George Ciesinski

Humber College  205 Humber College Blvd., Toronto ON M9W 5L7

Drupal Websites

[Introduction 4](#_Toc141099781)

[Start Date 4](#_Toc141099782)

[Team Members 4](#_Toc141099783)

[Requirements 4](#_Toc141099784)

[Tools 4](#_Toc141099785)

[Humber Web Accessibility Compliance (AODA) 4](#_Toc141099786)

[Guidelines 4](#_Toc141099787)

[Humber Interim Web Guidelines 4](#_Toc141099788)

[Stack 5](#_Toc141099789)

[Humber ITS Stack 5](#_Toc141099790)

[Development 6](#_Toc141099791)

[Backups 6](#_Toc141099792)

[Complete Backup 6](#_Toc141099793)

[Git 7](#_Toc141099794)

[Requirements 7](#_Toc141099795)

[Git Flow 7](#_Toc141099796)

[Initializing a Repository 8](#_Toc141099797)

[Initial Commit Error 8](#_Toc141099798)

[Develop Branch 8](#_Toc141099799)

[More Information 9](#_Toc141099800)

[Backup Database 10](#_Toc141099801)

[Concept: Data backups 10](#_Toc141099802)

[Backup using command line 10](#_Toc141099803)

[pg\_dump 10](#_Toc141099804)

[PG\_restore 10](#_Toc141099805)

[Development Site 10](#_Toc141099806)

[Making a Development Site 10](#_Toc141099807)

[Maintenance 11](#_Toc141099808)

[Maintenance Mode 11](#_Toc141099809)

[Updating 11](#_Toc141099810)

[How to tell there are updates 11](#_Toc141099811)

[Should they be updated 11](#_Toc141099812)

[Updating Core Software 11](#_Toc141099813)

[Updating Modules 12](#_Toc141099814)

[Updating Theme 12](#_Toc141099815)

[Testing updates 12](#_Toc141099816)

[Settings.php 13](#_Toc141099817)

[Location of settings.php 13](#_Toc141099818)

[Making changes to settings.php 13](#_Toc141099819)

[Frequently Required Changes 13](#_Toc141099820)

[Trusted Host Settings 13](#_Toc141099821)

[Update Free access 14](#_Toc141099822)

[Roles 15](#_Toc141099823)

[Users 15](#_Toc141099824)

[Configuring User Account Settings 15](#_Toc141099825)

[User 1 15](#_Toc141099826)

[Anonymous Users 15](#_Toc141099827)

[Authenticated Users 15](#_Toc141099828)

[Additional Roles 15](#_Toc141099829)

[Creating a new role 15](#_Toc141099830)

[Assigning permissions to a role 16](#_Toc141099831)

[Changing a user’s role 16](#_Toc141099832)

[Modules 17](#_Toc141099833)

[Installation & Maintenance 17](#_Toc141099834)

[Installing Modules 17](#_Toc141099835)

[Using Composer to Manage Dependencies 17](#_Toc141099836)

[Considerations before updating 17](#_Toc141099837)

[Shared Modules 18](#_Toc141099838)

[ADMIN TOOLBAR 18](#_Toc141099839)

[R4032 Login 18](#_Toc141099840)

[PathAuto 18](#_Toc141099841)

[LDAP / Active directory integration 18](#_Toc141099842)

[Gin Theme 18](#_Toc141099843)

[Gin Admin Theme 18](#_Toc141099844)

[Gin Login 18](#_Toc141099845)

[Gin Toolbar 19](#_Toc141099846)

[Useful Modules 20](#_Toc141099847)

[Album Photos 20](#_Toc141099848)

[Video 20](#_Toc141099849)

[Field Group 20](#_Toc141099850)

[Field Permissions 20](#_Toc141099851)

[Delete Unused Files 21](#_Toc141099852)

[Administration 22](#_Toc141099853)

[One time login 22](#_Toc141099854)

[Additional Info 23](#_Toc141099855)

[Documentation 23](#_Toc141099856)

[Glossary 23](#_Toc141099857)

[Further Reading 24](#_Toc141099858)

[React for Drupal 24](#_Toc141099859)

# Introduction

This document contains information that is shared between all the Humber ITS Websites built using Drupal.

