# Setting up the Neato SmartApps Server

Raja Software

Last updated: 01/26/2014

|  |  |  |
| --- | --- | --- |
| Date | Version | Author/Comments |
| 12/11/2012 | 0.1 | Initial draft |
| 12/18/2012 | 0.2 | Adding more details to app config and added folder screenshots. |
| 12/19/2012 | 0.3 | Add steps to setup ejabberd, assumptions section, etc. |
| 12/20/2012 | 0.4 | Added Troubleshooting section |
| 02/07/2013 | 0.5 | Updated screenshots, generic IP/port info, etc. |
| 05/28/2013 | 0.6 | Updated for RabbitMQ |
| 10/17/2013 | 0.7 | Updated for WordPress Integration |
| 11/06/2013 | 0.8 | Added more details about the WordPress setup |
| 01/26/2014 | 0.9 | Added more details on ejabberd mod\_eventful setup, troubleshooting tips. |

# Summary

This document explains how to setup the Neato SmartApps server application on Production Server and Developer machines. This document first explains how to setup the required environment and then how to setup the code and relevant databases, configurations.

# Assumptions

This document assumes that server environment is Ubuntu and development environment is Windows.

If Server environment is any other flavor or Linux, steps mentioned in the Server Environment Setup will still work with minor changes. If the development environment is Mac OS X, *Development Environment Setup* steps will still work fine with minor path changes.

This document uses the host name and IP address in various places. Please note that:

* IP address of a machine can be found by typing command “ipconfig” on windows or “ifconfig” on a Linux machine.
* You can use “localhost” OR “127.0.0.1” as IP address, instead of the machine’s IP address if you are browsing the application from the same machine from where your application is running (i.e. Apache web server is running on the same machine).
* Host name is the DNS name that is mapped to the specific IP address.

If you come across a line like “*open a browser window and type: http://<ipaddress>/ neato/index.php”*, you can type any of these four addresses in the browser and it would give some output:

* [http://myip/neato/index.php](http://50.56.121.252/neato/index.php)
* <http://myhostname/neato/index.php> assuming that myhostname.com is mapped to IP address **myip**
* <http://localhost/neato/index.php> if you opened the browser window on same machine where the application is running.
* <http://127.0.0.1/neato/index.php> if you opened the browser window on same machine where the application is running.

# Requirements

Neato web application requires following installations on the server/developer machines:

* PHP 5.3 and above (<http://php.net/>)
* Apache 2.2.22 web server (<http://www.apache.org/>)
* MySQL 5.5 and above (<http://www.mysql.com/>)
* PHPMyAdmin (<http://www.phpmyadmin.net/>) (An optional web based MySQL management console if you want easy access to MySQL database instead of using command line utilities.)
* ejabberd (<http://www.ejabberd.im/>) (This is required on server but can be skipped for development environment.)
* RabbitMQ ([http://www.rabbitmq.com](http://www.rabbitmq.com/)). This is required on server but is optional on the development environment.

The next few sections explain how to install these dependencies on Server and Developer environments.

# Server Environment Setup

### PHP, MySQL and Apache installation

First, login as root on your Linux instance and follow the steps below. If you do not have direct access to root account, you would need to add **sudo** in-front of every command mentioned below like

**mkdir abc** would change to **sudo mkdir abc**

* "apt-get" is a tool to do the 'installations' on Ubuntu machines. First update apt-get itself by running this command so that we get all the installations to the latest versions:

**apt-get update**

* Install PHP, Apache, and the 'connectivity' that we require between these two.

**apt-get install apache2 php5 libapache2-mod-php5**

After the command above, if you open your browser and type

[**http://myip**](http://myip) **OR** [**http://myhostname.com**](http://neatoroboticswebapp.com)

You should see 'It’s working!' as a response. This means that Apache is running fine.

* Apache creates a new directory structure inside var folder. If you go to **/var/www/** directory you would see an **index.php** file. This file has “it’s working!” written here. Thing to note here is, by default **/var/www/** is treated as the document root directory of Apache. This means, whatever you would be adding inside **/var/www/** directory**,** would be served by Apache. For example, if you create a folder named test inside **/var/www** (i.e. ***/var/www/test/***) and put a text file named ***sample.txt*** inside test folder (i.e. **/var/www/test/sample.txt**), you can see this file by typing in following URL in your web browser:

[http://myip/test/sample.txt](http://50.56.121.252/test/sample.txt)

* As application also needs MySQL, lets install mysql-server, mysql client and php5-mysql connectivity related libraries. You would be asked for password for the root user of the MySQL in this installation step. Remember this password as it would come handy when you are setting up the Application environment.

