Java Developers Guide

Table of Contents

Tools 1

Git 1

Eclipse 1

Creating workspaces and projects 1

Installing Eclipse 2

Installing Eclipse add-ons 2

Importing existing preferences 3

Manually configuring preferences 3

WildFly 5

Installing WildFly 8.2.0 on Mac OS X 5

Installing WildFly 8.2.0 on CentOS 5

Installing WildFly 8.2.0 on Windows 7

Setting up a PostgreSQL data source on WildFly 8 7

Enabling WildFly 8 on all network interfaces 8

Apache 10

Installing Apache on CentOS 10

Setting up Apache as a web server fronting JBoss 7 on CentOS 10

Enabling SSL support for Apache on CentOS 11

PostgreSQL 12

Using PostgreSQL with Mac OS X 12

Using PostgreSQL with CentOS 13

# Tools

A number of development tools are employed to produce the various Java development artifacts. These tools include the Intellij and Eclipse Java IDEs, the Wildfly JEE application server and the PostgreSQL relational database management system. Designs, documentation and source code artifacts are managed using the Git version control system.

## Git

“Git is a distributed revision control system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Every Git working directory is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server.” -- Wikipedia.

## Eclipse

“Eclipse is a multi-language software development environment comprising an IDE and a plug-in system to extend it. It is written primarily in Java and can be used to develop applications in Java and, by means of the various plug-ins, in other languages as well, including C, C++, COBOL, Python, Perl, PHP, and others. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C, Eclipse JDT for Java and Eclipse PDT for PHP.   
  
Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The runtime system of Eclipse is based on Equinox, an OSGi standard compliant implementation.

This plug-in mechanism is a lightweight software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX,[3] networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with Subversion support provided by third-party plug-ins.

The key to the seamless integration (but not of seamless interoperability) of tools with Eclipse is the plug-in. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are created equal. Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include UML plug-in for Sequence and other UML diagrams, plug-in for Database explorer, and many others.

The Eclipse SDK includes the Eclipse Java Development Tools, offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards. The Visual Editor project allows interfaces to be created interactively, thus allowing Eclipse to be used as a RAD tool.

Eclipse's widgets are implemented by a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.” -- Wikipedia

### Creating workspaces and projects

Related Eclipse projects are normally grouped under the same workspace e.g. all the Eclipse projects for a particular product will be placed under the workspace associated with that product. The workspaces and projects for Eclipse should be created in specific locations to ensure consistency across development workstations.

All workspaces should be created under the folder **C:\Workspaces** on Windows or on Unix and OS X under the folder **~/Workspaces**.

Eclipse projects are divided into two groups. The first group consists of those projects that are added to the Git version control repository. The second group consists of projects that are created for experimentation or testing purposes only. These projects are not added to Git.



Eclipse workspaces contain hard-coded paths that are relevant only to a particular development workstation. For this reason, they should not be added to the Git repository. Instead, a new workspace should always be created on each development machine and the Eclipse projects stored in the Git repository should be imported into the workspace.

### Installing Eclipse

Download and install the Eclipse IDE for Java EE Developers (Luna) from **https://eclipse.org**.

### Installing Eclipse add-ons

The following sections describe how to install the Eclipse IDE add-ons to support Java and JEE development.

**NOTE:** Before completing these steps you may need to correctly configure Eclipse to use your network proxy.

#### Install the JBoss Core Tools Eclipse Add-on

Complete the following steps to install the JBoss Core Tools Eclipse Add-on into the Eclipse IDE:

1. Start the Eclipse IDE.
2. Select **Help > Install New Software…** from the menu.
3. Click **Add…** to add a new update site.
4. Enter the following details for the new update site (Repository) and click **OK**:  
     
   **Name:** JBoss Core Tools  
   **Location:** http://download.jboss.org/jbosstools/updates/stable/luna/
5. Expand the **\* Abridged JBoss Tools** node in the tree and select the following items:  
     
   **JBossAS Tools**
6. Click **Next**.
7. Click **Next**.
8. Accept the license agreement and click **Finish** to start the installation.
9. Restart the Eclipse IDE when requested.