This includes information about settings, core and module updates, backing up and restoring sites, content updates, or possible bugs.

## Start Date

The design & build process for the ITS websites started on April 2023.

## Team Members

George Ciesinski – Web Developer

Michael Boadu – Marketing and Communications Coordinator

## Requirements

The ITS website serves tens of thousands of Faculty Members, Staff and Students in a College environment. The below requirements will help to provide a positive and secure experience to all website users:

1. **Security** – The website should stay up to date with major security updates, and should include authentication for all users who modify the content of the site.
2. **Optimization** – All images, videos, and other assets should be optimized before usage on the site to ensure a fast browsing experience.
3. **Accessibility** – Content should meet the accessibility requirements outlined in the AODA (Accessibility for Ontarians with Disabilities Act)

## Tools

### Humber Web Accessibility Compliance (AODA)

This page provides access to the WCAG Quick-Reference guide and AODA Compliance Reference, as well as various tools that assist in evaluating Web Accessibility.

<https://humber.ca/tutorial/web-accessibility-compliance.html>

## Guidelines

### Humber Interim Web Guidelines

This document provides guidelines for standards Humber Websites should meet.

<https://humber.ca/brand/sites/default/files/publications/interim-web-guidelines.pdf>

# Stack

Drupal is based on the LAMP stack: Linux, Apache, MySQL, and PHP Scripting Language.

Drupal can also run on other technology stacks. For example:

* Windows, Mac or Linux
* Apache, Nginx or IIS
* PostgreSQL, SQLite, or MySQL

## Humber ITS Stack

The Humber ITS Stack uses:

* Linux OS
* Apache Server
* PostgreSQL
* PHP

# Development

## Backups

It is important to back up the website whenever there is a core/module update, or when significant amounts of content are added. This ensures that the site can be restored in case of any unforeseen event. Additionally, this allows us to migrate the site from one server to another if needed.

Backups are very important and should be tested to ensure it is possible to restore the site from the backups.

### Complete Backup

In order to make a complete backup, several parts of the Drupal site must be backed up separately.

1. Source Code / Web Root

The Drupal root folder must be backed up with Git. It is important to note that a Git backup is incomplete and doesn’t backup the settings.php file or the database.

### Git

Git is a kind of version control used by the Humber ITS team to back up Drupal site directories. This is not a complete back-up of the site, but instead contains changes to source code and some configurations for the site.

When using these docs on an existing site, some parts of this section can be ignored. For example, you may not need to know how to initialize a new repository for an existing site, but you should always create branches for new major features and commit your changes regularly.   
  
You can learn more about managing the repository using Git:

<https://www.drupal.org/docs/user_guide/en/extend-git.html>

#### Requirements

Some level of **Git knowledge** is required in order to follow this section. Version control is vital for modern web and software development and is used globally throughout the world. Git is currently the most used version control standard.

To learn more about Git, check out the Atlassian git documents: <https://www.atlassian.com/git/tutorials>

#### Git Flow

It is also important to understand git-flow. Atlassian has a great article about it here:

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

In short, git normally starts off with one single main branch. Generally, we branch off from the main branch with a new develop branch. Typically all development happens on this branch, or its children. Once it is tested and verified to be working, it is merged back into main. However, when there are more than a single developer, then it is recommended to use children branches.

Children branches can be anything, although gitflow recommends you use feature, release, and hotfix branches. The only difference in these branches is the name. When creating a feature branch, you would use a command such as:

git checkout -b feature/create\_navbar

If you need to create a release or hotfix branch, simply put those in place of feature in the command above. From there, you would carry out the development, and merge those changes back into develop when they are completed and tested.