**apt-get install mysql-server mysql-client php5-mysql**

* After the command above, you would be able to access the MySQL using the command line client by giving following command. Note that there will not be any space between -p and the password that you gave in the steps mentioned above.

**mysql -u root -p<Password that you gave in the step above>**

* By now, we got PHP, Apache, and MySQL Setup. Problem is, it is tedious for the developers to login to the system and use MySQL using command line.
* We would install phpmyadmin that would help us access this MySQL instance directly using a browser. The command below would install phpmyadmin. It would first ask you on which web server you want to run phpmyadmin, please select apache. Next it would ask you admin password for phpmyadmin.

**apt-get install phpmyadmin**

* After this installation you should be able to view the phpmyadmin by going to following URL in your browser. From here you can create the DB Schema/data that you need for application. PHPMyadmin has a decent help manual, please refer it in case you need any help in there.

[**http://myip/phpmyadmin**](http://myip/phpmyadmin) **OR** [**http://myhostname.com/phpmyadmin**](http://myhostname.com/phpmyadmin)

* By now, you are done with the basic setup and now we are ready to move your code. If you want to move the code to the root directory, you can do so by using **SCP/Filezilla** and move codebase to **/var/www** directory. Make sure that you remove index.html or else you would have to explicitly mention **http://<IP\_ADDRESS>/index.php** to see your code working (as by default, index.html takes precedence over index.php, in Apache).

As the source code is checked into Subversion (SVN), we need to install the clients for them using following commands:

**apt-get install subversion**

* Few other small setup that come handy are mentioned below. Ubuntu 11, comes with nano editor. If you like vi over any other text editors, you can install it by giving following commands. You can edit your config giles using vim (vi) or nano

**apt-get install vim**

* Other module that we need is, enabling mod\_rewrite (an apache module which is used for SEO friendly URLs).. Type following command for the same

**a2enmod rewrite**

* Now we have enabled the mod rewrite but Apache still will not allow the rewrites as the default behavior of Apache is to not to disable allow rewrites.
* Follow the command below and make changes as mentioned below. Make sure that you should NOT have **AllowOverride = None** and it should be changed to **All** here.

**vim /etc/apache2/sites-available/default**

**<Directory /var/www/>**

**Options Indexes FollowSymLinks MultiViews**

**AllowOverride *All***

**Order allow,deny**

**allow from all**

**</Directory>**

After that you would have to restart the apache so that this **mod\_rewrite** can take effect. The command below would restart the apache.

**/etc/init.d/apache2 restart**

Application also needs **curl** tosupport various web services. You can enable curlby this command:

**apt-get install php5-curl**

* Now let’s install pear and mail and smtp packages that are required for sending emails.

**apt-get install php-pear**

**pear install -o Mail**

**pear install -o Net\_SMTP**

* Other thing that we need is changing some PHP settings. We want to change the file upload limit so that user can upload bigger maps/schedule related data.

**vim /etc/php5/apache2/php.ini**

and edit

**post\_max\_size = 32M**

**upload\_max\_filesize = 32M**

Note that ANY change in php.ini will NOT take effect unless you restart the server

**/etc/init.d/apache2 restart**

### Ejabberd

The Neato SmartApps mobile application and web application uses XMPP server to interact with robots, and vice-versa. We have used ejabberd as the XMPP server.

**Please note that XMPP server needs to run on the same machine where the apache web server is running.**

To install ejabberd, run:

**sudo apt-get install ejabberd**

After it is installed, you need to edit the configuration file that can be found at

***/etc/ejabberd/ejabberd.cfg***

Open this configuration file in an editor (we have been using vi) and add the host name you want the server to use. We would be changing the line where it mentions

**{hosts, ["localhost"]}.**

to

**{hosts, ["localhost","myhostname.com"]}**

Please note that host name here (**myhostname.com**) would mean that all the chat IDs that are created are created with suffix @ **myhostname.com**. Please take a note of the host name that you have specified as it would be required while you are providing the host name in the configuration file of the application (explained at the later part of this document).

