

Installing MySQL

```
$> apt-get update
```

1. First we install MySQL 5.0 like this:

```
$> apt-cache search mysql
```

```
libmysqlclient15off - mysql database client library
mysql-common - mysql database common files (e.g. /etc/mysql/my.cnf)
libdbd-mysql-perl - A Perl5 database interface to the MySQL database
phpmyadmin - Administrate MySQL over the WWW
mysql-server-5.0 - mysql database server binaries
php5-mysql - MySQL module for php5
mysql-client-5.0 - mysql database client binaries
```

```
$> apt-get install mysql-server-5.0
```

2. Create a password for the MySQL user root (replace yourrootsqlpassword with the password you want to use):

```
mysqladmin -u root password yourrootsqlpassword
```

Then check with

```
netstat -tap | grep mysql
```

on which addresses MySQL is listening. If the output looks like this:

```
tcp    0    0 localhost.localdo:mysql *.*          LISTEN  2713/mysqld
```

which means MySQL is listening on localhost.localdomain only, then you're safe with the password you set before. But if the output looks like this:

```
tcp    0    0 *:mysql *.*          LISTEN  2713/mysqld
```

you should set a MySQL password for your hostname, too, because otherwise anybody can access your database and modify data:

```
mysqladmin -h server1.example.com -u root password yourrootsqlpassword
```

MySQL config file:

```
/etc/mysql/my.cnf
```

Changing the datadir directory

Here is the procedure how to change Mysql default data directory

By default, MySQL's datadir is placed in the /var/lib/mysql directory.

Create the directory that will be new datadir (ex:-/home/db)

chown the directory to the mysql:mysql user

```
sudo chown -R mysql:mysql /home/db/*
```

You need to stop the mysql server using the following command

```
sudo /etc/init.d/mysql stop
```

Now you need to edit the /etc/mysql/my.cnf file

```
sudo vi /etc/mysql/my.cnf
```

and look for "datadir = /var/lib/mysql" this is where mysql database default data directory here you need to change this one to your new directory

```
datadir = /home/db
```

copy the files from the old datadir to the new location. However, make sure that the files named

ib_arch_log_0000000000, ib_logfile0 etc. are not copied to the newer location.

Make sure that the files and directories are owned by mysql user

Make changes in the my.cnf to point the new datadir.

Restart the MySQL database

```
sudo /etc/init.d/mysql start
```

Installing Apache/PHP5/phpMyAdmin

Next we install **Apache2** together with **PHP5** and **phpMyAdmin**. phpMyAdmin is not required, but it's always good to have it in case you want a GUI for accessing your MySQL database:

```
apt-get install apache2 libapache2-mod-php5 php5 php5-mysql phpmyadmin
```

Apache2 Configuration under Debian GNU/Linux Files and Directories in /etc/apache2 (see Attachment)

phpMyAdmin Configuration etc/phpadmin/config.inc.php

```
<?php
/**
 * Debian local configuration file
 *
 * This file overrides the settings made by phpMyAdmin interactive setup
 * utility.
 *
 * For example configuration see /usr/share/doc/phpmyadmin/examples/config.default.php.gz
 *
 * NOTE: do not add security sensitive data to this file (like passwords)
 * unless you really know what you're doing. If you do, any user that can
 * run PHP or CGI on your webserver will be able to read them. If you still
 * want to do this, make sure to properly secure the access to this file
 * (also on the filesystem level).
 */

/**
 * Server(s) configuration
 */
$i = 0;

/* Server localhost (http) [1] */
$i++;
$config['Servers'][$i]['host'] = 'localhost';
$config['Servers'][$i]['extension'] = 'mysql';
$config['Servers'][$i]['connect_type'] = 'tcp';
$config['Servers'][$i]['compress'] = false;
$config['Servers'][$i]['auth_type'] = 'http';

/* End of servers configuration */

?>
```

Fatal error: Call to undefined function: mysql_pconnect()

Uncomment the following line in der php.ini: (/etc/php5/apache)

; Example lines:

```
extension=mysql.so  
;extension=gd.so
```

Restart apache

Installing servlet container Tomcat 5

Out of the box, Tomcat 5.5 requires the Java 2 Standard Edition Runtime Environment (JRE) version 5.0 or later.

(1) Download and Install the J2SE Runtime Environment (JRE)

(1.1) Download the Java 2 Standard Edition Runtime Environment (JRE), release version 5.0 or later, from <http://java.sun.com/j2se>.

(1.2) Install the JRE according to the instructions included with the release.

(1.3) Set an environment variable named JAVA_HOME to the pathname of the directory into which you installed the JRE, e.g. c:\j2sdk5.0 or /usr/local/java/j2sdk5.0.

(2) Download and Install the Tomcat Binary Distribution

NOTE: As an alternative to downloading a binary distribution, you can create your own from the Tomcat source repository, as described in "BUILDING.txt". If you do this, the value to use for "\${catalina.home}" will be the "dist" subdirectory of your source distribution.

