Stack360 Build Guide

Blake McBride

April 9, 2024

Contents

1 Introduction

Stack360 is comprehensive web-based business process software for service-oriented organizations.

Stack360 has been a reliable solution in production for over a decade, offering a suite of integrated functions tailored for service-oriented organizations. This comprehensive system streamlines various business operations, including:

- Applicant Tracking: Automate your hiring process.
- HR (Human Resources Management): Efficiently manage your workforce.
- Project Management: Keep your projects on track and within budget.
- Worker Time Tracking: Worker time flows to payroll and client billing.
- CRM (Customer Relationship Management): Manage sales prospects and clients.
- Client Invoicing: Streamline client billing.
- Employee Benefit Administration: Manage employee health benefits, insurance, time off, etc.
- and a lot more...

Stack360 is also described at stack360.io

This repo contains the entire back-end. Other repos have various front-ends.

Technical Specifications:

- Back-end
  - Java & Groovy
  - SQL Database
  - SOAP & REST Services
  - microservices
  - Deployment to cloud or in-house
  - Deployment to Linux or Windows servers
- Front-end
  - HTML/CSS
  - JavaScript

Stack360 also utilizes the KISS Open-source Web Development Framework kissweb.org

These build instructions work on:

- Linux
- Mac
- Windows

Documentation

In addition to the source code release, we are publishing a system manual that covers system architecture, acts as a developer guide, and covers system configuration. It is available on *Amazon* and is called the *Stack360 System Manual*.

Staying In Touch

By letting us know who you are, we can send you information on system and documentation updates. We do not share your email address or any other information with anyone.

To let us know who you are, please send us an email to info@stack360.io with the subject "SUBSCRIBE". You can unsubscribe by sending an email to the same address with the subject "UNSUBSCRIBE".

Support

Please help fund this project at `https://gofundme.com/stack360`

For those seeking commercial support or a commercial license, please don't hesitate to reach out to support@stack360.io . We're here to assist you in integrating Stack360 into your business seamlessly.

Information

A few bits of information -

This repository contains the whole back-end of the system. Various front-ends are being made available (under the same license) in sibling repositories.

Stack360 has some history that will be helpful for you to understand.

Development of Stack360 was started in 2006. It was using the best technology available at that time, including:

- Java SOAP web services
- SQL (we ran on PostgreSQL & Microsoft SQL server)
- Hibernate ORM

All communications between the back-end and front-end are through web services. The back-end never generates any front-end code.

The back-end typically runs on a Linux server either in-house or in the cloud. We primarily use Java 17 and PostgreSQL. However, the system has also been run on Microsoft servers and Microsoft SQL Server. We use no vendor-specific SQL features.

The front-end was originally written in Flash but re-written in modern HTML & JavaScript. The front-end only requires a standard browser. All common browsers are known to work.

During the rewrite to HTML, we added REST web services, microservices, and a new SQL API. All of this works fine along with our pre-existing code.

We eventually culled out and moved the generic parts of our system to create the Kiss Web Development framework (also open-source). So, Stack360 depends on the Kiss web development framework (see kissweb.org)

Stack360 has some historic names that you'll see floating around. Just know that they're all referring to the same system. These names include: Prophet, Arahant, Unifi, Stack360.

The way Stack360 is structured is a single back-end that supports multiple front-ends. Each front-end is seen by the web server as a separate system.

You'll need two main pieces that reside in separate repos including:

- Stack360–Backend
- Stack360–Frontend (desktop front-end)

There are several other pieces, such as a few mobile front-ends, a tablet front-end that works without an Internet connection. We will be releasing all of this as time permits.

The system will not run without a database. Demo and production databases scripts used to populate SQL databases are included with this release.

Credits

The Stack360 system was architected by Blake McBride and built with a team of developers. All documentation written by Blake McBride.

2 Overview

These are the quickest and shortest instructions needed to get the system up and running as fast as possible. It has been tested on Linux, Mac, and Windows.

The front-end and back-end are located in two different repos and two different directories on your machine. In a development environment, two separate servers run - one for the front-end and one for the back-end. (In a production environment, they run from the same server but as two different applications from the server's perspective.)