Additionally, ejabberd has a web based admin console so that you can manage this XMPP server from a web console. Access to this console needs to be restricted so we would create an admin user. In the ***/etc/ejabberd/ejabberd.cfg*** config file, let’s add an admin level user. Please change the line that says,

**{acl, admin, {user, "", "localhost"}}**

to

**{acl, admin, {user, "administrator", "myhostname.com"}}.**

This change would add an admin level user named “administrator” (you can use any other username that you prefer) to the ejabberd admin for the host **"myhostname.com".**

If you notice, we have not given a password to this administrator user. Please run following command:

**sudo ejabberdctl register administrator myhostname.com myPassword**

Please change **myPassword** to the administrator password that you want to give. As any configuration changes would not take effect until you restart the ejabberd server, let’s restart the service by giving following commands:

**sudo service ejabberd restart**

Now you can browse the administration console of this XMPP server by going to the URL:

<http://myip:myport/admin/> or <http://myhostname.com:myport/admin/>

Typically, ejabberd runs on port 5280 (replace myport with 5280 above) – this is configurable in the ***ejabberd.cfg*** file. (Note: If are having trouble accessing the admin page it may be because port 5280 is blocked by your firewall – see the troubleshooting section for all the ports you need to open for this setup).

For detailed information on how to use admin console of ejabberd and what all can be done from there, please refer online documentation of ejabberd at [http://www.ejabberd.im/files/doc/guide.html#htoc20](http://www.ejabberd.im/files/doc/guide.html%23htoc20)

There are two additional ejabberd modules that need to be installed – mod\_admin\_extra and mod\_eventful – these are discussed next.

#### mod\_admin\_extra - Module Installation

The mod\_admin\_extra module should be installed by default – but in some configurations it is not part of the default ejabberd installation. If it is not part of your install, you need to do the following steps to install the module.

Download ‘ejabberd-modules’ using ejabberd’s SVN repository

**svn co https://svn.process-one.net/ejabberd-modules**

Change the current directory to ‘ejabberd-modules/mod\_admin\_extra/trunk/’

**cd ejabberd-modules/mod\_admin\_extra/trunk/**

Compile the module

**./build.sh**

This will generate a  file ‘mod\_admin\_extra.beam’ and it will be present inside the ‘ebin’ folder.

Copy this file to the directory holding other .beam files for ejabberd

**cp ./ebin/mod\_admin\_extra.beam /lib/ejabberd/ebin/**

Enable the module by editing the config file. Open up ‘ejabberd.cfg’ file present at ‘/etc/ejabberd/’ folder.

**vi /etc/ejabberd/ejabberd.cfg**

Around line 581 in the modules section add the following code:

**{mod\_admin\_extra, []},**

Restart ejabberd server

**ejabberdctl restart**

The mod\_admin\_extra module is now part of ejabberd.

#### mod\_eventful - Module Installation

The mod\_eventful module is need to track the robot online status. If this module is not part of your default install, you need to do the following:

**git clone git://github.com/theozaurus/mod\_eventful.git**

**cd mod\_eventful**

**./build.sh**

**sudo cp ebin/\*.beam /usr/lib/ejabberd/ebin**

Update the configuration in /etc/ejabberd/ejabberd.cfg  - a sample configuration would look like the following:

**{mod\_eventful,**

**[**

**{url,**

**[**

**{online\_hook, "http://localhost/Neato\_Server/Server\_Yii/Neato/api/robotStatus/online"},**

**{offline\_hook, "http://localhost/ Neato\_Server/Server\_Yii/Neato/api/robotStatus/offline"},**

**{unset\_presence\_hook, undefined},**

**{set\_presence\_hook, undefined},**

**{message\_hook, undefined}**

**]**

**},**

**{user, "HTTP BASIC USERNAME"},**

**{password, "HTTP BASIC PASSWORD"}**

**]**

**}**

Then restart ejabberd to complete the installation of the mod\_eventful module.

### RabbitMQ

The Neato application uses RabbitMQ to send push notifications asynchronously. Please note that RabbitMQ needs to run on the same machine where the apache webserver is running.

#### Installing on Ubuntu

Latest package for the RabbitMQ server is not available on Ubuntu. To get the latest RabbitMQ (and avoid warnings about unsigned packages) add [public key](http://www.rabbitmq.com/rabbitmq-signing-key-public.asc) of RabbitMQ to your trusted key list using following commands.

wget http://www.rabbitmq.com/rabbitmq-signing-key-public.asc

sudo apt-key add rabbitmq-signing-key-public.asc

Once it is added to the trusted sites, update the apt-get itself by running

apt-get update

Install packages using command

sudo apt-get install rabbitmq-server

After successful installation of RabbitMQ, run following command to start RabbitMQ Server,

sudo invoke-rc.d rabbitmq-server start

To stop RabbitMQ Server, Run following command

sudo invoke-rc.d rabbitmq-server stop

There are various RabbitMQ clientside implementations available out of which we are using AMQPLib (<https://github.com/videlalvaro/php-amqplib>).

