# PHP with SQLite on Digi Embedded Linux

SQLite 2.8.17 supported with sql_..() API

SQLite 3.x support via PDO

# Document History

| Date | Version | Change Description |
|---|---|---|
| 10/14/2009 | V1.0 | Initial entry/outline |
| 10/16/2009 | V1.1 | Switched back to compiling php-5.2.11 |
| 11/09/2009 | V1.2 | Integrated SQlite support, tested on CC9M2443 |
| 06/01/2010 | V1.3 | Retested with php-5.2.13 DEL5.2 toolchain and pdo-sqlite for sqlite3 support on CC9P9215 |

# Table of Contents



# 1 Problem Description

Digi embedded modules running Digi Embedded Linux are delivered with pre-compiled out of the box web servers with PHP capabilities. Also a pre-compiled small embedded data base SQLite is provided. This document describes how to cross compile a third party PHP interpreter and configure the web server(s) to execute it (e.g. with a test.php page). It also describes how to configure the firmware for running SQLite data base and an example to access it from PHP pages ran in the web server.

# 2 Requirements

To try the example in this document you need:

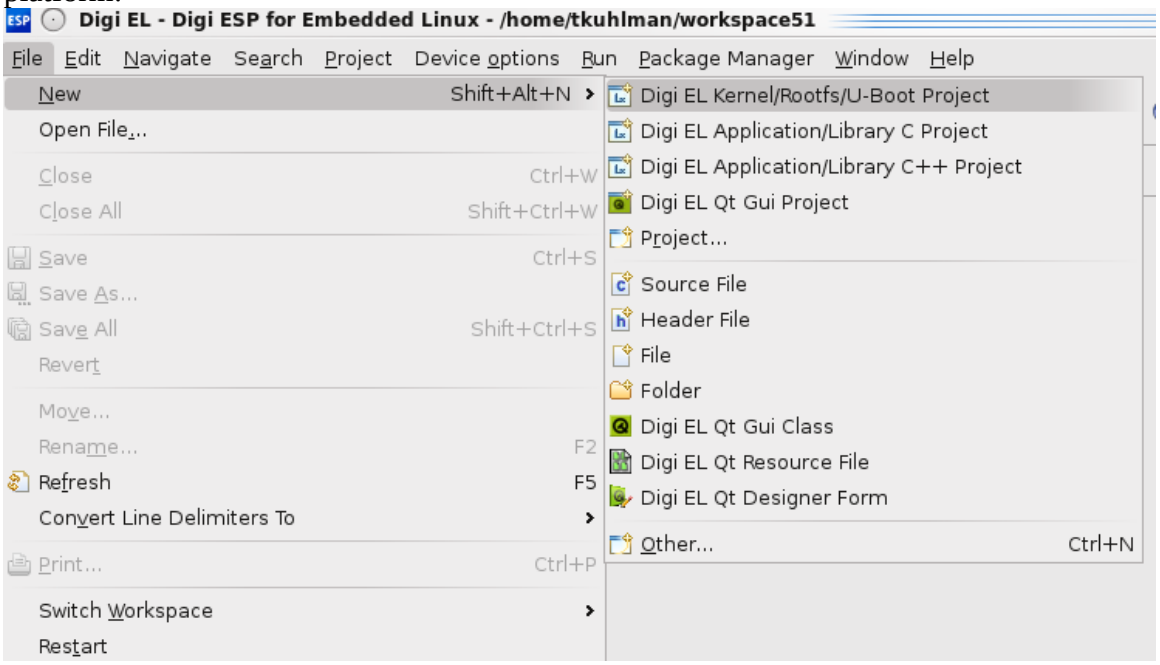
- Digi Connect ME 9210 or ConnectCore module mounted on Digi development board.
- Digi Embedded Linux (DEL) 5.1 or above development environment.
- You can get everything together in a Digi Linux JumpStart Kit

# 3 Software Setup

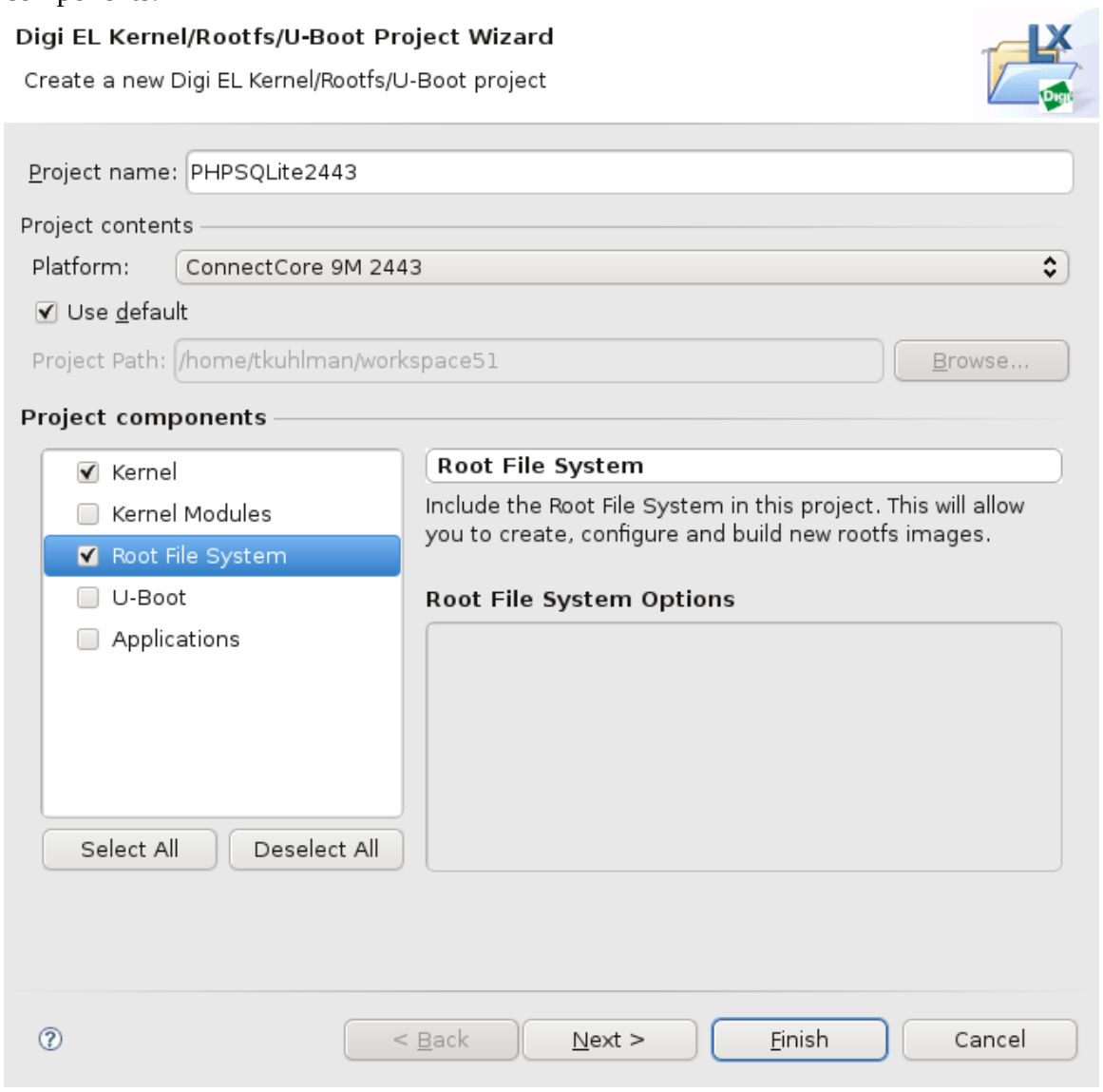
For your convenience find all files and images compiled into the archive this instructions came with, you might want to skip this section.

