Site Preparation for deployment

For a site to be deployed on a live server a number of things need to be considered and prepared.

Server Type

First and foremost it is important that the correct type of web server is selected. The eRevive site has been developed using PHP 7.0 on an Apache server, when selecting the live server it is important to select one which supports PHP 7.0.

It is also important to select a host which also supports a MySQL database, although this is not as important as the database could technically be hosted in a different location but for ease of set up it makes sense to use one host to provide both

Database

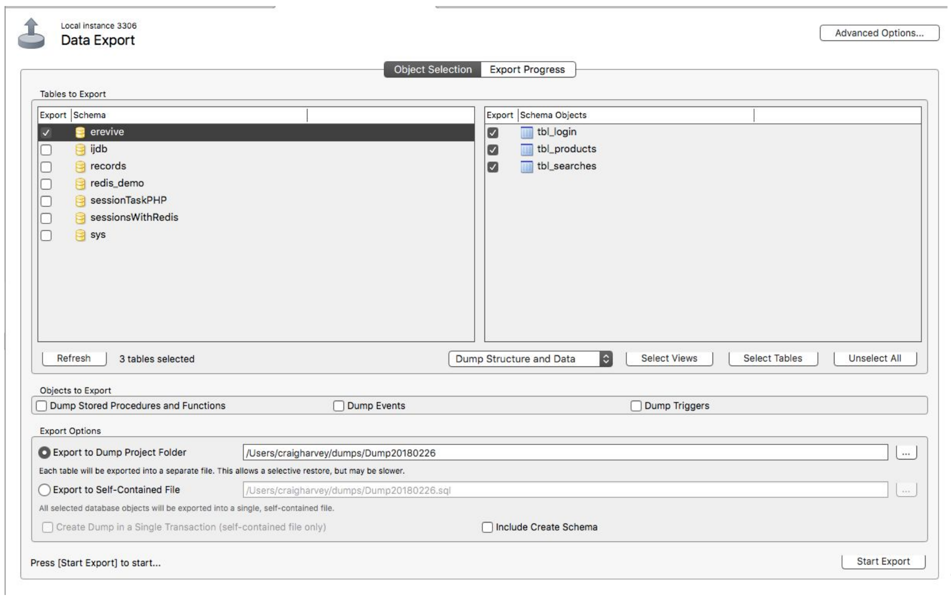
For the deployment of this site the we will need to have a database that is the same as the development database.

This can either be re-created by using the hosts GUI or SQL statements so that it matches the one used for development or the development databases structure can be exported and uploaded to the new database server.

By re-creating the database care must be taken to ensure that the correct database name is used, the correct name for each table is used, the correct column name is used and the correct data type is set for each column. This is a high risk strategy as there are multiple ways in which the database could be set up incorrectly resulting in an error on how the site is run. This method is not the recommended method

The best method would be export the current database structure so that it can be automatically created when imported on the deployment server.

This can be done by going to the MySQL GUI and using the export database feature. In my instance I've used MySQL Workbench , screenshot next page. This allows for the selection of multiple databases and tables for export.



Local instance 3306

Data Export

Tables to Export

Export	Schema
✓	erevive
☐	ijdb
☐	records
☐	redis_demo
☐	sessionTaskPHP
☐	sessionsWithRedis
☐	sys

Export	Schema Objects
✓	tbl_login
✓	tbl_products
✓	tbl_searches

3 tables selected

Dump Structure and Data

Objects to Export

☐ Dump Stored Procedures and Functions ☐ Dump Events ☐ Dump Triggers

Export Options

[✓ Export to Dump Project Folder] /Users/craigharvey/dumps/Dump20180226

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

Export to Self-Contained File /Users/craigharvey/dumps/Dump20180226.sql

All selected database objects will be exported into a single, self-contained file.