#### Install the Eclipse BPMN2 Modeler – Diagram Editor Eclipse Add-on

Complete the following steps to install the Eclipse BPMN2 Modeler – Diagram Editor Eclipse Add-on into the Eclipse IDE:

1. Start the Eclipse IDE.
2. Select **Help > Install New Software…** from the menu.
3. Click **Add…** to add a new update site.
4. Enter the following details for the new update site (Repository) and click **OK**:  
     
   **Name:** Eclipse BPMN2 Modeler  
   **Location:** http://download.eclipse.org/bpmn2-modeler/updates/luna/1.1.0/
5. Expand the **Eclipse BPMN2 Modeler** node in the tree and select the following items:  
     
   **BPMN2 Modeler – Diagram Editor**
6. Click **Next.**
7. Click **Next**.
8. Accept the license agreement and click **Finish** to start the installation.
9. Restart the Eclipse IDE when requested.

#### Optional: Install the Jindent Eclipse Add-on

Complete the following steps to install the Jindent Eclipse Add-on into the Eclipse IDE:

1. Start the Eclipse IDE.
2. Select **Help > Install New Software…** from the menu.
3. Click **Add…** to add a new update site.
4. Enter the following details for the new update site (Repository) and click **OK**:  
     
   **Name:** Jindent  
   **Location:** http://downloads.jindent.com/plugins/eclipse/3.x/
5. Expand the **Newforms – Software Development** node in the tree and select the following items:  
     
   **Jindent – Source Code Formatter**
6. Click **Next**.
7. Click **Next.**
8. Accept the license agreement and click **Finish** to start the installation.
9. Restart the Eclipse IDE when requested.

#### Optional: Install the Android Development Tools (ADT) Eclipse Add-on

Complete the following steps to install the Android Development Tools (ADT) Eclipse Add-on into the Eclipse IDE:

1. Start the Eclipse IDE.
2. Select **Help > Install New Software…** from the menu.
3. Click **Add…** to add a new update site.
4. Enter the following details for the new update site (Repository) and click **OK**:  
     
   **Name:** Android Development Tools (ADT)  
   **Location:** https://dl-ssl.google.com/android/eclipse/
5. Expand the **Developers Tools** node in the tree and select the following items:  
     
   **Android DDMS  
   Android Development Tools  
   Android Hierarchy Viewer  
   Android Traceview**
6. Click **Next**.
7. Accept the license agreement and click **Finish** to start the installation.
8. Restart the Eclipse IDE when requested.

### Importing existing preferences

Eclipse can be configured by importing an existing set of preferences using the **File > Import… > General > Preferences** option. These preferences must have been previously exported using the **File > Export… > General > Preferences** option.



Preferences are normally specific to a particular developer’s workstation since they include hard-coded paths. They are also configured for a particular workspace. This means that once you have manually configured the preferences, for a workspace on your developer workstation, you should export them as described above. The preferences can then be imported to configure a new workspace. You will normally **NOT** be able to use the preferences exported on another machine.

### Manually configuring preferences

Complete the following steps to configure Eclipse for Java development:

1. Switch to the **Java EE** perspective.
2. Select the **Window > Preferences…** or **Eclipse > Preferences…** menu option.
3. Select the **General > Editors > Text Editors** option.
   1. Set the **Displayed tab width** to **2**.
   2. Select **Insert spaces for tabs**.
   3. Select the **Show print margin** option and set the **Print margin column** to **100**.
4. Select the **Java > Code Style > Formatter** option and import the Java profile from Git:  
     