To setup AMQPLib, Take git clone using

git clone git://github.com/videlalvaro/php-amqplib.git

Or you can directly download the AMQPLib from

https://github.com/videlalvaro/php-amqplib.git

After downloading (or cloning), let’s assume you kept it inside: /var/www/php/amqplib directory

Class autoloading and dependencies are managed by composer so install it in directory of composer.json file (e.g. var/www/php-amqplib/) using

curl --silent https://getcomposer.org/installer | php

And then install the library dependencies and generate the autoload.php file using

php composer.phar install

#### Installing on Windows

Download and run the Erlang Windows Binary File from <http://www.erlang.org/download.html>. Then just run the installer, rabbitmq-server-3.1.1.exe which is available at

<http://www.rabbitmq.com/install-windows.html>.

It takes around 2 minutes, and will set RabbitMQ up and would be running it as a service, with a default configuration.

The RabbitMQ service starts automatically. You can stop/reinstall/start the RabbitMQ service from the Start Menu.

To setup AMQPLib in windows, Take git clone using or download and extract it.

git clone git://github.com/videlalvaro/php-amqplib.git

Let’s say it is kept at following location

C:\xampp\htdoc\php-amqplib\

Class autoloading and dependencies are managed by composer so install it in directory of composer.json file(e.g. C:\xampp\htdoc\php-amqplib\) using

php -r eval('?>'.file\_get\_contents('https://getcomposer.org/installer'));"

If the above fails due to file\_get\_contents, use the http url or enable php\_openssl.dll in php.ini and create a new .bat file alongside composer using

echo @php "%~dp0composer.phar" %\*>composer.bat

And then install the library dependencies and generate the autoload.php file using

php composer.phar install

#### Usage

If you decide to use any other implementation, please include the push\_notification\_standalone.php file in the consumer as all the notification logic is inside push\_notification\_standalone.php file.

If you want to use PHPAMQPLib, first include the push\_notification\_standalone.php file in the following file.

php-amqplib/demo/amqp\_consumer.php

After that, execute the following commands to start the consumer:

$ cd php-amqplib/demo

$ php amqp\_consumer.php

# Developer Environment Setup

### PHP, MySQL, PHPMyAdmin and Apache installation

If a developer is using Linux operating system, he would have to follow the steps mentioned in the Server Environment Setup. If developer is using Window Operating Systems, please follow the steps below.

On Windows, instead of installing PHP/MySQL/Apache separately, we would be using a packed installer called XAMPP (NOT to be confused with XMPP chat server) from ApacheFriends

(<http://www.apachefriends.org/en/xampp-windows.html>) and it would install PHP, MySQL, PHPMyAdmin and Apache web server.

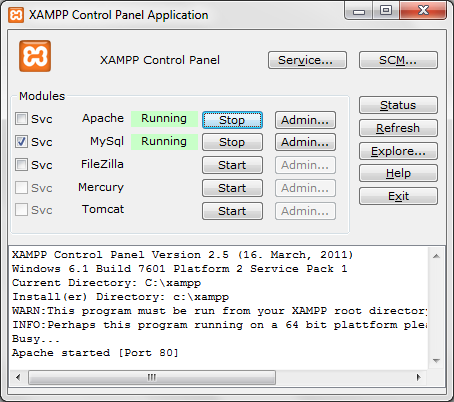
* Download packaged XAMPP on your machine from <http://www.apachefriends.org/en/xampp-windows.html> URL anywhere you want to install it.
* Install this downloaded exe file with default settings. In installation steps, it would ask for the place where you want to install this XAMPP package. Our assumptions is that you installed in C:\ directory (i.e. C:\xampp)
* This would install PHP, MySQL and Apache web server on your machine.
* If you do default installation, you would notice that after installation, a directory would be created inside your c:\ directory named xampp (i.e. **c:\xampp**).
* **C:\xampp** is the document root for the Apache. This means if you go to

[**http://myip/neato/index.php**](http://50.56.121.252/neato/index.php)you would be browsing **index.php** file inside **c:\xampp\neato** folder.

* When you install XAMPP, PHPMyAdmin is installed by default. You can open it by going to:

[**http://myip/phpmyadmin**](http://50.56.121.252/phpmyadmin)

* XAMPP package also provides an easy way to start/stop Apache and MySQL. Inside c:\xampp directory, you would find a file named **xampp-control.exe.** Click on this file and you can start and stop Apache and MySQL (as displayed in the screenshot below). To keep the Neato application running, you have to make sure that both MySQL and Apache are running fine.
* Open **c:\xampp\php\php.ini** and change the upload limits to 32M if you want to test map blob data with big files.
* Enable curl extension in same php.ini by uncommenting php\_curl in php.ini file. Curl extension is required by the web services as this extension enables HTTP client for PHP.