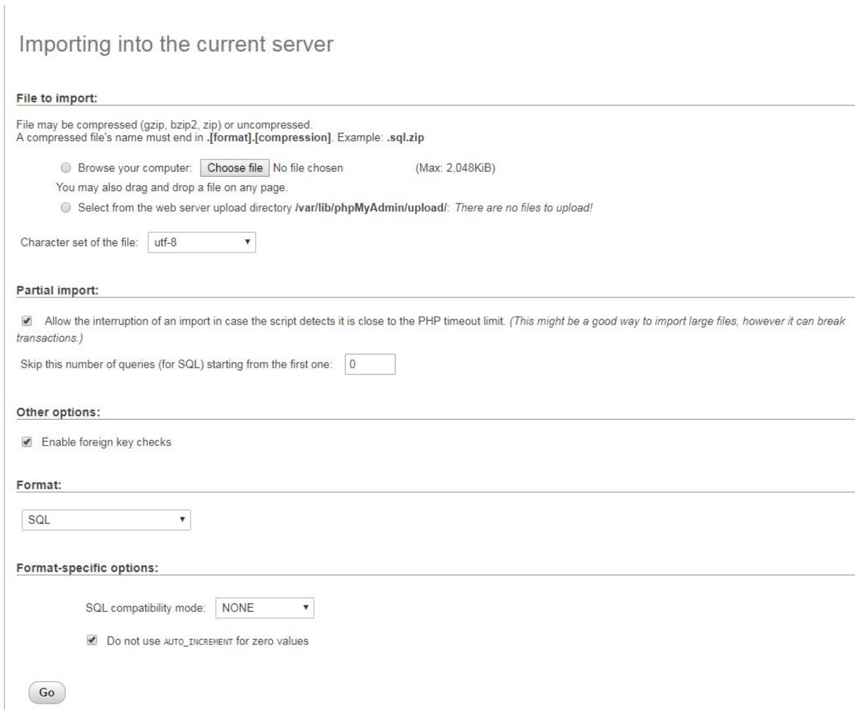
☐ Create Dump in a Single Transaction (self-contained file only) ☐ Include Create Schema

Press [Start Export] to start...

The database and the tables you with to export selected.

The export feature creates an SQL files which contain all the relevant SQL statements for the database to be recreated. If you open this file in a text editor the code can be read for the creation of different database or tables.

Once these files have been exported then can then be imported using the hosts GUI interface.



Importing into the current server

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: Choose file No file chosen (Max: 2,048KiB)

You may also drag and drop a file on any page.

Select from the web server upload directory **/var/lib/phpMyAdmin/upload/**: *There are no files to upload!*

Character set of the file: utf-8

Partial import:

✓ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files, however it can break transactions.)*

Skip this number of queries (for SQL) starting from the first one: 0

Other options:

✓ Enable foreign key checks

Format:

SQL

Format-specific options:

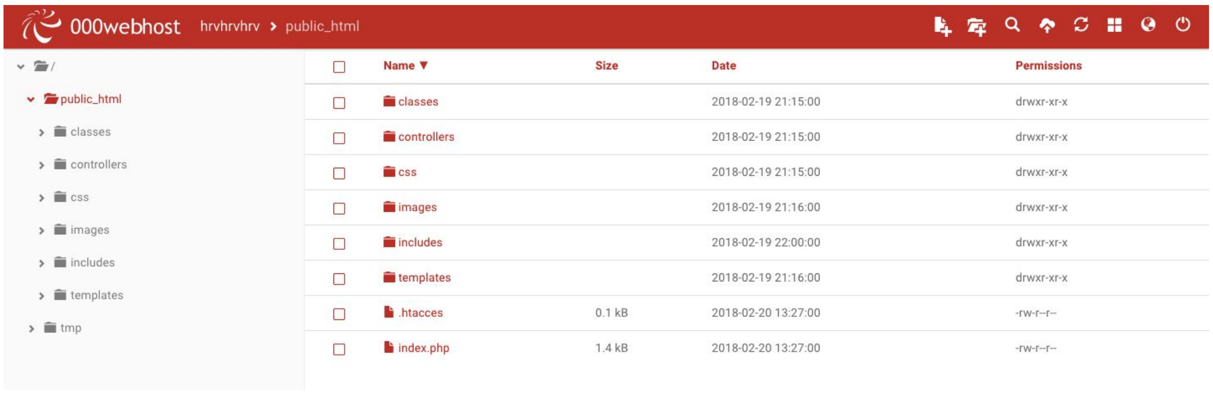
SQL compatibility mode: NONE

✓ Do not use AUTO_INCREMENT for zero values

This should automatically recreate recreate the same database on the hosts server.

