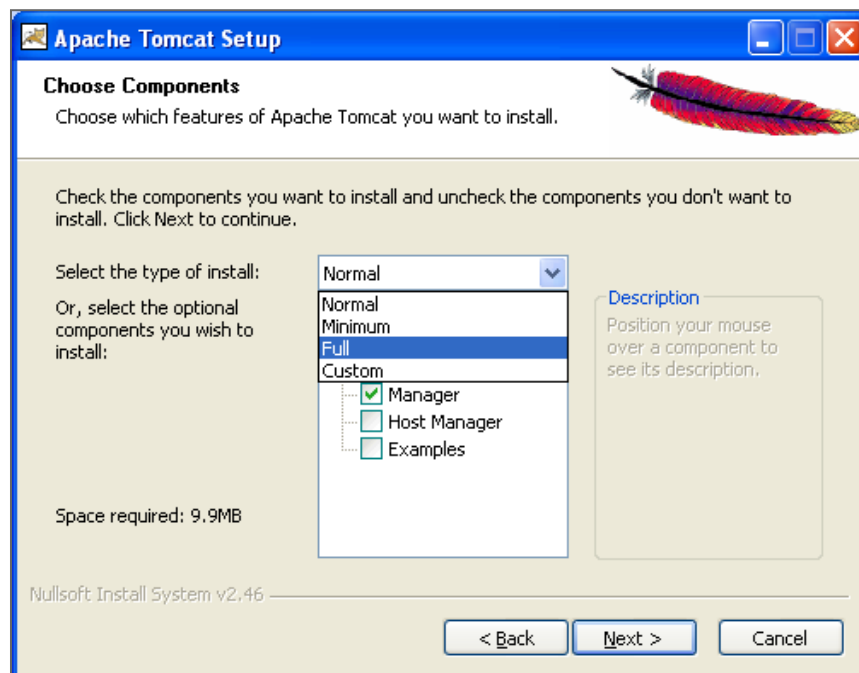
2.1.7 Apache Tomcat Installation

Download or use the latest version of Apache Tomcat Server installer executable file.

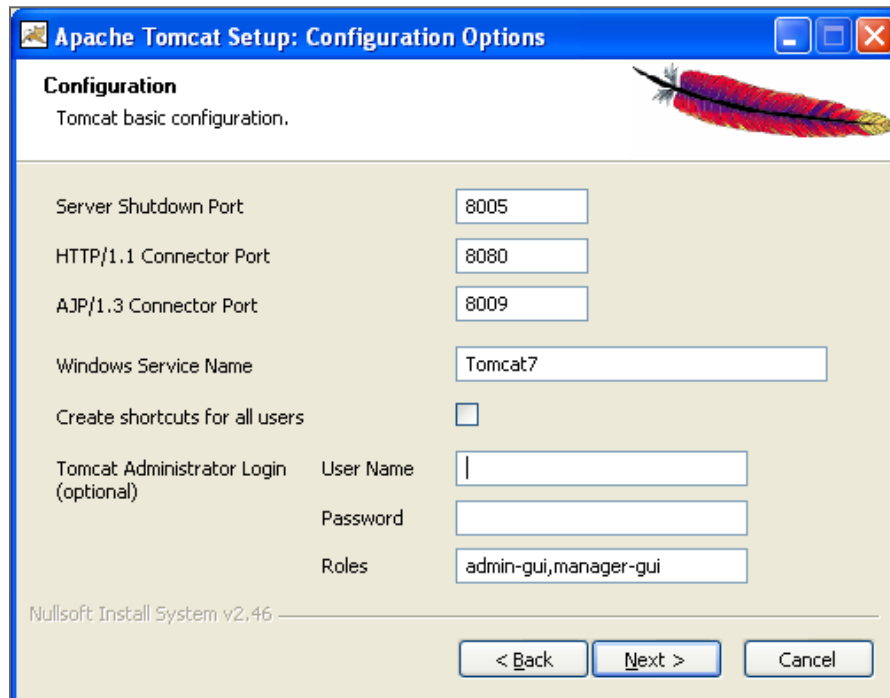
- Click Next.