Each of these children branches have their own purpose:

**Feature:** This is a new feature. It can be a new element, new logic, or any other code that results in a new feature being added to the project.

**Hotfix:** This is essentially a bug fix. A hotfix branch would be created specifically to address a bug or issue. Like the feature branch, the changes can later be merged back into develop.

**Release:** Unlike Feature or hotfix, a release branch is typically a child of develop and is merged into main. The purpose of the release branch is to do final testing and last-minute changes before creating a tagged release on the main branch.

Following git-flow ensures that development proceeds smoothly without loss of work or data, and allows multiple developers to collaborate safely.

#### Initializing a Repository

1. Create a repository on Github
2. Using Terminal, cd into the project directory
3. Create a .gitignore file if one doesn’t exist:

touch .gitignore

1. Add the following to the .gitignore file using a text editor like Vim:

web/sites/\*/settings\*.php

web/sites/\*/files

web/config

1. Save the config file, and enter the following commands in Terminal:

git init

git add -A

**Note:** The above commands initialize the repository and add all of the files in the directory to the initial commit.

1. Commit and push the initial commit to the repository by typing the below commands into Terminal:

git commit -m "Initial file add"

git remote add origin (ssh git link)

git push -u origin main

##### Initial Commit Error

If you are committing to a newly created repository or a repository with a readme, you may get an error that the push was rejected:

ciesinsg@NB-00304-H14872 its-site % git push -u origin main

To github.com:Humber-ITS/ITS-site.git

! [rejected] main -> main (fetch first)

error: failed to push some refs to 'github.com:Humber-ITS/ITS-site.git'

hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

You can use the below command to resolve it:

git push -f origin main

**Note:** It is only safe to use this command on initial commit

#### Develop Branch

Generally, it is good to have at least a main branch and a develop branch. This allows you to make changes on the develop branch and to keep only working and tagged commits on the main branch.

On a new repository, you can create a develop branch with:

git checkout -b develop

You can push the newly created branch up to origin/develop with the below:

git push --set-upstream origin develop

After pushing the branch up, the branch can subsequently be updated with a regular git push.

#### More Information

Some more information can be found in these links. These are meant to get a better understanding of how Git can be used on a Drupal site, as well as best practices and the Git-flow method of creating branches. It should not be used in place of the other instructions in this document.

<https://www.drupal.org/docs/installing-drupal/building-a-drupal-site-with-git>

<https://www.drupal.org/docs/user_guide/en/extend-git.html>

<https://www.drupal.org/docs/installing-drupal/building-a-drupal-site-with-git>

<https://nvie.com/posts/a-successful-git-branching-model/>

### Backup Database

#### Concept: Data backups

<https://www.drupal.org/docs/user_guide/en/prevent-backups.html>

#### Backup using command line

<https://www.drupal.org/docs/7/backing-up-and-migrating-a-site/back-up-your-site-using-the-command-line>

##### pg\_dump

<https://www.postgresql.org/docs/7.1/app-pgdump.html>

##### PG\_restore

<https://www.postgresql.org/docs/7.1/app-pgrestore.html>

### Development Site

It is not recommended to carry out core and module updates on the live/production version of the site because it could introduce breaking changes that are difficult to reverse. It may also result in an unexpected outage. Instead, it is recommended to set up a local development site which is essentially a clone of the live/production server. The process of setting up a local development site can also be used to verify the integrity of the backup files.

This development site is used for updates and major changes as well as for testing. Once the testing is complete, the changes can be pushed to the live site.

#### Making a Development Site

<https://www.drupal.org/docs/user_guide/en/install-dev-making.html>

## Maintenance

### Maintenance Mode

Presents message that the site is under maintenance to users who do not have the right permissions. Authorized users can maintain the site and update content during this time.

<https://www.drupal.org/docs/user_guide/en/extend-maintenance.html>

### Updating

Drupal core and the various modules the site uses are frequently updated to address security issues and bugs that come up over time. In many cases, it is beneficial to update these components to ensure that the Humber site is secure and working correctly.