Through this process, you will have four windows as follows:

- A terminal that is used to build, start, and stop the back-end
- A terminal that will display the back-end log
- A terminal that is used to run the front-end server
- Your browser to access the system as a user would

The system mainly uses an SQL Database to store the data. However, there is a big exception. The system has the ability to store pictures in many areas within the system. For example, job applicants may upload their resume, picture of their driver's license, etc. Projects often have pictures associated with them to show diagrams or pictures of finished projects, etc. Over time, we experienced a major problem with the database's size growing in leaps and bounds. It quickly became unwieldy. We eventually moved all of those big uploads to external storage. Cloud vendors have S3 storage that is nearly limitless and far cheaper. There is a configuration in the system that tells the system where to store those files. (It does this intelligently so that you don't end up with too many files in a single directory.)

3 Build Setup

Please ensure the following requirements before starting the build procedure.

1. This system was tested with Java 17 JDK. Linux users can get this through their repo. Mac and Windows users can get it from https://jdk.java.net/archive
2. It is assumed you have a PostgreSQL database up and running. (Any version.) On Linux, you can get PostgreSQL from your repo. Mac and Windows users can get it at https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
3. The database password must be "postgres" for this demo. (You can configure another password later.) If it isn't, you can configure it as follows:

```
psql postgres postgres
alter user postgres with password 'postgres';
\q
```

4. You are starting with a fresh checkout of the repo
5. All of these command are run from the root directory of your repo clone.
6. On a Windows machine, you modify all the commands given by eliminating the leading "./" from each command.

7. You will need *groff* in order to generate many of the reports. Linux users can get the full version through your repo. Mac and Windows users should see https://www.gnu.org/software/groff On Windows you'll also need *ghostscript* installed.

4 Build Procedure

1. A few preliminary things to note:
   (a) Arguments to the "bld" command are case-sensitive. For example: "./bld configDemo" will work. "./bld configdemo" will not.
   (b) On Windows the leading "./" on all commands should not be used. In other words, on Linux or Mac you use "./bld configDemo". On Windows you would use "bld configDemo"
   (c) On Linux and Mac, you may need to ensure that some of the scripts are executable via: `chmod +x bld view-log`

2. The system requires a database. The system comes with a demo database script named `demo-data.sql.jar This file must be unjarred before it can be used. This can be done with the following command:`

   ```
   jar xf demo-data.sql.jar
   ```

   Once this is done, you will have the file *demo-data.sql*

3. We will start with the demo database, which has demo data so you can play with the system. Once you are ready, a blank production database script is provided.

   As configured, the system assumes the database name will be "demo" and the database password will be "postgres". This configuration can be changed.

   You'll need to load the demo database script into a database named *demo*. In PostgreSQL, you can do this with the following commands:

   ```
   psql postgres postgres      (if system asks, the password is postgres)
   create database demo;
   \c demo
   \i demo-data.sql
   \q
   ```

4. Stack360 has its own build system that makes things really easy. Build the system by typing the following command:

   ```
   ./bld configDemo       (Windows: bld configDemo)
   bld setupTomcat        ONLY NEEDED ON WINDOWS (one time)
   ./bld build            (Windows: bld build)
   ```

   That will build the entire system. On a fast machine, this takes less than a minute.

5. SKIP THIS STEP ON WINDOWS. The back-end server produces a log of things that are going on inside it. You can monitor this log file by starting a new terminal, going to the root directory of the back-end, and typing:

   ```
   ./view-log
   ```

6. Go back to the previous terminal and start the server with the following command:

```
./bld run              (Windows: bld run)
```

   You will see the other terminal that is displaying the log showing the progress of the server boot. It will also display any errors which you'll likely have to fix.

   Bringing up the server takes at least 30 seconds up to about a minute. You will know it is up when you see the last line contains the text "Server startup".

7. In a development environment (like what you're doing here), the back-end and front-end run on two separate servers. You need to start the front-end server.

   The front-end is located in a separate repo and should be cloned to a separate directory tree.

   There is no build procedure for the front-end. It runs as-is.

   Go into the root directory of the front-end and type the following:

```
./bld configStack360       (Windows: bld configStack360)
./serve                    (Windows: serve)
```

8. That's it. The entire system is up and running. Go to your browser and enter the following:

```
Url: http://localhost:8001

Username: demo
Password: password
```

   That's it. You're in.