   mmp-java/resources/Eclipse Java Profile.xml
5. Select the **Maven > Errors/Warnings** option.
   1. Select **Ignore** for the **"groupId" duplicate of parent groupId** option.
   2. Select **Ignore** for the **"version" duplicate of parent version** option.
6. Select the **Web > CSS Files > Editor** option. If this option is not available then switch to the J2EE view and enable any required capabilities.
   1. Set the **Line width** to **100**.
   2. Select the **Indent using spaces** option and set the **Indentation size** to 2.
   3. Set the **Capitilization style** for **Identifier** to **Lowercase**.
7. Select the **Web > HTML Files > Editor** option.
   1. Set the **Line width** to **100**.
   2. Select the **Indent using spaces** option and set the **Indentation size** to 2.
8. Select the **Web > HTML Files > Validation** option.
   1. Expand the **Elements** section and set the **Unknown tag name** option to **Ignore**.
   2. Expand the **Elements** section and set the **Invalid tag location** option to **Ignore**.
   3. Expand the **Attributes** section and set the **Undefined attribute name** option to **Ignore**.
9. Select the **XML > XML Files > Editor** option.
   1. Set the **Line width** to **100**.
   2. Select the **Indent using spaces** option and set the **Indentation size** to 2.
10. Select the **XML > XML Files > Validation** option.
    1. Set the **Validating files > No grammar is specified** option to **Ignore**.
11. Click **OK**.
12. If desired, save your preferences using the **File > Export… > General > Preferences** option. These preferences can be imported when creating a new workspace using the **File > Import… > General > Preferences** option.

## WildFly

WildFly is an application server that implements the Java Platform, Enterprise Edition (Java EE).

WildFly is written in Java and as such is cross-platform: usable on any operating system that supports Java.

WildFly was developed by JBoss, now a division of Red Hat. Licensed under the terms of the GNU Lesser General Public License, WildFly is free and open source software.

### Installing WildFly 8.2.0 on Mac OS X

Complete the following steps to install and configure WildFly 8.2.0 on Mac OS X:

1. Launch the **Terminal** application and switch to the **root** user by executing the following command:  
     
   sudo su -
2. Execute the following commands as the **root** user to download the **wildfly-8.2.0.Final.tar.gz** file from **www.wildfly.org** and extract under **/opt**:  
     
   curl -o /tmp/wildfly-8.2.0.Final.tar.gz \  
    <http://download.jboss.org/wildfly/8.2.0.Final/wildfly-8.2.0.Final.tar.gz>  
     
   tar zxf /tmp/wildfly-8.2.0.Final.tar.gz -C /opt
3. Execute the following commands as the **root** user to grant all users write access to the WildFly deployment and to create the version independent symbolic link:  
     
   chmod -R a+rw /opt/wildfly-8.2.0.Final  
     
   ln -s /opt/wildfly-8.2.0.Final /opt/wildfly
4. Install the PostgreSQL JDBC driver module by completing the following steps:
   1. Clone the **mmp-java** project from the Bitbucket repository using the following URL:  
        
      https://marcusportmann@bitbucket.org/marcusportmann/mmp-java.git
   2. Extract the module **mmp-java/resources/wildfly-postgresql-module.tar.gz** under **/opt/wildfly/modules/system/layers/base**.

### Installing WildFly 8.2.0 on CentOS

Complete the following steps to install and configure WildFly 8.2.0 on CentOS:

1. Logon as the **root** user.
2. Execute the following commands as the **root** user to download the **wildfly-8.2.0.Final.tar.gz** file from **www.wildfly.org** and extract under **/opt**:  
     
   curl -o /tmp/wildfly-8.2.0.Final.tar.gz \  
    <http://download.jboss.org/wildfly/8.2.0.Final/wildfly-8.2.0.Final.tar.gz>  
     
   tar zxf /tmp/wildfly-8.2.0.Final.tar.gz -C /opt
3. Execute the following commands as the **root** user to grant all users write access to the WildFly deployment and to create the version independent symbolic link:  
     
   chmod -R a+rw /opt/wildfly-8.2.0.Final  
     
   ln -s /opt/wildfly-8.2.0.Final /opt/wildfly
4. Install the PostgreSQL JDBC driver module by completing the following steps:
   1. Clone the **mmp-java** project from the Bitbucket repository using the following URL:  
        
      https://marcusportmann@bitbucket.org/marcusportmann/mmp-java.git
   2. Extract the module **mmp-java/resources/wildfly-postgresql-module.tar.gz** under **/opt/wildfly/modules/system/layers/base**.
5. Execute the following commands as the **root** user to create the WildFly user:  
     
   mkdir /home/wildfly  
     
   groupadd --gid 201 wildfly  
     
   useradd --uid 201 --gid 201 -s /bin/bash -d /home/wildfly wildfly  
     
   chown -R wildfly /home/wildfly  
     
   chgrp -R wildfly /home/wildfly
6. Create the new file **/etc/profile.d/wildfly.sh** with the following contents:  
     
   export JBOSS\_HOME=/opt/wildfly  
   export WILDFLY\_HOME=/opt/wildfly  
   export PATH=$WILDFLY\_HOME/bin:$PATH  
   export MODULEPATH=
7. Create the new file **/etc/init.d/wildfly** with the following contents:  
     