Site Files

The files used for the site should then be uploaded to the host server. It is important that the same file structure is used when uploading the files, changes in the structure will cause the files to not behave correctly when calling pages from another page if the structure has changed as the files will not be where expected.



000webhost hrvhrvhrv › public_html

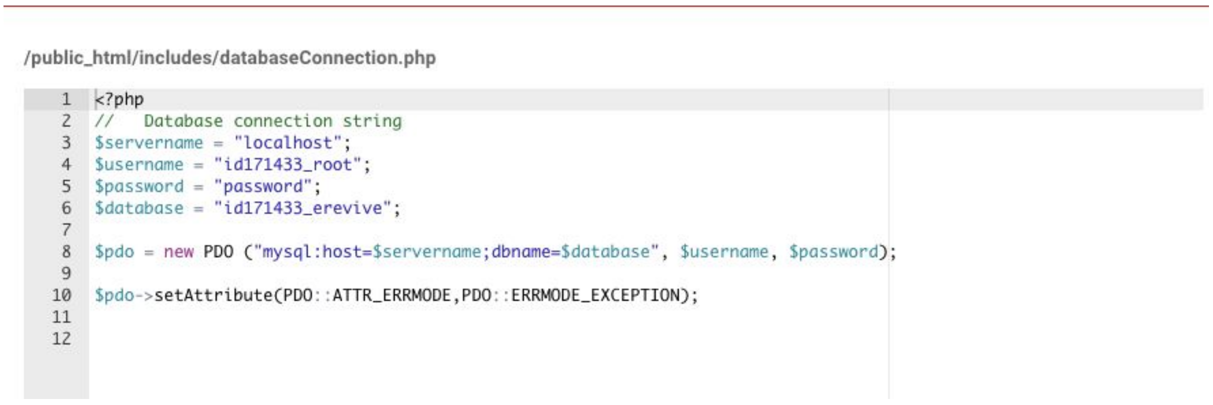
```
/
  public_html
    classes
    controllers
    css
    images
    includes
    templates
  tmp
```

Name ▼	Size	Date	Permissions
classes		2018-02-19 21:15:00	drwxr-xr-x
controllers		2018-02-19 21:15:00	drwxr-xr-x
css		2018-02-19 21:15:00	drwxr-xr-x
images		2018-02-19 21:16:00	drwxr-xr-x
includes		2018-02-19 22:00:00	drwxr-xr-x
templates		2018-02-19 21:16:00	drwxr-xr-x
.htacces	0.1 kB	2018-02-20 13:27:00	-rw-r--r--
index.php	1.4 kB	2018-02-20 13:27:00	-rw-r--r--

Connecting to the new database

Once the files are uploaded to the new server and the database has been correctly recreated you would expect the site to run correctly? Wrong.

Unless the database name, host and password are all exactly the same as the development server, the database connection string will need to be updated so that the website knows where to connect to the database. This information should be provided by the host when setting up the database.



/public_html/includes/databaseConnection.php

```php
<?php
//   Database connection string
$servername = "localhost";
$username = "id171433_root";
$password = "password";
$database = "id171433_erevive";

$pdo = new PDO ("mysql:host=$servername;dbname=$database", $username, $password);

$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


```

Once this has been updated to the details provided by the web host ensuring the correct database, user and server name are used and the correct database password is input the site should work.

Testing

Once the database and site files have been uploaded the next stage of site deployment is testing. This should be carried out on the site to ensure that all functionality is working correctly as the hosts server may have little quirks which may result in unexpected behavior. It is important to catch all of these quirks with testing so that when the site is launched live to the public that none of these errors are experienced by a sites user.