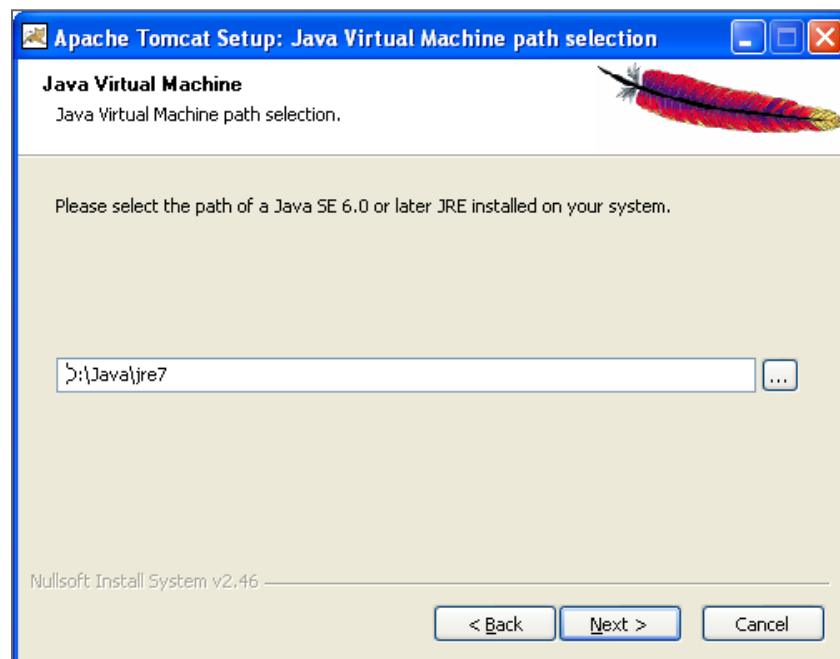
- Choose Type of Install as FULL to install all features.



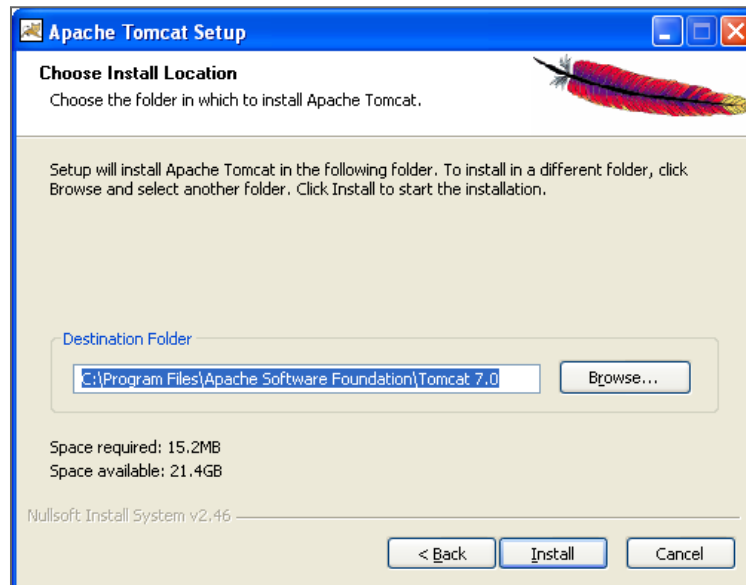
- In configuration option keep everything as default. Provide the user name and password and note it somewhere for future reference.



- Select the path of JRE as installed in the local system/server.



- Provide installation path of Tomcat and click on install. This would install the Tomcat



Server as a Service in the local system/ Server.

2.1.8 Apache Tomcat Configuration

Below steps define configuration of Apache Tomcat application server to be used with development environment to deploy the application in debug/run mode.