   #!/bin/sh  
   ### BEGIN INIT INFO  
   # Provides: wildfly  
   # Required-Start: $local\_fs $remote\_fs $network $syslog  
   # Required-Stop: $local\_fs $remote\_fs $network $syslog  
   # Default-Start: 2 3 4 5  
   # Default-Stop: 0 1 6  
   # Short-Description: Start/Stop WildFly AS 8  
   ### END INIT INFO  
   #  
   # Source some script files in order to set and export environmental variables as well as add the appropriate executables to $PATH  
   [ -r /etc/profile.d/java.sh ] && . /etc/profile.d/java.sh  
   [ -r /etc/profile.d/wildfly.sh ] && . /etc/profile.d/wildfly.sh  
     
   start(){  
   echo "Starting WildFly 8"  
   su -l wildfly -c "sh ${WILDFLY\_HOME}/bin/standalone.sh -b 0.0.0.0 >/dev/null 2>/dev/null &"  
   }  
     
   stop(){  
   echo "Stopping WildFly 8"  
   su -l wildfly -c "sh ${WILDFLY\_HOME}/bin/jboss-cli.sh --connect --command=:shutdown"  
   }  
     
   restart(){  
   stop  
   # give stuff some time to stop before we restart  
   sleep 60  
   # protect against any services that can't stop before we restart  
   su -l wildfly -c "killall java"  
   start  
   }  
     
   case "$1" in  
   start)  
   start  
   ;;  
   stop)  
   stop  
   ;;  
   restart)  
   restart  
   ;;  
   \*)  
   echo "Usage: /etc/init.d/wildfly {start|stop|restart}"  
   exit 1  
   ;;  
   esac  
     
   exit 0
8. Execute the following command to set the permissions for the WildFly files:  
     
   chown -R wildfly.wildfly /opt/wildfly-8.2.0.Final   
     
   chmod a+x /etc/init.d/wildfly
9. Optional: Execute the following commands to permit access to WildFly through the CentOS firewall:  
     
   For CentOS 6:  
     
   iptables -I INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT  
     
   service iptables save  
     
   service iptables restart  
     
   ip6tables -I INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT  
     
   service ip6tables save  
     
   service ip6tables restart  
     
   For CentOS 7:  
     
   firewall-cmd --zone=public --add-port=8080/tcp --permanent  
     
   firewall-cmd --reload
10. Optional: Execute the following commands to grant AJP access to a specific host (This is required if using another server as an HTTP server):  
      
    NOTE: Replace XXX.XXX.XXX.XXX with the IP address of the HTTP server.  
      
    For CentOS 6:  
      
    iptables -I INPUT --source XXX.XXX.XXX.XXX -m state --state NEW -m tcp -p tcp --dport 8009 -j ACCEPT  
      
    service iptables save  
      
    service iptables restart  
      
    ip6tables -I INPUT --source XXX.XXX.XXX.XXX -m state --state NEW -m tcp -p tcp --dport 8009 -j ACCEPT  
      
    service ip6tables save  
      
    service ip6tables restart  
      
    For CentOS 7:  
      
    firewall-cmd \  
     --add-rich-rule='rule family="ipv4" source address="XXX.XXX.XXX.XXX" port port="8009" protocol="tcp" accept'  
      
    firewall-cmd --reload
11. OPTIONAL: Complete the following steps to setup the WildFly admin user:
    1. Start the WildFly application server.
    2. Execute the ***<WILDFLY\_HOME>*/bin/add-user.sh** script.
    3. Select option **(a)** to add a **Management User**.
    4. Enter the name of the new admin user e.g. admin.
    5. Enter the password for the new admin user e.g. admin.
    6. Complete the steps to add the new admin user.