(2.1) Download a binary distribution of Tomcat from:

<http://tomcat.apache.org>

(2.2) Unpack the binary distribution into a convenient location so that the distribution resides in its own directory (conventionally named "apache-tomcat-[version]"). For the purposes of the remainder of this document, the symbolic name "\$CATALINA_HOME" is used to refer to the full pathname of the release directory.

```
/usr/local> tar xvfz apache-tomcat-5.5.26.tar.gz
```

(3) Start Up Tomcat

- (1) The most common hiccup is when another web server (or any process for that matter) has laid claim to port 8080. This is the default HTTP port that Tomcat attempts to bind to at startup. To change this, open the file:

```
$CATALINA_HOME/conf/server.xml
```

and search for '8080'. Change it to a port that isn't in use, and is greater than 1024, as ports less than or equal to 1024 require superuser access to bind under UNIX.

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->  
<Connector port="8091" maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"  
maxSpareThreads="75" enableLookups="false" redirectPort="8443" acceptCount="100" connectionTimeout="20000"  
disableUploadTimeout="true"/>
```

Restart Tomcat and you're in business. Be sure that you replace the "8080" in the URL you're using to access Tomcat. For example, if you change the port to 8091, you would request the URL `http://localhost: 8091/` in your browser.

- (3.1) Tomcat can be started by executing the following commands:

```
$CATALINA_HOME\bin\startup.bat    (Windows)
```

```
$CATALINA_HOME/bin/startup.sh    (Unix)
```

- (3.2) After startup, the default web applications included with Tomcat will be available by visiting:

```
http://localhost: 8091/
```

- (3.3) Further information about configuring and running Tomcat can be found in the documentation included here, as well as on the Tomcat web site:

```
http://tomcat.apache.org
```

(4) Shut Down Tomcat

- (4.1) Tomcat can be shut down by executing the following command:

```
$CATALINA_HOME\bin\shutdown    (Windows)
```

```
$CATALINA_HOME/bin/shutdown.sh    (Unix)
```

Installing jk Module (Apache Tomcat Connector)

```
$> apt-get install libapache-mod-jk
```

Configuration includes **workers.properties** and **uriworkermap.properties** files, that should be placed in **/etc/libapache2-mod-jk**
See Attachment.

Installing Apache Cocoon

Apache Cocoon is a web development framework built around the concepts of separation of concerns (making sure people can interact and collaborate on a project, without stepping on each other toes) and component-based web development.

Cocoon implements these concepts around the notion of "component pipelines", each component on the pipeline specializing on a particular operation. This makes it possible to use a "building block" approach for web solutions, hooking together components into pipelines without any required programming.

Cocoon is "*web glue for your web application development needs*". It is a glue that keeps concerns separate and allows parallel evolution of the two sides, improving development pace and reducing the chance of conflicts.

Cocoon has been designed to coexist and interoperate side-by-side with your existing J2EE solutions or to give them new functionality without requiring any change in the existing infrastructure.

Cocoon interacts with many data sources, including filesystems, RDBMS, LDAP, native XML databases, SAP® systems and network-based data sources. It adapts content delivery to the capabilities of different devices like HTML, WML, PDF, SVG, and RTF, to name just a few. You can run Cocoon as a Servlet as well as through a powerful, commandline interface. The deliberate design of its abstract environment gives you the freedom to extend its functionality to meet your special needs in a highly modular fashion.

Get the latest release 2.1.11 at: <http://cocoon.apache.org/>

1) Unpack the distribution

Obviously you've done this already, but if you got errors when unpacking the archive with tar, you might need to use gnutar instead. Our archives contain long paths and filenames which cause problems with some versions of the tar command.

2) Set your JAVA_HOME environment (if not already set)

You have to set your JAVA_HOME environment to point to the root directory of the Java Virtual Machine (JDK 1.3.x or later) installed on your machine.

To do this simply type:

```
[unix] JAVA_HOME=/path/to/java/  
[win32] SET JAVA_HOME=c:\path\to\java
```

your mileage may vary depending on your shell, but you know how to setup environments, right?

If you're using JDK 1.3.x, you have to exclude the JCR block before building

Cocoon. Have a look at the "Choosing the blocks" section below on how to excludue a block. Please note that some samples require JDK 1.4 or above.

- 3) Build Cocoon by typing "build" or "**./build.sh war**"
- 4) Run Cocoon by typing "cocoon" or "**./cocoon.sh**"
- 5) Open <http://localhost:8888/> with your browser
- 6) If you want Cocoon to run as an application in Tomcat container, just copy the cocoon.war file in the Tomcat webapps directory and restart Tomcat.

Open <http://localhost:8091/cocoon> with your browser

Choosing the blocks

Cocoon is composed by its 'core' and several 'blocks'.

The core (a.k.a. naked cocoon') contains the system with services that are needed to all cocoon users. Blocks provide services that you might not need, therefore the build system allows you to remove them from the build.

Now, do the following steps to configure the blocks you want in your cocoon:

- 1) cp blocks.properties local.blocks.properties
- 2) edit local.blocks.properties
- 3) rebuild (do a "build clean" first if you deactivated some blocks)

do not modify blocks.properties directly!

Main config files are:

- cocoon/**sitemap.xmap**
- WEB-INF/**web.xml**
- WEB_INF/**cocoon.xconf**