- Install Digi Embedded Linux (DEL) 5.1 or higher, apply latest patches with the Package Manager.

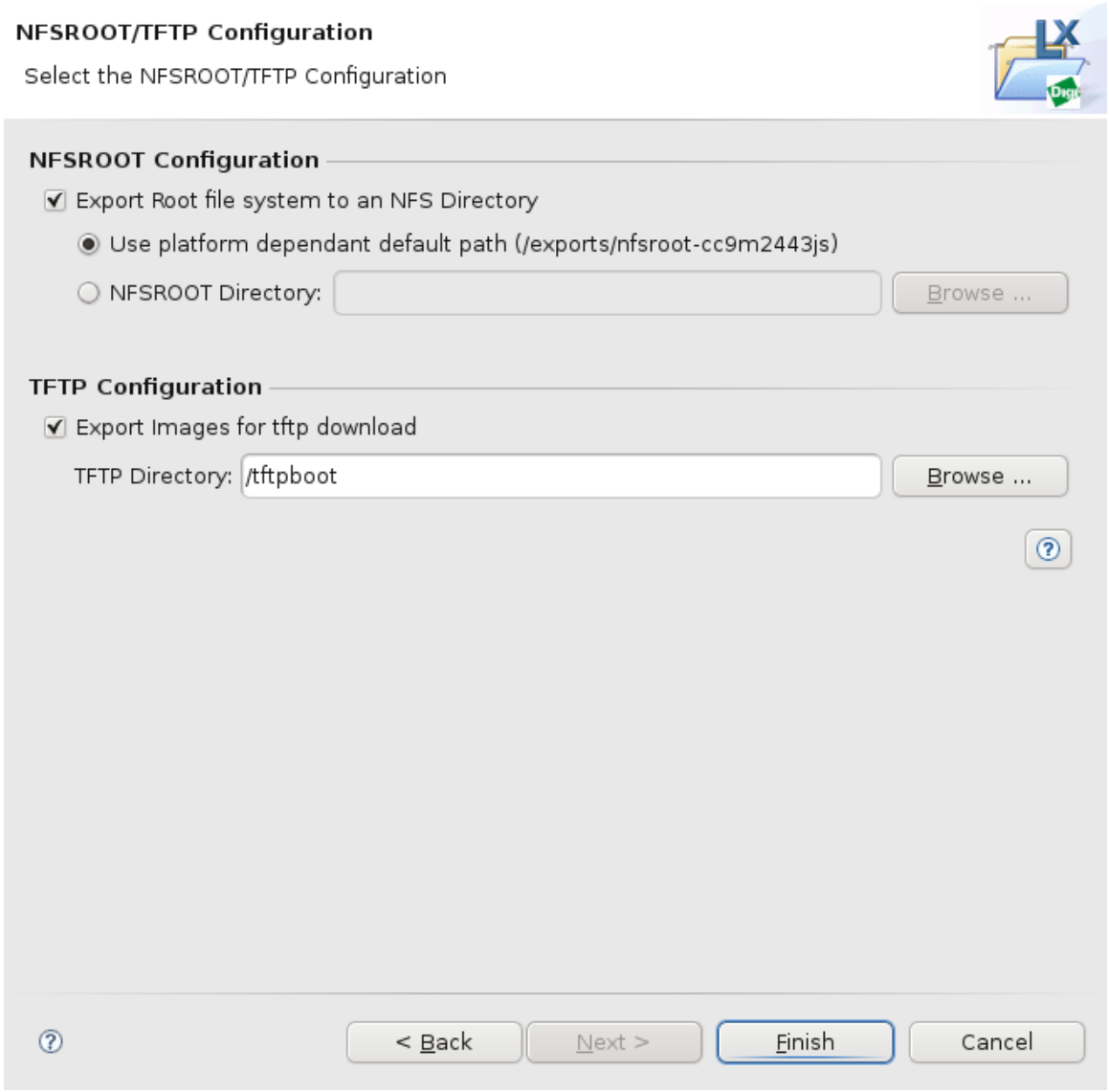
- Create a new Digi EL Kernel/Rootfs/U-Boot Project for your platform:



- Select Kernel and Root File System as project components:



- Select appropriate NFSROOT/TFTP Configuration




## 3.1 Download and cross compile PHP

Download a recent stable PHP source code package from http://php.net/. This document has been tested with php-5.2.13.tar.bz2 and store/deflate it into your project directory:

```
# cd $HOME/$YOURWORKSPACE/$YOURPROJECTNAME
# tar xjf php-5.2.13.tar.bz2
# cd php-5.2.13
```

Cross compile the PHP interpreter like this:

*Note: cross compilation will fail with php-5.3.2! Use php-5.2.13*

```
# export PATH=/usr/local/DigiEL-5.2/usr/bin:$PATH
```

```
# CC=arm-linux-gcc ./configure --host=arm-linux --prefix=/usr --without-iconv --
disable-xml --without-pear --disable-libxml --disable-dom --disable-simplexml --
disable-xmlreader --disable-xmlwriter --with-pdo-sqlite=/usr/local/DigiEL-
5.2/usr
# make
# make INSTALL_ROOT=$HOME/$YOURWORKSPACE/$YOURPROJECTNAME install
```

If you don't see php-cgi in your project directory (the make install failed?), copy it manually from built directory (or take the pre-compiled php-cgi from the zip attached).