### Installing WildFly 8.2.0 on Windows

Complete the following steps to install and configure WildFly 8.2.0 on Windows:

1. Download the latest version of the Java EE 7 Full & Web Distribution for WildFly 8 from **wildfly.org**.
2. Extract the WildFly package under C:\Java\JEE\wildfly-8.2.0.Final.
3. Install the PostgreSQL JDBC driver module by completing the following steps:
4. Clone the **mmp-java** project from the Bitbucket repository using the following URL:  
     
   <https://marcusportmann@bitbucket.org/marcusportmann/mmp-java.git>
5. Extract the module **mmp-java/resources/wildfly-postgresql-module.tar.gz** under **/opt/wildfly/modules/system/layers/base.**

### Setting up a PostgreSQL data source on WildFly 8

Complete the following steps to setup a PostgreSQL Data Source on WildFly 8:

1. Clone the **mmp-java** project from the Bitbucket repository using the following URL:  
     
   https://marcusportmann@bitbucket.org/marcusportmann/mmp-java.git
2. Create the following folder structure under the WildFly distribution folder:  
     
   ***<WILDFLY\_HOME>*/modules/system/layers/base/org/postgresql/main**
3. Copy the following JAR from the **mmp-java mmp-java/resources** folder to the ***<WILDFY\_HOME>*/modules/system/layers/base/org/postgresql/main** folder:  
     
   **postgresql-9.3-1101.jdbc41.jar**
4. Create a new file named **module.xml** under the **main** folder with the following contents:  
     
   <?xml version="1.0" encoding="UTF-8"?>  
   <module xmlns="urn:jboss:module:1.0" name="org.postgresql">  
    <resources>  
    <resource-root path="postgresql-9.3-1101.jdbc41.jar"/>  
    </resources>  
    <dependencies><module name="javax.api"/></dependencies>  
   </module>
5. Add an entry for the data source to the ***<WILDFLY\_HOME>*/standalone/configuration/standalone.xml** file under the **<datasources>** node:  
     
   <datasources>  
    ...  
    <datasource jndi-name="java:jboss/datasources/SampleDS" pool-name="SampleDS" enabled="true" jta="true"   
    use-java-context="true" use-ccm="true">  
    <connection-url>jdbc:postgresql://localhost:5432/sampledb</connection-url>  
    <driver-class>org.postgresql.Driver</driver-class>  
    <driver>postgresql-jdbc4</driver>  
    <pool>  
    <min-pool-size>2</min-pool-size>  
    <max-pool-size>5</max-pool-size>  
    <prefill>true</prefill>  
    <use-strict-min>false</use-strict-min>  
    <flush-strategy>FailingConnectionOnly</flush-strategy>  
    </pool>  
    <security>  
    <user-name>sampledb</user-name>  
    <password>Password1</password>  
    </security>  
    <validation>  
    <check-valid-connection-sql>SELECT 1</check-valid-connection-sql>  
    <validate-on-match>false</validate-on-match>  
    <background-validation>false</background-validation>  
    <use-fast-fail>false</use-fast-fail>  
    </validation>  
    </datasource>  
      
   </datasources>
6. Add an entry for the data source to the ***<WILDFLY\_HOME>*/standalone/configuration/standalone.xml** file under the **<datasources>** node:  
     
   <datasources>  
    ...  
    <drivers>  
    <driver name="postgresql-jdbc4" module="org.postgresql"/>  
    ...  
    </drivers>  
   </datasources>

### Enabling WildFly 8 on all network interfaces

Complete the following steps to setup a WildFly 8 server instance to listen on all network interfaces:

1. Add an entry for the **global** interface to the ***<WILDFLY\_HOME>*/standalone/configuration/standalone.xml** file under the **<interfaces>** node:  
     
   <interfaces>  
    ...  
    <interface name="global">  
    <!-- Use the wildcard address -->  
    <any-address/>  
    </interface>  
    ...  
   </interfaces>
2. If required modify the existing **management** interface in the ***<WILDFLY\_HOME>*/standalone/configuration/standalone.xml** file under the **<interfaces>** node:  
     