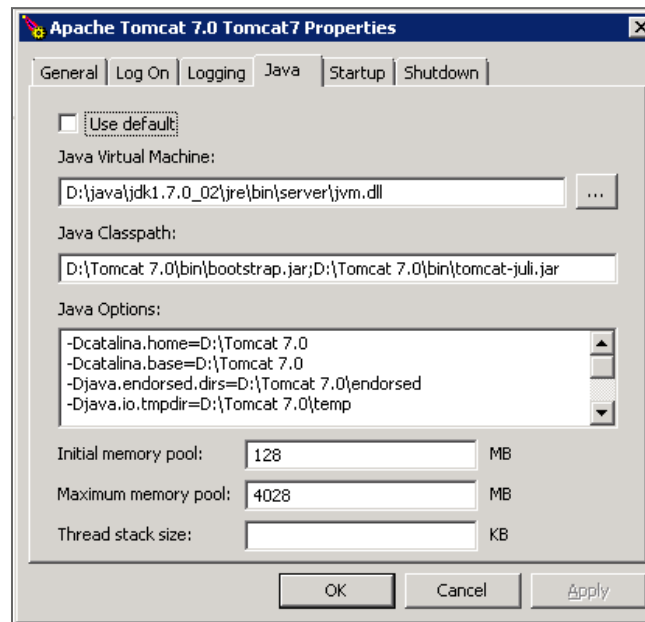
2.1.8.1 Memory Allocation

Go to Tomcat Home/BIN folder right click on Tomcat7w.exe file and click on Run as Administrator.

- Apache Tomcat Properties window will popup. Click on Java Tab.
- Add the following line at the end in Java Options Textbox.

“-XX:MaxPermSize=256m”

- Set the Initial memory pool to 128 and maximum memory pool to 4028.
- Click OK.



2.1.8.2 Configuring Context.xml and Server.xml

Context.xml : Add the line marked in RED font to context.xml file (Blue colored font describes the content of default context.xml file).

```
<?xml version='1.0' encoding='utf-8'?>
<!-- The contents of this file will be loaded for each web application -->
<Context>
  <!-- Default set of monitored resources -->
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
```

```

<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<!-- Uncomment this to enable Comet connection tacking (provides events
    on session expiration as well as webapp lifecycle) -->
<!-- <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
-->
<ResourceLink global="jdbc/dbname" name="jdbc/dbname"
type="javax.sql.DataSource"/>
</Context>

```

Server.xml: Edit the server.xml file. Add the lines as shown in red font in the server.xml file. Change the database/user and password of the postgres database to be configured with Spatialvue (as provided during postgres installation).

Please note that these changes are to be done only in the lines (shown in red font below) that are to be added in server.xml.

```

<?xml version='1.0' encoding='utf-8'?>
<Server port="8005" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
  <!-- Global JNDI resources
      Documentation at /docs/jndi-resources-howto.html
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
        UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
        type="org.apache.catalina.UserDatabase"
        description="User database that can be updated and saved"
        factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
        pathname="conf/tomcat-users.xml" />
    <Resource type="javax.sql.DataSource"
        name="jdbc/dbname"

```

```

        factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
        driverClassName="org.postgresql.Driver"
        url="jdbc:postgresql://host:5432/dbname"
        username="username"
        password="password"
            testWhileIdle="true"
        testOnBorrow="true"
        testOnReturn="false"
        validationQuery="SELECT 1"
        validationInterval="30000"
        timeBetweenEvictionRunsMillis="5000"
        maxActive="100"
        minIdle="10"
        maxWait="10000"
        initialSize="10"
        removeAbandonedTimeout="60"
        removeAbandoned="true"
        logAbandoned="true"
        minEvictableIdleTimeMillis="30000"
        jmxEnabled="true"
        jdbcInterceptors="org.apache.tomcat.jdbc.pool.interceptor.ConnectionState;
            org.apache.tomcat.jdbc.pool.interceptor.StatementFinalizer;

org.apache.tomcat.jdbc.pool.interceptor.SlowQueryReportJmx(threshold=10000)"
    />
</GlobalNamingResources>

<!-- A "Service" is a collection of one or more "Connectors" that share
a single "Container" Note: A "Service" is not itself a "Container",
so you may not define subcomponents such as "Valves" at this level.
Documentation at /docs/config/service.html
-->
<Service name="Catalina">

    <!--The connectors can use a shared executor, you can define one or more named thread
pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->

    <!-- A "Connector" represents an endpoint by which requests are received
and responses are returned. Documentation at :
    Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
    Java AJP Connector: /docs/config/ajp.html