### Ejabberd

The Neato SmartApps mobile application and web application use XMPP server to interact with robots, and vice-versa. We have used ejabberd as the XMPP server. **Please note that XMPP server needs to run on the same machine where the Apache web server is running.** To install ejabberd on Windows,

* Download the executable from <http://www.process-one.net/en/ejabberd/downloads/>
* Run it as administrator and follow the installation steps
* Let’s say it was installed inside **c:\Program Files (x86)\ejabberd-2.1.9\bin,** to startejabberd service,
  + Open command prompt and go to **c:\Program Files (x86)\ejabberd-2.1.9\bin** and type

**ejabberdctl start**

* + You can also see the status of your ejabberd by using

**ejabberdctl status**

* Please note that the host for this instance would be “localhost”. We do not need to change it or create any administrator account as it is installed on development environment.

### RabbitMQ

Follows same step which are used to setup RabbitMQ on Server Environment.

# Application Setup on Server

**Setup code**

* If you do not have access to repository,
  + Move the provided code in the /var/www/
* If you do have access to repository,
  + Go to the folder /var/www and do an **SVN checkout**

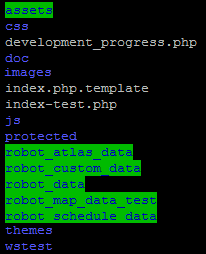
**cd /var/www/**

**svn co <path to the checkout url>**

* After taking a checkout, your directory structure should be **/var/www/server/neato and /var/www/server/yii-1.1.12.b600af**

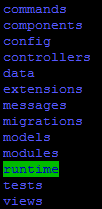


* Create following folders inside the /var/www/server/neato folder (i.e.at the same level as the protected folder)
  + **assets** (This folder is used by Yii framework to create minimized and compiled versions of JavaScript and CSS files)
  + **robot\_map\_data** (This directory would store the robot\_map\_data. You need to specify this directory name in main.php)
  + **robot\_schedule\_data** (This directory would store the robot schedule data. You need to specify this directory name in main.php)
  + **robot\_custom\_data** (This directory would store any custom data that wee store for robot. You need to specify this directory name in main.php)
  + **robot\_atlas\_data** (This directory would store the robot atlas data. You need to specify this directory name in main.php)
  + After making all these changes, your directory structure inside /var/www/server/neato should be:



* Create another directory inside /var/www/server/neato/protected/ called **runtime.**

After making all these changes, your directory structure inside /var/www/server/neato/protected should be:



* Give read/write permission to all the users or just the Apache user (i.e. user named www-data) on assets, robot\_custom\_data, robot\_map\_data, robot\_schedule\_data, robot\_atlas\_data and the runtime directories.
  + **chmod –R 777 /var/www/server/neato/assets**
  + **chmod –R 777 /var/www/server/neato/robot\_schedule\_data**
  + **chmod –R 777 /var/www/server/neato/robot\_map\_data**
  + **chmod –R 777 /var/www/server/neato/robot\_atlas\_data**
  + **chmod –R 777 /var/www/server/neato/robot\_custom\_data**
  + **chmod –R 777 /var/www/server/neato/protected/runtime**
* After this you should be able to view the web application by hitting following URLs:

[**http://myip/server/neato**](http://50.56.121.252/server/neato) **OR** [**http://myhostname.com/server/neato**](http://neatoroboticswebapp.com/server/neato)

### Database Setup

* Now go to browser window and go to

[**http://myip/phpmyadmin**](http://myip/phpmyadmin) **OR** [**http://myhostname.com/phpmyadmin**](http://myhostname.com/phpmyadmin)

* Login to the phpmyadmin using root and the password given while you were installing the PHPMyAdmin
* Create a new database (let’s say named neato)
* Click on import and browse to the db\_schema and select neato\_schema.sql file and click OK.
* The above step would import the database in the Neato database.
* This DB dump file has all the meta-data required. This DB dump also has an admin account created (admin@neatorobotics.com/neatorsl123) by default.

### Configuration Changes

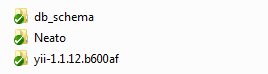
There are 2 configuration files that you need to rename and configure to get up and running.