#### How to tell there are updates

Updates can be checked using the Admin toolbar on the website by visiting the Available Updates section.

Reports > Available Updates

This will show a screen like this which shows if Drupal core and modules are up to date:

A screenshot of a computer

Description automatically generated

If any of these components do not show Up to date and a green check mark, those components may have updates.

##### Should they be updated

Updates should be made if they address a security flaw or a bug that may affect the website. You can check this by visiting the pages of the Drupal core version or the module project pages and reading the release notes. Any significant changes will be listed there, and a decision can be made based on this information.

#### Updating Core Software

<https://www.drupal.org/docs/user_guide/en/install-composer.html>

#### Updating Modules

<https://www.drupal.org/docs/user_guide/en/security-update-module.html>

#### Updating Theme

<https://www.drupal.org/docs/user_guide/en/security-update-theme.html>

#### Testing updates

Updates should be made on the development website and tested thoroughly before pushing to production. This would help with catching any incompatibilities or issues that may come up from the updates.

## Settings.php

The settings.php file contains sensitive data about the website such as the database connection, trusted host, and configuration information.

This file should NOT be uploaded to Github. Instead, it should be backed up in a secure location.

The file is also read-only by default which is required to ensure the site is secure, however, there are many instances where the administrator may have to edit this file. The instructions to do this are below.

### Location of settings.php

The settings file can be found in: /project-folder/web/sites/default/

### Making changes to settings.php

The settings.php file is read-only by default which is a security precaution to ensure the site is not able to alter it in any way. In order to make changes, you must locally change the permission of the file to give yourself write access, make the required changes, and once again harden the permissions. This ensures that the file contents are only changed in an authorized way and not by malicious actors.

1. Open the terminal and cd into the above directory.
2. Use the below command to make the file editable:

chmod a+w settings.php

1. Make the required changes using a text editor like vim or nano.
2. Harden the permissions after editing the file:

Chmod 444 settings.php (Results in permissions -r-r--r--)

Alternative command *(not recommended as it gives permissions -rw-r--r--)*:

chmod go-w settings.php

### Frequently Required Changes

#### Trusted Host Settings

The trusted host settings tell the website which hosts are authorized to access the site. This is an important security measure to ensure that fraudulent hosts cannot be used to access the site or create unauthorized clones.

This setting essentially tells the site which URL is allowed to access the site, and it affects both the production/live version of the site and the local development version. In the event that the host name changes, the trusted host settings must be modified to reflect this.

The trusted host setting looks like this in settings.php:

$settings['trusted\_host\_patterns'] = [

'^www\.its-cab\.test$',

'^its-cab\.test$'

];

This is an array containing multiple comma-separated lines with the various host patterns. The standard is to indent each line with two spaces. The patterns themselves are defined using REGEX.

#### Update Free access

This setting is used during core updates.

Todo

## Roles

### Users

Users are accounts with usernames and passwords. Drupal sites come with a User 1 account and all other users must be created by the admin, or by new users requesting accounts on the site.

More information about users can be found here:

<https://www.drupal.org/docs/user_guide/en/user-concept.html>

#### Configuring User Account Settings

It is possible to configure many aspects of user accounts including whether users should be able to request/open their own accounts, or whether an admin has to manually do this step.

More information can be found here:

<https://www.drupal.org/docs/user_guide/en/config-user.html>

### User 1

The User 1 account is the first account created along with the site and is known as the diety account in Drupal. This is because it has significant administrative powers and can be used to create other admin accounts. It also has the power to perform any action on the site regardless of permissions.

### Anonymous Users

Anonymous users are any users who visit the site without logging in. This is defined as a role and the permissions for these users can be separately configured.

### Authenticated Users

Authenticated users are any users who have accounts and can sign in.

### Additional Roles

Various roles can be created on the site. These roles can be named anything you want, and the permissions for each role can be customized in many ways. For example, it is possible what content an account can see, edit, create and delete. There is much more to the permissions system, so to get a thorough understanding of it, it is necessary to dig into the official Drupal docs.