   <interfaces>  
    ...  
    <interface name="management">  
    <!-- Use the wildcard address -->  
    <any-address/>  
    </interface>  
    ...  
   </interfaces>
3. Edit the **<socket-binding>** node to use the **global** interface as the default interface:  
     
   <socket-binding-group name="standard-sockets" default-interface="**global**">

## Apache

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of the Apache project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

### Installing Apache on CentOS

Complete the following steps to install Apache on CentOS:

1. Open a terminal and switch to the **root** user.
2. Execute the following commands to download and install Apache from the **yum repositories**.  
     
   yum install httpd mod\_ssl  
     
   chkconfig httpd on  
     
   service httpd start

### Setting up Apache as a web server fronting JBoss 7 on CentOS

Complete the following steps to setup Apache as a web server fronting JBoss 7 on CentOS:

1. Open a terminal and switch to the **root** user.
2. Edit the ***<JBOSS\_HOME>*/standalone/configuration/standalone.xml** file and put the following **<system-properties>** element inside the **<server>** element after the **<extensions>** element.  
     
   <system-properties>  
    <property name="jvmRoute" value="centos"/>  
   </system-properties>
3. Edit the ***<JBOSS\_HOME>*/standalone/configuration/standalone.xml** file and put the following **<connector>** element inside the **<subsystem>** element after the http **<connector>** element:  
     
   <subsystem xmlns="urn:jboss:domain:web:1.0" default-virtual-server="default-host">  
    <connector name="http" protocol="HTTP/1.1" socket-binding="http" scheme="http"/>  
    <connector name="ajp" protocol="AJP/1.3" socket-binding="ajp" scheme="http"/>  
    …
4. Edit the ***<JBOSS\_HOME>*/standalone/configuration/standalone.xml** file and put the following **<socket-binding>** element inside the **<socket-binding-group>** element after the http **<socket-binding>** element:  
     
   <socket-binding-group name="standard-sockets" default-interface="public">  
    <socket-binding name="http" port="8080"/>  
    <socket-binding name="ajp" port="8009"/>  
    …
5. Create a file named **jboss.conf** under the directory **/etc/httpd/conf.d** with the following contents, replacing the application path in red as appropriate:  
     
   <Proxy balancer://**sample**>  
    BalancerMember ajp://127.0.0.1:8009/**sample** route=centos  
    ProxySet stickysession=JSESSIONID  
   </Proxy>  
     
   ProxyPass /**sample** balancer://**sample**  
   ProxyPassReverse /**sample** balancer://**sample**
6. Execute the following commands to permit access to Apache through the CentOS firewall:  
     
   iptables -I INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT  
   iptables -I INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT  
   service iptables save  
   service iptables restart  
     
   ip6tables -I INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT  
   ip6tables -I INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT  
   service ip6tables save  
   service ip6tables restart
7. Execute the following command to restart Apache:  
     
   service httpd restart

### Enabling SSL support for Apache on CentOS

Complete the following steps to enable SSL support for Apache on CentOS:

## PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS) released under a BSD-style license. -- Wikipedia.

### Using PostgreSQL with Mac OS X

The following sections explain how to use the PostgreSQL object-relational database management system (ORDBMS) on Mac OS X.

#### Installing PostgreSQL using MacPorts

Complete the following steps to install PostgreSQL on Mac OS X using the MacPorts software repository:

1. Install the Xcode command line tools.
   1. Open a new Terminal.
   2. Execute the following command:  
        
      xcode-select --install
   3. When prompted to install the command line developer tools click **Install**.
   4. Agree to the Command Line Tools License Agreement by clicking **Agree**.
2. Install **MacPorts** and **PostgreSQL v9.3.x**.
   1. Download the "pkg" installer for the latest MacPorts release for your version of OS X from **macports.org**.
   2. Run the "pkg" installer and install MacPorts.
   3. Launch the **Terminal** application and execute the following commands:  
        
      sudo /opt/local/bin/port selfupdate   
        
      sudo /opt/local/bin/port install postgresql93-server  
        
      sudo mkdir -p /opt/local/var/db/postgresql93/defaultdb  
        
      sudo chown -R postgres:postgres /opt/local/var/db/postgresql93  
        
      sudo su postgres -c '/opt/local/lib/postgresql93/bin/initdb \  
       -D /opt/local/var/db/postgresql93/defaultdb'
   4. Create the file **/opt/local/var/db/postgresql93/.profile** with the following contents:  
        
      export PATH=/opt/local/lib/postgresql93/bin:$PATH
   5. Add the **/opt/local/lib/postgresql93/bin** directory to your OS X user's **PATH** variable in the **~/.profile** file e.g. /Users/marcus/.profile.