* **/var/www/server/Neato/index.php.template**: Rename this file to just index.php. This hook is left so that staging/production environment can be configured separately.
* **/var/www/server/Neato/protected/config/main.php.template**: This file contains all the configuration parameters that you would require to contextualize the application. All the placeholders for various configurations are prefixed with “<YOUR\_”. Few changes that are recommended are:
  + SMTP Mail configuration parameters like host, username and password. This information is configurable from **mail array** in this file. This SMTP configuration is used for sending registration and forgot password emails. (The placeholders that you would need to replace are <YOUR\_SMTP\_HOST\_HERE>, <YOUR\_SMTP\_USER\_NAME\_HERE>, <YOUR\_SMTP\_PASSWORD\_HERE>)
  + Database configuration parameters like host, DB name, password. This information is configurable from **db array** in this file.( The placeholders that you would need to replace are <YOUR\_DB\_USERNAME\_HERE>, <YOUR\_DB\_PASSWORD\_HERE>)
  + We are using Facebook app to enable Facebook login. You should set app id and secret key in the **facebook array** in this file. (The placeholder that you need change are <YOUR\_FACEBOOK\_APP\_ID\_HERE>, <YOUR\_FACEBOOK\_APP\_SECRET\_HERE>)
  + As per earlier setup, whatever host you have specified while setting up ejabberd, should be mentioned as value against **ejabberdhost** key. (The placeholder that you would need to replace is <YOUR\_JABBER\_HOST\_HERE>)
  + All the query string parameters that are passed in the URLs follow 2-way encryption. You can specify your own value that is used as salt for encrypting a parameter in the **two-way-encrypt-key** key. (The placeholder that you would need to replace is <YOUR\_ENCRYPTION\_KEY\_HERE>)
  + On development environment, you might want disable Ejabbers. You can do that by setting **isjabbersetup** to false.
  + You can also change the log level by setting the **levels** key inside **log** array**.** Possible values are **error, warning** and **debug.**
  + If you want to name the directories that contain robot related data you can create them with different name and mention their names corresponding keys like **robot-schedule\_data\_directory\_name** and so on. For example, if you want to store all the schedule data for the robot inside a directory name robot\_schedule instead of robot\_schedule\_data, as mentioned earlier, you can do that by creating the directory with robot\_schedule name and specifying this against **robot\_schedule\_data\_directory\_name** key.
  + As we are using RabbitMQ to send notifications asynchronously, add path of AMQPLib publisher in params array (The placeholder that you would need to replace is <YOUR\_NEATO\_AMQP\_PUBLISHER\_PATH\_HERE>)
* After setting all these configuration parameters, rename this file to main.php inside same folder

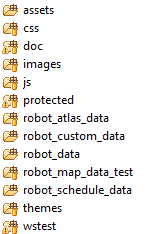
# Application Setup on Developer Machine

**Setup code**

* If you do not have access to repository,
  + Move the provided code in the c:/xampp/htdocs
* If you do have access to repository,
* Go to the folder **c:/xampp/htdocs** and do an **SVN checkout.**
* Nowyou should seeadirectory structure like this.

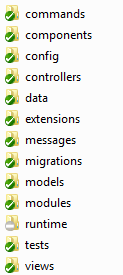


* Create following folders inside the **c:/xampp/htdocs**/**neato** folder (i.e.at the same level as the protected folder)
  + **assets** (This folder is used by Yii framework to create minimized and compiled versions of JavaScript and CSS files)
  + **robot\_map\_data** (This directory would store the robot\_map\_data. You need to specify this directory name in main.php)
  + **robot\_schedule\_data** (This directory would store the robot schedule data. You need to specify this directory name in main.php)
  + **robot\_custom\_data** (This directory would store any custom data that wee store for robot. You need to specify this directory name in main.php)
  + **robot\_atlas\_data** (This directory would store the robot\_atlas\_data. You need to specify this directory name in main.php)
  + After making all these changes, your directory structure inside /var/www/server/neato should be:



* Create another directory inside /var/www/server/neato/protected/ called **runtime.**

After making all these changes, your directory structure inside /var/www/server/neato/protected should be:

****

### Database Setup

* Now go to browser window and go to

[**http://myip/phpmyadmin**](http://myip/phpmyadmin) **OR** [**http://myhostname.com/phpmyadmin**](http://myhostname.com/phpmyadmin)

* Login to the phpmyadmin using root and the password given while you were installing the PHPMyAdmin
* Click on import and browse to the db\_schema folder and select neato\_schema.sql file and click OK.
* The above step would import the database in the Neato database.
* This DB dump file has all the meta data required. This DB dump also has an admin account created (admin/neatorsl123) by default.