9. The first time you run the system, you need to tell it where you want your external files stored. You do this by setting a system property on a running system. Once you log in, select:

   Administrative Functions / System Configuration / System Properties

   Look for *EXTERNAL_FILE_ROOT*

   Edit the property value to the path you'd like to use to store your external files.

10. The back-end can be debugged through port 9090 (generally using an IDE and attaching to the already running process). The front-end can be debugged using the browser's developer console or through your IDE. The whole build process, running, and debugging can be done through an IDE via the included `build.xml` *ant* file.

11. There are some things to know that may be helpful as follows:

    (a) The system has far more screens than what you can see. The demo configuration is just designed to give you a broad overview of the available functionality.

    (b) Each user or group of users can be defined a different hierarchy of screens they are presented with. This way, each user sees only the screens applicable to their job function. This is done in the following two places:

        Administrative Functions / System Configuration / Screens

Human Resources / Employees & Dependents / (select an employee) / Profile / Login

(c) The system also has an elaborate security system that can control what fields different users can see and change.

(d) As configured on the demo system, the "demo" user is the system administrator and not an employee. Therefore, there are some things you can't do with the demo user that can be done with regular users. For example, you can't assign a project to the demo user, edit their HR record, or apply billable time to projects.

12. When you're done, you need to stop the front-end server and stop the back-end server.

    The front-end server can be stopped by typing *Ctl-C* in the terminal that is running the `./serve` command.

    The back-end server can be stopped by going into the terminal that you used to start the back-end server and typing the following.

```
                    (Windows: ^C to exit the command file)
./bld stop          (Windows: bld stop)
```

    You can use *Ctl-C* on the terminal that is displaying your back-end log if you're done for the day. Or, it can be left running for future runs of the system.

5 Building A Production System

Once you are ready, you can switch to a blank database which you can use in a production environment. There are only two steps necessary in order to switch to a production system as follows:

1. Just like the demo database, you will require a production database. The production database name is "stack360". Create and load the production database script with the following commands:

```
psql postgres postgres          (if system asks, the password is postgres)
create database stack360;
\c stack360
\i stack360-data.sql
\q
```

2. Re-build the back-end to use the stack360 database with the following commands:

```
./bld configStack360            (Windows: bld configStack360)
bld setupTomcat                 ONLY NEEDED ON WINDOWS (one time)
./bld build                     (Windows: bld build)
```

There are no front-end changes. You use the same front-end as with the demo.

Once this is done, you can run the system as before (see steps starting at step 5 of the *Build Procedure*).

With the production database, there is only one user in the system:

```
Username: stack360
Password: stack360
```

Unfortunately, while you have everything at this point, the system is rather useless without a lot of configuration. All configuration can be done from within the system utilizing the configuration screens.

In addition to many internal details, the *Stack360 System Manual* we offer provides step-by-step information on system configuration.

6 Production Deployment

The stack360 database must exist on the server.

From the web server's perspective, the back-end and front-end are two separate applications.

The back-end is deployed as a single *war* file. After building the system, you can create this file with the following command:

```
./bld war           (Windows: bld war)
```

This will create a file named: `dist/Stack360Backend.war`

That is the only file you will need to deploy the back-end.

On the front-end you can create a file to deploy by running the command:

```
./makedist          (Windows: makedist)
```

That will create a file named `Stack360Frontend.jar` You will need to extract that file as a separate web application on the server.

You can deploy that file from the webapps directory (if using *tomcat*) by typing:

```
mkdir stack360
cd stack360
jar xvf ../Stack360Frontend.jar
```

After that, a couple of files on the front-end will need to be changed in order for the front-end to know where the back-end is.

1. File: `framework.js`
   (a) Uncomment and update the line containing:

   ```
   AWS.setURL('https://[YOUR-URL]/Stack360Backend');
   ```

2. File: `index.html`
   (a) Set *controlCache* to true.
   (b) Update *releaseDate* and *softwareVersion*. Their values are unimportant, except that the value assigned to *softwareVersion* must be unique each time you update the software.

7 More Information

We published a *Stack360 System Manual* through *Amazon*. This manual covers the system architecture, acts as a developer guide, and covers system configuration. Just look for the "*Stack360 System Manual*" on `https://www.amazon.com/dp/B0D1245Q1F`