#### Creating a PostgreSQL Database

Complete the following steps to create and initialize a PostgreSQL Database on Mac OS X:

1. Execute the command below as the **root** user to create the database.  
     
   sudo su postgres -c 'createdb --template=template0 --encoding=UTF8 sampledb'
2. Execute the command below as the **root** user to create the database user.  
     
   sudo su postgres -c "psql –d sampledb –c \"create role sampledb with superuser login password 'Password1'\""
3. Execute the SQL statements to initialise the database.  
     
   sudo su postgres -c 'psql -d sampledb -f SamplePostgres.sql'

#### Deleting a PostgreSQL Database

Complete the following steps to delete a PostgreSQL Database on Mac OS X:

1. Execute the command below as the **root** user to clean-up unreferenced large objects on the database.  
     
   sudo su postgres -c 'vacuumlo sampledb'
2. Execute the command below as the **root** user to delete the database.  
     
   sudo su postgres -c 'dropdb sampledb'

#### Restarting the PostgreSQL Database Server

Complete the following steps to restart the PostgreSQL Database Server on Mac OS X:

1. Execute the commands below as the **root** user to restart the database.  
     
   sudo su postgres -c 'pg\_ctl -D /opt/local/var/db/postgresql93/defaultdb -l /opt/local/var/db/postgresql93/postgres.log stop'  
     
   sudo su postgres -c 'pg\_ctl -D /opt/local/var/db/postgresql93/defaultdb -l /opt/local/var/db/postgresql93/postgres.log start'  
     
   sudo su postgres -c 'tail –n500 /opt/local/var/db/postgresql93/postgres.log'

#### Configuring the PostgreSQL Database Server

Complete the following steps to configure the PostgreSQL Database Server on Mac OS X:

1. Open the PostgreSQL configuration file using the following command:  
     
   sudo vi /opt/local/var/db/postgresql93/defaultdb/postgresql.conf
2. Change the **max\_prepared\_transactions** value to 10.
3. Change the **max\_connections** value to 10.
4. Close and save the file.

### Using PostgreSQL with CentOS

The following sections explain how to use the PostgreSQL object-relational database management system (ORDBMS) on CentOS.

#### Installing PostgreSQL

Complete the following steps to install PostgreSQL on CentOS:

1. TODO: Complete this section.
2. Execute the following command to start PostgreSQL automatically as a service:  
     
   systemctl enable postgresql-9.3.service

#### Creating a PostgreSQL Database

Complete the following steps to create and initialize a PostgreSQL Database on CentOS:

1. Execute the command below as the **root** user to create the database.  
     
   sudo su postgres -c 'createdb --template=template0 --encoding=UTF8 sampledb'
2. Execute the command below as the **root** user to create the database user.  
     
   sudo su postgres -c "psql –d sampledb –c \"create role sampledb with superuser login password 'Password1'\""
3. Execute the SQL statements to initialise the database.  
     
   sudo su postgres -c 'psql -d sampledb -f SamplePostgres.sql'

#### Deleting a PostgreSQL Database

Complete the following steps to delete a PostgreSQL Database on CentOS:

1. Execute the command below as the **root** user to clean-up unreferenced large objects on the database.  
     
   sudo su postgres -c 'vacuumlo sampledb'
2. Execute the command below as the **root** user to delete the database.  
     
   sudo su postgres -c 'dropdb sampledb'

#### Restarting the PostgreSQL Database Server

Complete the following steps to restart the PostgreSQL Database Server on CentOS:

1. Execute the following commands as the **root** user to restart the database.  
     
   service postgresql-9.3 stop  
     
   service postgresql-9.3 start