### Configuration Changes

There are 2 configuration files that you need to rename and configure to get up and running.

* **C:/xampp/server/neato/index.php.template**: Rename this file to just index.php. This hook is left so that staging/production environment can be configured separately.
* **C:/xampp/server/neato/protected/config/main.php.template**: This file contains all the configuration parameters that you would require to contextualize the application. All the placeholders for various configurations are prefixed with “<YOUR\_”. Few changes that are recommended are:
  + Database configuration parameters like host, DB name, password. This information is configurable from **db array** in this file.( The placeholders that you would need to replace are <YOUR\_DB\_USERNAME\_HERE>, <YOUR\_DB\_PASSWORD\_HERE>)
  + SMTP Mail configuration parameters like host, username and password. This information is configurable from **mail array** in this file. This SMTP configuration is used for sending registration and forgot password emails. (The placeholders that you would need to replace are <YOUR\_SMTP\_HOST\_HERE>, <YOUR\_SMTP\_USER\_NAME\_HERE>, <YOUR\_SMTP\_PASSWORD\_HERE>)
  + We are using Facebook app to enable Facebook login. You should set app id and secret key in the **facebook array** in this file. (The placeholder that you need change are <YOUR\_FACEBOOK\_APP\_ID\_HERE>, <YOUR\_FACEBOOK\_APP\_SECRET\_HERE>)
  + One more facebook detail need to set is facebook app name in **params array.**  (The placeholder that you need change is <YOUR\_FACEBOOK\_APP\_NAME\_HERE>)
  + As per earlier setup, whatever host you have specified while setting up ejabberd, should be mentioned as value against **ejabberdhost** key. (The placeholder that you would need to replace is <YOUR\_JABBER\_HOST\_HERE>)
  + All the query string parameters that are passed in the URLs follow 2-way encryption. You can specify your own value that is used as salt for encrypting a parameter in the **two-way-encrypt-key** key. (The placeholder that you would need to replace is <YOUR\_ENCRYPTION\_KEY\_HERE>)
  + On development environment, you might want disable Ejabbers. You can do that by setting **isjabbersetup** to false.
  + You can also change the log level by setting the **levels** key inside **log** array**.** Possible values are **error, warning** and **debug.**
  + If you want to name the directories that contain robot related data you can create them with different name and mention their names corresponding keys like **robot-schedule\_data\_directory\_name** and so on. For example, if you want to store all the schedule data for the robot inside a directory name robot\_schedule instead of robot\_schedule\_data, as mentioned earlier, you can do that by creating the directory with robot\_schedule name and specifying this against **robot\_schedule\_data\_directory\_name** key.
* After setting all these configuration parameters, rename this file to main.php inside same folder

# WordPress Integration

The SmartApps server has its own built-in user management module. However, it provides an option to integrate with an external WordPress user database. This configuration allows the SmartApps server to use the same user database that is part of the Neato Robotics website (<http://www.neatorobotics.com>) – hereafter referred to as the Neato “CORP” website (the CORP website is built using WordPress). Using this mechanism, a user can log into the SmartApps server using credentials from an account created in the CORP website.

If you want to setup a fresh WordPress installation and configure it to work with the SmartApps server, do the following steps:

* Download and unzip the latest version of WordPress inside a folder (let's say inside a folder named wp)
* Move wp folder inside c:/xmpp/htdocs/ OR /var/www directory depending upon whether you are using Windows OR Linux operating system
* Open your favorite browser and browse to <http://localhost/wp> and follow the WordPress installation steps. These steps would ask you to create a database and would create a sample WordPress site along with WordPress admin user account.

At this stage either you have a fresh WordPress installed or you have an existing instance of WordPress running. In order to enable the RESTful APIs for WordPress, there is a standard plugin json-api is available. This plugin is committed in the SmartApp code repository (/var/www/server/WordPress-connecor/json-api)

* Copy the json-api folder, and paste it inside wp/wp-content/plugins folder
* Login as WordPress admin on <http://localhost/wp/wp-login.php> and click on plugin link in the left menu.
* It would display all the available plugins. It would also display this newly added json-api plugin. Enable this plugin so that WordPress can expose the core features using JSON APIs.

If you do NOT want to enable WordPress integration, please set the variable is\_wp\_enabled to false. If you want to enable the WordPress integration, you would have to set the is\_wp\_enabled to false and also set the wordpress\_api\_url to the actual WordPress path as mentioned below:

These parameters should be set in the protected/config/main.php file:

**'is\_wp\_enabled' => true,**

**'wordpress\_api\_url' => '**[**http://localhost/wp/**](http://localhost/wp/)**'**

wordpress\_api\_url should be the URL of your WordPress installation. This WordPress installation must have JSON-API plugin enabled so that WordPress can expose the core features over JSON based web services and these web services can be consumed by the Neato SmartApp backend.