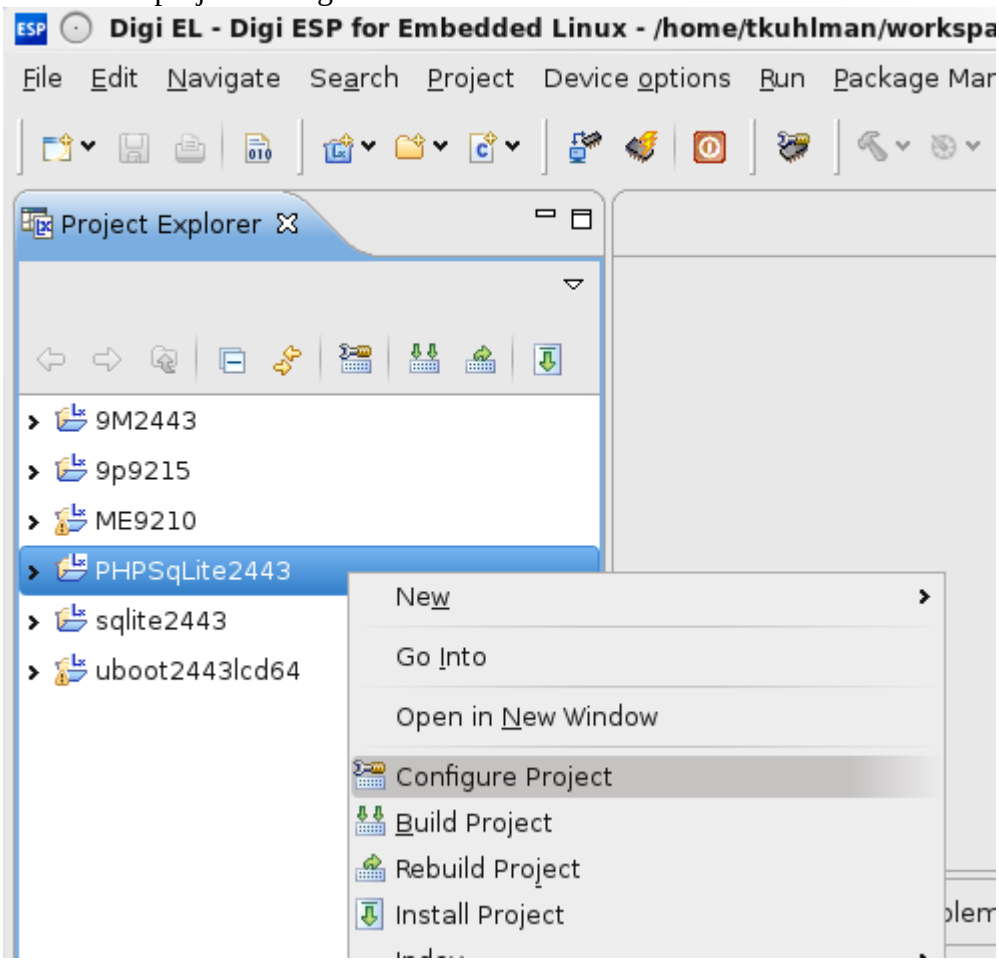
```
# find . -name php-cgi
./sapi/cgi/php-cgi
# cp sapi/cgi/php-cgi ..
```

## 3.2 Cross compile example application

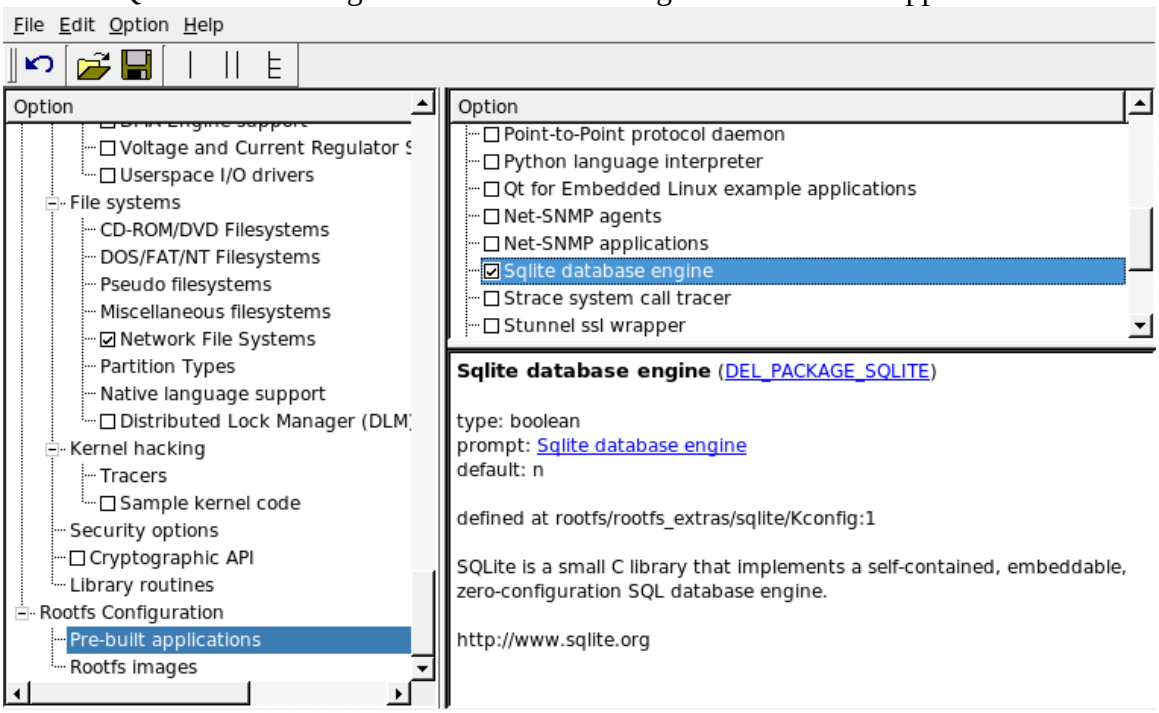
```
# export PATH=/usr/local/DigiEL-5.2/usr/bin:$PATH
# cd $HOME/$YOURWORKSPACE/$YOURPROJECTNAME
# arm-linux-gcc sqlite_test.c -o sqlite_test -lsqlite3
```