```

APR (HTTP/AJP) Connector: /docs/apr.html
Define a non-SSL HTTP/1.1 Connector on port 8080

```
-->
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
    port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" />
-->
<!-- Define a SSL HTTP/1.1 Connector on port 8443
    This connector uses the JSSE configuration, when using APR, the
    connector should be using the OpenSSL style configuration
    described in the APR documentation -->
<!--
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
-->

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

<!-- An Engine represents the entry point (within Catalina) that processes
    every request. The Engine implementation for Tomcat stand alone
    analyzes the HTTP headers included with the request, and passes them
    on to the appropriate Host (virtual host).
    Documentation at /docs/config/engine.html -->

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Catalina" defaultHost="localhost" jvmRoute="jvm1">
-->
<Engine name="Catalina" defaultHost="localhost">

    <!--For clustering, please take a look at documentation at:
        /docs/cluster-howto.html (simple how to)
        /docs/config/cluster.html (reference documentation) -->
    <!--
    <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
-->

<!-- Use the LockOutRealm to prevent attempts to guess user passwords
```

```

        via a brute-force attack -->
<Realm className="org.apache.catalina.realm.LockOutRealm">
    <!-- This Realm uses the UserDatabase configured in the global JNDI
         resources under the key "UserDatabase". Any edits
         that are performed against this UserDatabase are immediately
         available for use by the Realm. -->
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
           resourceName="UserDatabase"/>
</Realm>
<Host name="localhost" appBase="webapps"
      unpackWARs="true" autoDeploy="true">
    <!-- SingleSignOn valve, share authentication between web applications
         Documentation at: /docs/config/valve.html -->
    <!--
    <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
    -->

    <!-- Access log processes all example.
         Documentation at: /docs/config/valve.html
         Note: The pattern used is equivalent to using pattern="common" -->
    <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
          prefix="localhost_access_log." suffix=".txt"
          pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>
</Engine>
</Service>
</Server>

```

2.1.9 Eclipse Luna

Eclipse Luna Integrated Development Environment is used as development platform for coding and compiling of MAST web infrastructure source code downloaded from GitHub.

- Download and install Eclipse Luna in local machine.
- Install Maven plugin dependency. Maven plugin can be installed from Eclipse market place.

2.2 MAST Mobile Application (Android)

The core tools and software used to develop MAST mobile application are:

- Java JDK7
- Eclipse Luna IDE
- Android Development Tools (ADT Plugin)
- Android Software Development Kit (SDK)

2.2.1 Installing Java JDK 7

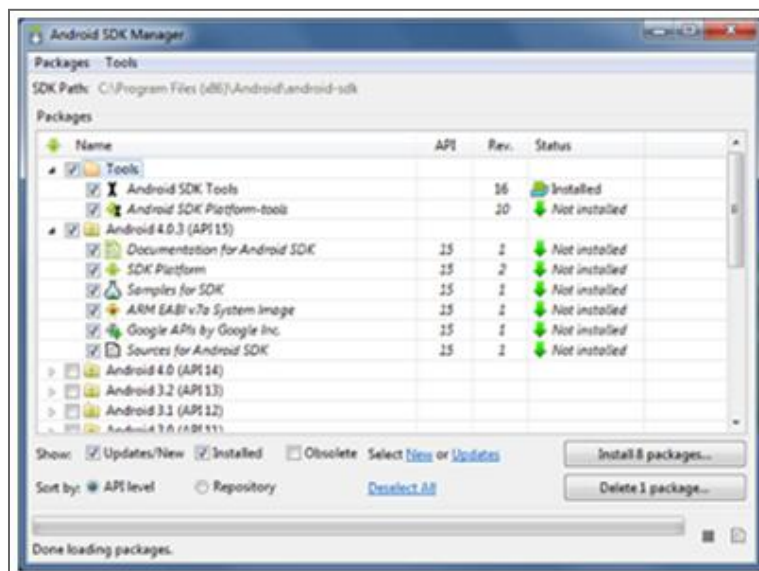
Installation method is same as defined in [section 2.1.1](#).

2.2.2 Installing the Stand-alone SDK Tools

Download the SDK from the following url: <https://developer.android.com/sdk/index.html#Other>

The downloaded package is an executable file that starts an installer. The installer checks your machine for required tools and installs it if necessary.

- Double-click the executable (.exe file) to start the install.
- Once the installation completes, the installer starts the Android SDK Manager.



As a minimum when setting up the Android SDK, you should download the latest tools and Android platform:

Open the Tools directory and select:

- Android SDK Tools
- Android SDK Platform-tools
- Android SDK Build-tools (highest version)

Open the first Android X.X folder (the latest version) and select:

- SDK Platform
- A system image for the emulator, such as ARM EABI v7a System Image.