# Troubleshooting

* I am not able to start the web server.
  + By default Apache uses port 80 – make sure no other app (like Skype) is running. Try stopping the other app and restart Apache.
* I want to run the web server on some other port other than the default port (i.e. 80)
  + You would have to change the Apache’s configuration file to make these changes.
  + On Linux, open **/etc/apache2/ports.conf** and change the value **Listen 80** to **Listen 81**,if you want to run the Apache on port number 81.
  + On Windows, open **C:\xampp\apache\conf\httpd.conf** file and change the value **Listen 80** to **Listen 81**,if you want to run the Apache on port number 81.
  + As configuration changes would take effect only after a restart, restart the Apache by:
    - On Windows, using XAMPP Control panel
    - On Linux, by giving **/etc/init.d/apache2 restart** command
  + Please note that after this change you would have to append the port number in every call. For example, you would have to call it <http://localhost:81> instead of <http://localhost>
* Everything works fine but I am not able to view the application.
  + Please double check that you have given read and writes permissions to assets, robot\_map\_data, robot\_atlas\_data, robot\_schedule\_data, robot\_custom\_data and the runtime folder. Without these permissions, application would not run.
  + Also make sure that you have given correct DB configurations in the main.php file.
* I see error page when I submit forgot password or change password forms.
  + Please check that you have given correct SMTP credentials in the configuration.
* I am not able to login with Facebook credentials.
  + Please check if you have given correct API/Secret key to the server.
  + Also checks if you have given correct call back URL in Facebook app developer form.
* How can I view Apache logs?
  + On Ubuntu, you can open **/var/logs/apache2/access.log** OR **/var/logs/apache2/error.log** files.
  + On Windows, you can check it by opening **C:\xampp\apache\logs\access.log** file OR **C:\xampp\apache\logs\error.log** file.
* All the web services calls are failing.
  + Please make sure that you have enabled the php\_curl lib in the php.ini as mentioned in the PHP setup sections for server and developer machine.
* My DB connection doesn’t seem to be working
  + Double check that the DB configuration is set in these 4 different files
    - **/config/main.php**
    - **/config/database\_config.php**
    - **/push\_notification\_standalone.php**
    - **/xmpp\_notification\_mq\_standalone.php**
* I want to change MySQL settings like the where the backup files are stored and so on:
  + On Linux, open **/etc/mysql/my.cnf** and make changes.
  + On Windows, open **C:\xampp\mysql\bin\my.conf** and make change.
  + As configuration changes would take effect only after a restart, restart the MySQL by:
    - On Windows, using XAMPP Control panel
    - On Linux, by giving **/etc/init.d/mysql restart**
* I cannot access the ejabberd admin, or calls using non-standard ports are failing. If you are behind a corporate firewall, please make sure you open the following ports: 53359, 5269, 5280, 5222, 5672, 47010.
* What GCM key should I use? The gcm\_api\_key is required to send Google Push Notifications.. You can either use the current gcm\_api\_key “AIzaSyAnczo1eXzLo6EdkWCa\_EYqi-PqLA2kdBA”. A better option would be to create a new key using the steps described in <http://developer.android.com/google/gcm/gs.html>
* Having trouble switching between Dev and Prod environments – Make sure the ‘env’ value ‘env’ => ‘Prod’ / ‘dev’ is updated in both push\_notification\_standalone.php  and xmpp\_notification\_mq\_standalone.php – both files should have the same value.
* AMQP Publisher path is not setup correctly – Please remember to setup the file link correctly - 'neato\_amqp\_publisher\_path' is actually a link to the file '/home/smartappuser/public\_html/php/amqplib/php-amqplib/demo/amqp\_publisher.php'.
* My ejabberd notifications are not working properly
  + Double check that the consumer and the publisher have access to following files. Copy these two files in the AMQP Lib folder
    - **push\_notification\_standalone.php**
    - **xmpp\_notification\_via\_mq\_standalone.php**
* I am getting an “unrecognized token error <” error when **sudo ejabberdctl send-message-chat** command is fired.
  + The PHP function escapeshellarg is being used and you need to specify it to use UTF-8. Add the following following setting in php.ini

**setlocale(LC\_CTYPE, "en\_US.UTF-8");**

End.