## 3.3 configure Rootfs to contain SQLite pre-compiled binary

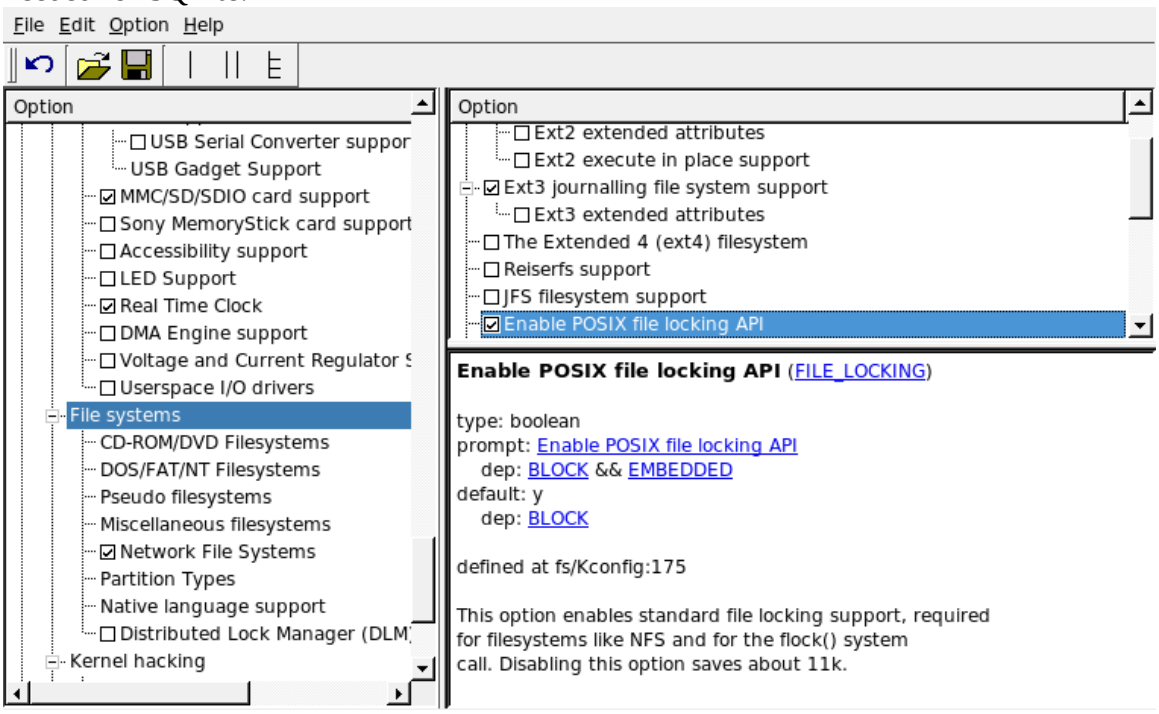
Launch the project configuration:



Enable SQlite data base engine in the Rootfs Configuration/Pre-built applications:



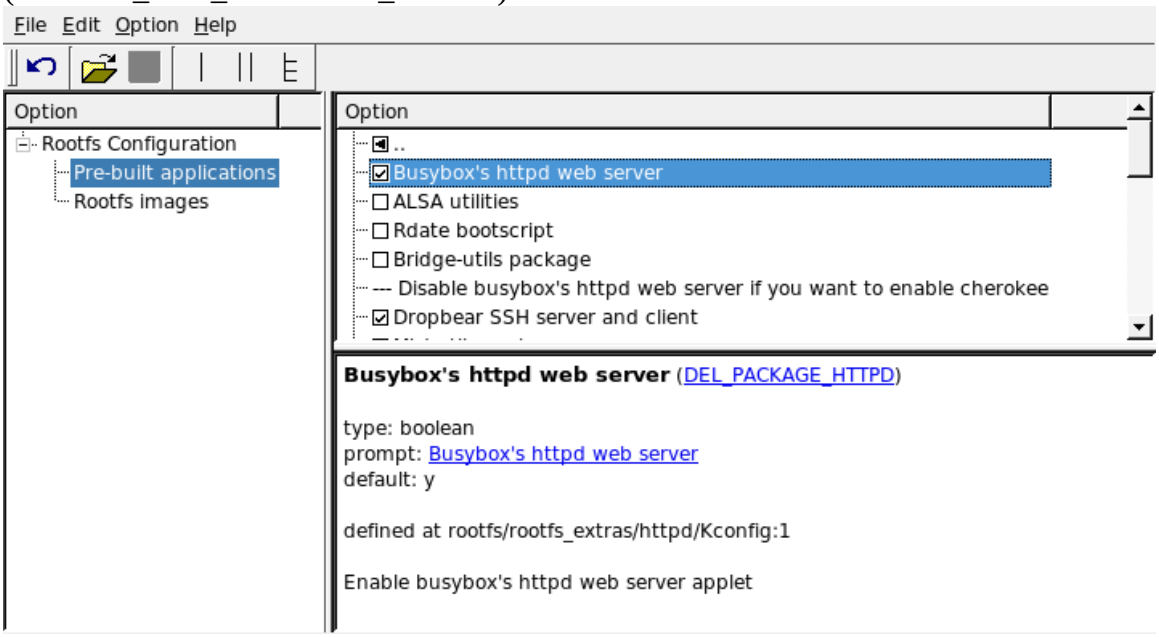

configure the kernel for POSIX file locking (CONFIG_FILE_LOCKING) which is needed for SQLite:




Save the changes.