For using Google Maps API download Google Play services package. Open the Extras directory and select:

- Google Repository
- Google Play services

Once you've selected all the desired packages, continue to install:

- Click Install X packages.
- In the next window, double-click each package name on the left to accept the license agreement for each.
- Click Install.

2.2.3 Installing the Eclipse Plugin

To add the ADT plugin to Eclipse:

- Start Eclipse, then select **Help > Install New Software**.
- Click **Add**, in the top-right corner.
- In the Add Repository dialog that appears, enter "ADT Plugin" for the *Name* and the following URL for the *Location*: <https://dl-ssl.google.com/android/eclipse/>

Note: The Android Developer Tools update site requires a secure connection. Make sure the update site URL you enter starts with HTTPS.

- Click **OK**.
- In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.
- In the next window, you'll see a list of the tools to be downloaded. Click **Next**.
- Read and accept the license agreements, then click **Finish**.

If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**.

- When the installation completes, restart Eclipse.

Once Eclipse restarts, you must specify the location of your Android SDK directory:

- In the "Welcome to Android Development" window that appears, select Use existing SDKs.
- Browse and select the location of the Android SDK directory you recently downloaded.
- Click Next.

Your Eclipse IDE is now set up to develop Android apps, but you need to add the latest SDK platform tools and an Android platform to your environment.

3 DOWNLOAD AND DEPLOY SOURCE CODE

3.1 Download & Deploy Source Code (Web Infrastructure)

Visit <https://github.com/MASTUSAID-BurkinaFaso/MAST-BF-DMI> to download MAST Data Management Infrastructure Source code. Download the zip file on to local system by clicking Download ZIP button.

3.1.1 Configure and Build Source Code

Prerequisite: For configuring the source code in Eclipse, it is mandatory to have working knowledge of Eclipse IDE and Maven. Without knowledge of these it will be difficult to set up the source code and configure jdk environment to work in debug environment.

Extract the downloaded zip file to local computer. Open Eclipse and create a new workspace in the folder where you extracted the source code.

On the Project Explorer/Package Explorer right click and select import→ import...

Expand Maven and select "Existing Maven Projects". Press next. Select the root folder of extracted MAST archive where pom.xml is located. Click finish.

Above activity would set up the maven project on to the new eclipse workspace that you created.

Set the Tomcat/webapp path for locally installed tomcat. This path can be set in pom.xml under artifact **maven-war-plugin**.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-war-plugin</artifactId>
  <version>2.1.1</version>
  <configuration>
    <warName>mast</warName>
    <outputDirectory>E:\apache-tomcat-7.0.56\webapps</outputDirectory>
    <packagingExcludes>WEB-INF/web.xml,WEB-INF/lib/org.apache.xerces-
2.9.0.jar,WEB-INF/lib/org.apache.xerces-2.9.0-SNAPSHOT.jar</packagingExcludes>
  </configuration>
</plugin>
```

From project menu click on clean option. Select project MAST and press OK. Ensure build automatically option is checked before running the clean.

Generating war file- For creating war file, on project explorer right click on the project root and select Run→Maven Install. This option would initially download all the maven dependencies as configured in pom.xml and then create war file in the tomcat/webapp path as configured in pom.xml.

3.1.2 Install Geoserver

Copy the geoserver.war to Tomcat/Webapps folder.

From Run Command type “Services.msc” to open services window. From the list of services select the **Apache Tomcat 7.0** service stop and restart the service.

When the service starts tomcat would automatically extract the war file in webapp folder.

3.2 Download & Deploy Source Code (MAST Mobile Application)

3.2.1 Download Source code

Visit <https://github.com/MASTUSAID-BurkinaFaso/MAST-MOBILE> to download MAST Mobile Application source code. Download the zip file on to local system by clicking Download ZIP button.

3.2.2 Configure and Build Source code

Prerequisite: For configuring the source code in Eclipse, it is mandatory to have working knowledge of Eclipse IDE and Android SDK. Without knowledge of these it would be difficult for the user to setup the source code and configure to work in debug environment.

Extract the downloaded zip file to local computer. Open Eclipse and create a new workspace in the folder where you extracted the source code.

On the Project Explorer/Package Explorer right click and select import → import. Choose Existing Android code into workspace option and browse to the project root folder and click Finish.

Above activity would set up the android project on to the new eclipse workspace that you just created.