## 3.4 configure Busybox httpd

If you decide to use the pre-compiled httpd included in the busybox, you need to select it into your project config (which is the default). Right click on your project and select "Configure project". Select Rootfs Configuration, Pre-built applications and Busybox's httpd web server
(CONFIG_DEL_PACKAGE_HTTPD)



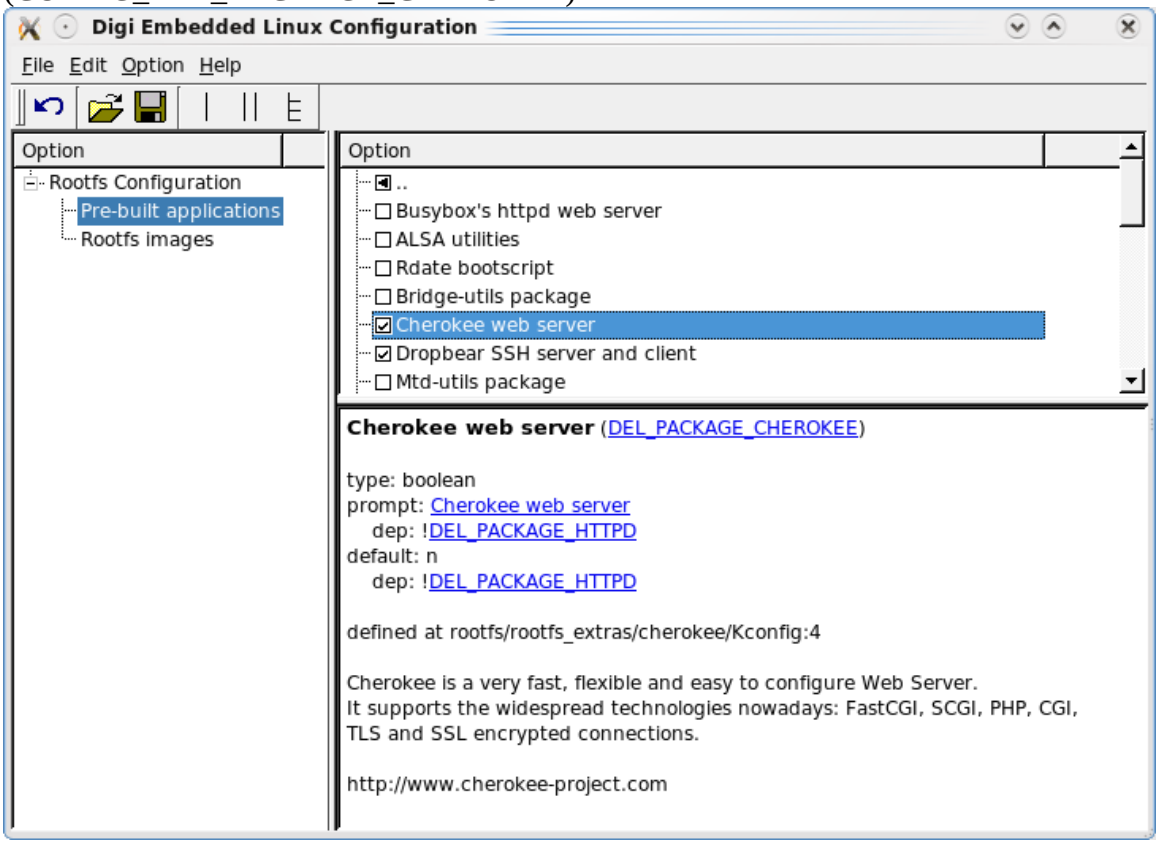

To enable the busybox httpd to launch the PHP interpreter when called with .php in the URL, add the following line to the /etc/httpd.conf of your rootfs:

```
*.php:/usr/bin/php-cgi
```

E.g. copy the default /usr/local/DigiEL-5.1/rootfs/rootfs_extras/httpd/rootfs/etc/httpd.conf into your project directory and edit it.

## 3.5 configure Cherokee webserver

If you decide to use the pre-compiled cherokee web server, you need to select it into your project config (disable busybox httpd for this first). Right click on your project and select "Configure project". Select Rootfs Configuration, Pre-built applications deselect Busybox's httpd web server and select Cherokee (CONFIG_DEL_PACKAGE_CHEROKEE)



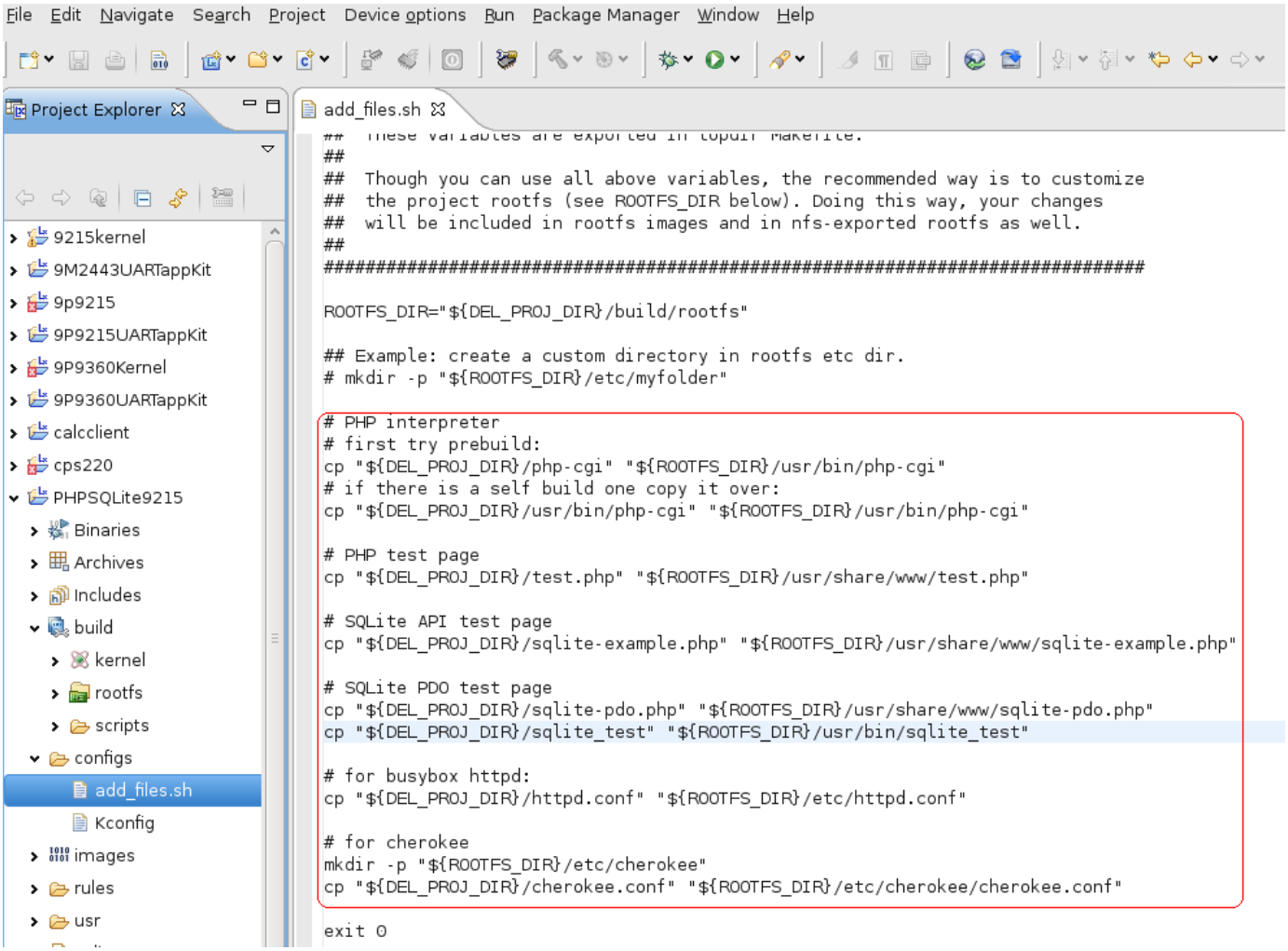

To configure Cherokee to launch the PHP interpreter, add the following lines to your cherokee.conf:

```
vserver!10!rule!600!encoder!deflate = 0
vserver!10!rule!600!encoder!gzip = 1
vserver!10!rule!600!handler = fcgi
vserver!10!rule!600!handler!balancer = round_robin
vserver!10!rule!600!handler!balancer!source!1 = 1
vserver!10!rule!600!handler!change_user = 0
vserver!10!rule!600!handler!check_file = 1
vserver!10!rule!600!handler!error_handler = 1
vserver!10!rule!600!handler!pass_req_headers = 1
vserver!10!rule!600!handler!xsendfile = 0
vserver!10!rule!600!match = extensions
vserver!10!rule!600!match!extensions = php
vserver!10!rule!600!only_secure = 0
source!1!host = localhost:50000
source!1!interpreter = /usr/bin/php-cgi -b localhost:50000
source!1!nick = PHP Interpreter
source!1!type = interpreter
```

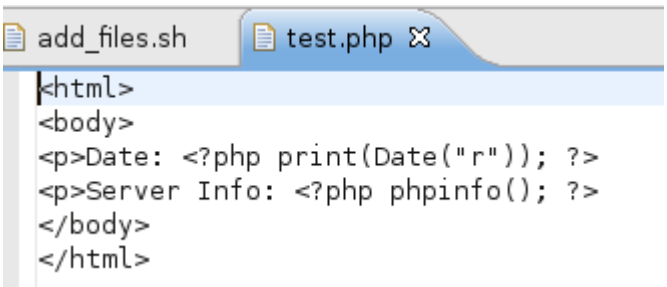
E.g. copy the default /usr/local/DigiEL-5.1/rootfs/rootfs_extras/cherokee/rootfs/etc/Cherokee/cherokee.conf into your project directory and edit it.

## 3.6 add files to your rootfs

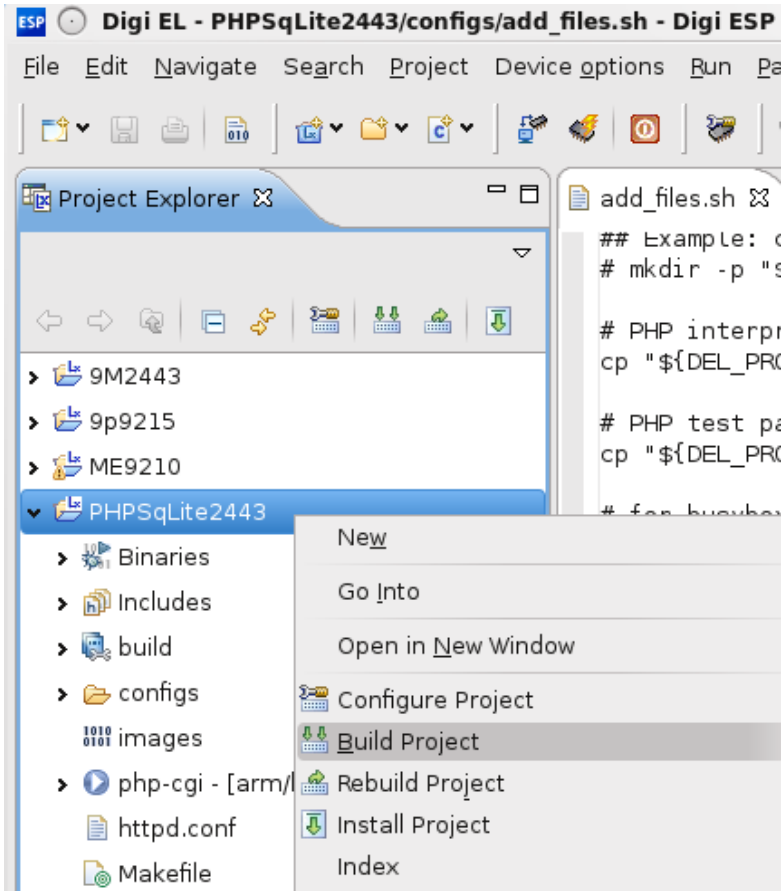
Unfold your project, select configs and open add_files.sh. Add the php-cgi, the test.php and the httpd.conf or cherokee.conf (according to your selected webserver) to the rootfs:



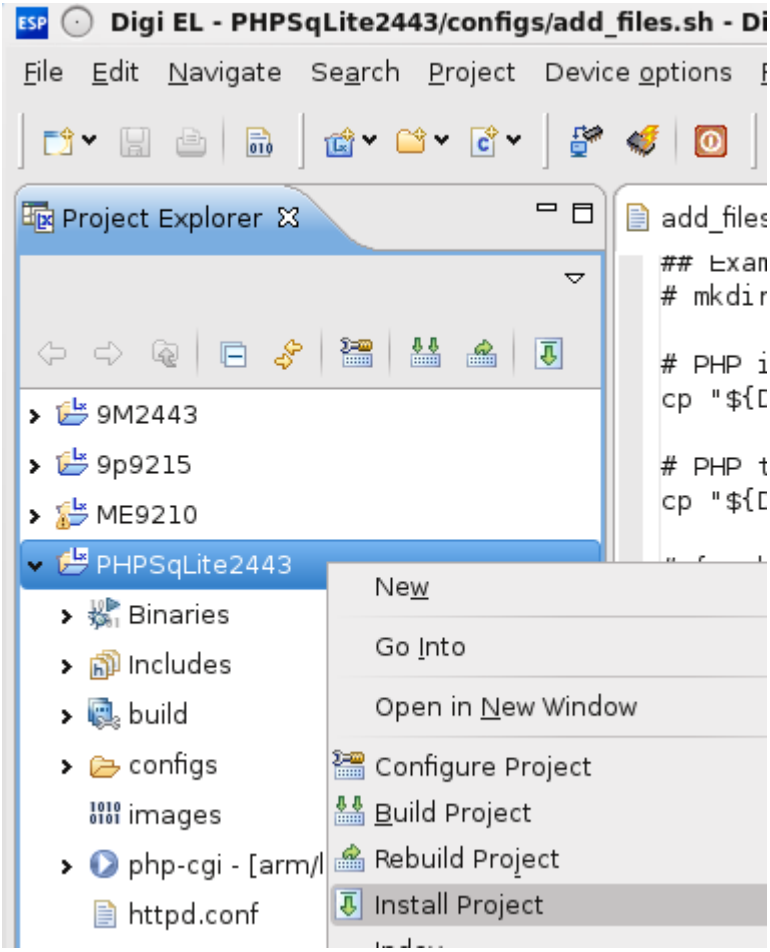

Also add test.php, sqlite-example.php sqlite-pdo.php to your project directory.



- Built the project:



- Install the project:




# 4 Hardware Setup

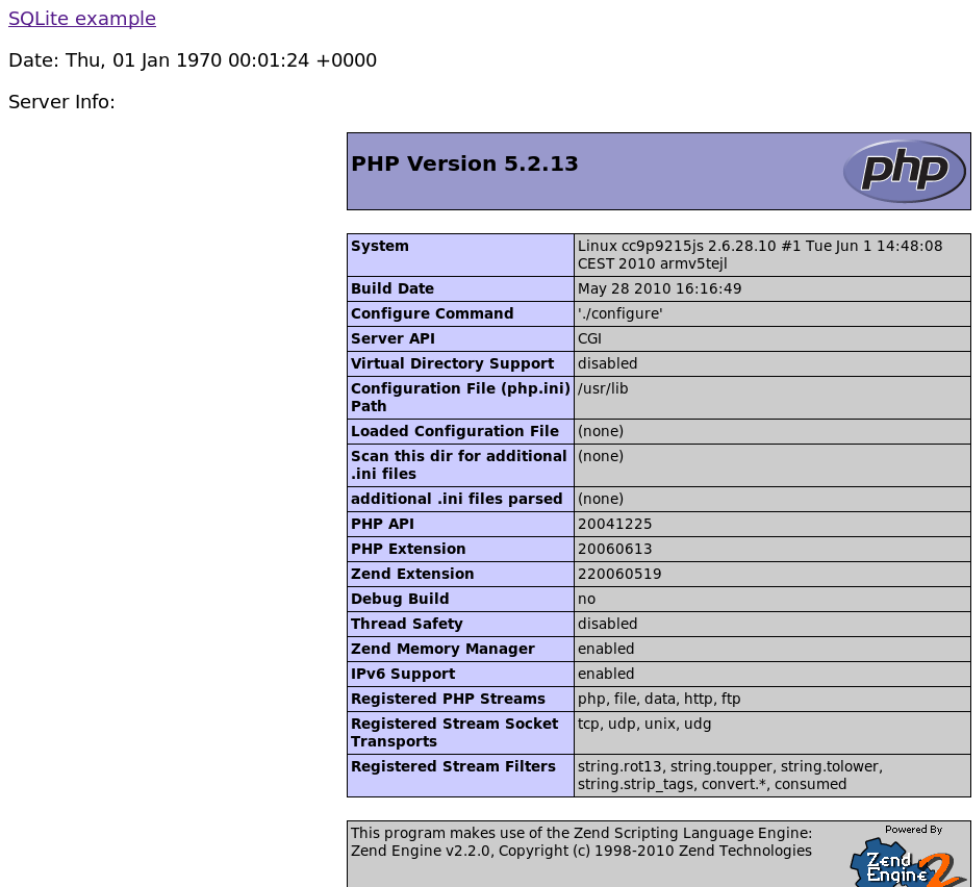
- Turn off the development board.
- Connect the power cable.
- Plug the Connect ME 9210 or ConnectCore module to the development board.
- Connect Ethernet between development board and your development PC or switch (for updating firmware).
- Connect a serial null modem cable (pins 2 and 3 crossed) to your host computer (e.g. COMA is the CONSOLE). Plug the cable into Serial Port A of the Digi development board. On ConnectCore 9P9215 development board use Serial Port D (CONSOLE).

# 5 Testing

Run the new built kernel and rootfs on module (e.g. update the images in flash built in section 3 “update linux tftp;rootfs tftp;dboot linux flash”, or boot with NFS-root: “dboot linux tftp”).

## *5.1 Testing with sqlite_open() API (SQlite 2.8.17)*

Start a web browser on your development host and point it to the IP of your module to check the php test page: http://192.168.42.30/test.php



SQLite example

Date: Thu, 01 Jan 1970 00:01:24 +0000

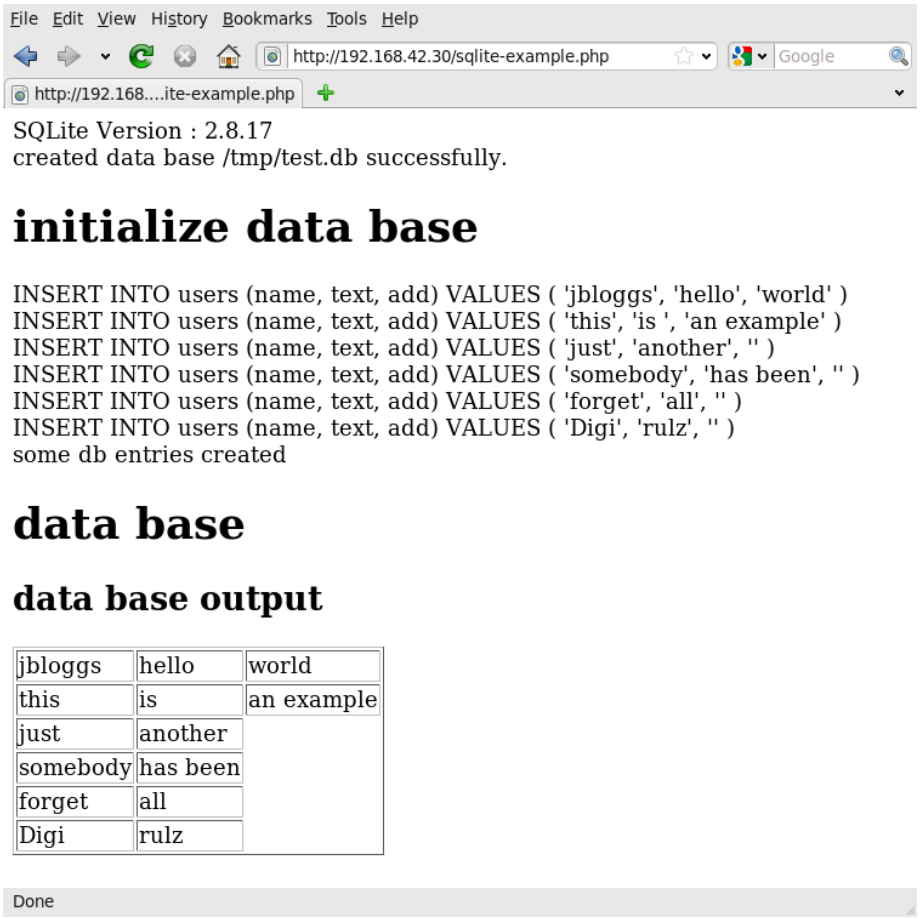
Server Info:

**PHP Version 5.2.13**

| | |
|---|---|
| **System** | Linux cc9p9215js 2.6.28.10 #1 Tue Jun 1 14:48:08 CEST 2010 armv5tejl |
| **Build Date** | May 28 2010 16:16:49 |
| **Configure Command** | './configure' |
| **Server API** | CGI |
| **Virtual Directory Support** | disabled |
| **Configuration File (php.ini) Path** | /usr/lib |
| **Loaded Configuration File** | (none) |
| **Scan this dir for additional .ini files** | (none) |
| **additional .ini files parsed** | (none) |
| **PHP API** | 20041225 |
| **PHP Extension** | 20060613 |
| **Zend Extension** | 220060519 |
| **Debug Build** | no |
| **Thread Safety** | disabled |
| **Zend Memory Manager** | enabled |
| **IPv6 Support** | enabled |
| **Registered PHP Streams** | php, file, data, http, ftp |
| **Registered Stream Socket Transports** | tcp, udp, unix, udg |
| **Registered Stream Filters** | string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed |

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.2.0, Copyright (c) 1998-2010 Zend Technologies

If you click on the SQLite example link (or enter URL: http://192.168.42.30/sqlite-example.php) you will see the output of the SQL statements executed in this PHP page.

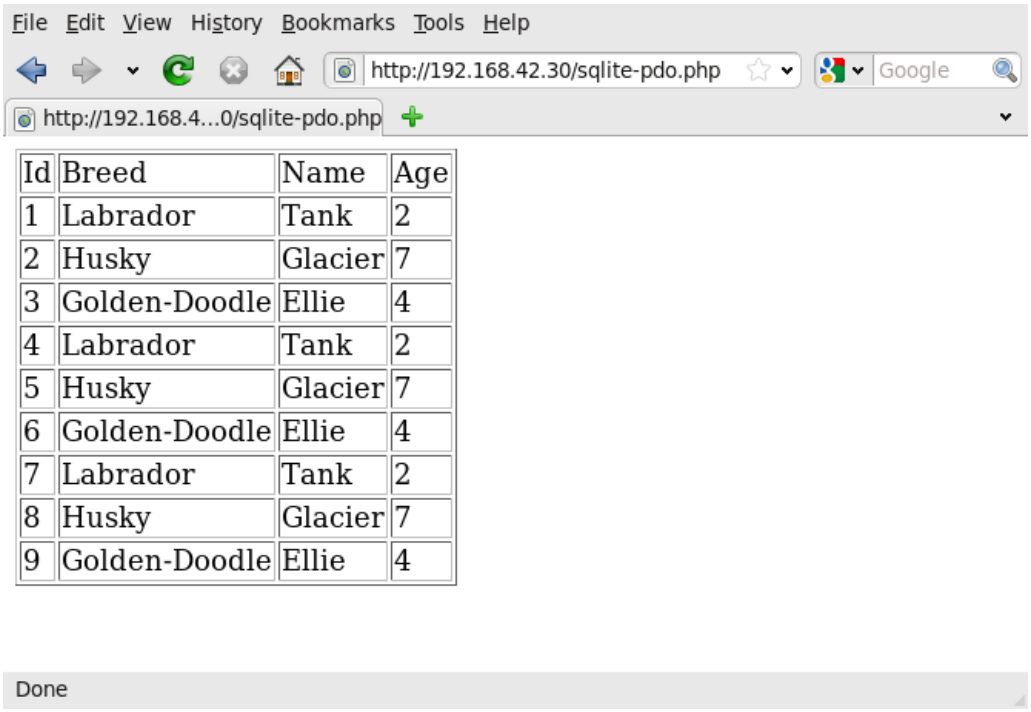



If you execute this a second time, you will get an error message, since the data base is already existing:


The generated test data base /tmp/test.db should by in SQlite 2.8.17 format.

## *5.2 Testing with PDO interface (SQlite 3.x support)*

Point your browser to sqlite-pdo.php example, to create a SQLite 3 example data base with PHP and PDO interface:




| Id | Breed | Name | Age |
|---|---|---|---|
| 1 | Labrador | Tank | 2 |
| 2 | Husky | Glacier | 7 |
| 3 | Golden-Doodle | Ellie | 4 |
| 4 | Labrador | Tank | 2 |
| 5 | Husky | Glacier | 7 |
| 6 | Golden-Doodle | Ellie | 4 |
| 7 | Labrador | Tank | 2 |
| 8 | Husky | Glacier | 7 |
| 9 | Golden-Doodle | Ellie | 4 |

On the serial console, run the sqlite_test program to access the PHP created SQLite 3 data base:

```
/ # sqlite_test /tmp/dogs_pdo.sqlite 'select * from Dogs'
Id = 1
Breed = Labrador
Name = Tank
Age = 2

Id = 2
Breed = Husky
Name = Glacier
Age = 7

Id = 3
Breed = Golden-Doodle
Name = Ellie
Age = 4

Id = 4
Breed = Labrador
Name = Tank
Age = 2

Id = 5
Breed = Husky
Name = Glacier
Age = 7

Id = 6
Breed = Golden-Doodle
Name = Ellie
Age = 4
```