The Drupal User Guide Chapter 7 discusses Managing User Accounts in more detail:

<https://www.drupal.org/docs/user_guide/en/user-chapter.html>

#### Creating a new role

A new role can be created by following the instructions here:

<https://www.drupal.org/docs/user_guide/en/user-new-role.html>

#### Assigning permissions to a role

Once a role is created, the permissions must be defined. It can be helpful to look at other roles’ permissions to see how they are set up, and create the permissions based off this. In some cases, you may need to dig deeper about which permission to add to a role in case you are trying to achieve a specific purpose.

**Note:** It is NOT recommended to add a permission to a role if it will only benefit a single user. This would grant that permission to every user who is assigned to that role. In that case, it would be beneficial to create a new unique role for that user.

The instructions to assign a permission to a role can be found here:

<https://www.drupal.org/docs/user_guide/en/user-permissions.html>

#### Changing a user’s role

Roles must be assigned to new users so they can enjoy the permissions of that role. Users with no assigned roles are simply authenticated users. In order to assign special permissions to those users, it is recommended to assign a role to them and not to modify the permissions of authenticated users.

To change a user’s role, follow the instructions here:

<https://www.drupal.org/docs/user_guide/en/user-roles.html>

## Modules

Modules extend functionality and add features. Installing the feature adds functionality, while uninstalling it removes it. Modules increase the time needed to generate a page, so it is important to only keep modules you use installed and remove the unused ones.

In general, modules will not be added to sites frequently. They will only be added to create a new feature that may require a contributed module in order to function correctly.   
  
Most actions we perform on modules will be maintenance as new versions are released.

### Installation & Maintenance

#### Installing Modules

Installing modules is well documented in the Drupal User Guide:

<https://www.drupal.org/docs/extending-drupal/installing-modules>

#### Using Composer to Manage Dependencies

All the Humber ITS Websites are builit with Composer. This makes adding and updating modules very simple. You can read more about it in the Drupal User Guide:

<https://www.drupal.org/docs/develop/using-composer/manage-dependencies#adding-modules>

#### Considerations before updating

The website might flag some modules that are outdated, but care should be taken before updating any module. It is important to go over the various modules in use on the site and check the module Drupal pages to see if there are any special instructions for carrying out updates.

Furthermore, it is important to make sure we are using the correct module for the core version. If using Drupal 10, only modules compatible with Drupal 10 should be used. If a new version of a module is built for Drupal 11, the core files should be updated first.

I will go into further detail about maintenance further in this document.

### Shared Modules

Shared Modules are any modules that are used across most or all of the Humber ITS Drupal sites.

#### ADMIN TOOLBAR

Improves the default toolbar and changes it into a drop-down menu with fast access to each administrative page.

In the Extend menu, the following options are also enabled:

* Admin Toolbar Content
* Admin Toolbar Extra Tools
* Admin Toolbar Search

<https://www.drupal.org/project/admin_toolbar>

#### R4032 Login

This module is used to force users to Login when they attempt to access pages or content that they do not have the permission to see. This is mainly used to force Anonymous users to sign in.

<https://www.drupal.org/project/r4032login>

#### PathAuto

This module is used to automatically generate unique URL’s for new content. Default Drupal behavior results in URL snippets like node/1, but PathAuto can be used to create templates so that URLs might instead look like photo/2 or change-request/143.

<https://www.drupal.org/project/pathauto>

<https://ostraining.com/blog/drupal/pathauto-2/> (Instructions)

#### LDAP / Active directory integration

This module adds Active Directory integration to sites so that this can be used as an alternative sign-in method.

<https://www.drupal.org/project/ldap_auth>

& Instructions: <https://www.drupal.org/docs/contributed-modules/ldap-integration/ldap-user-role-mapping>

#### Gin Theme

A number of Gin theme modules are also installed.

##### Gin Admin Theme

<https://www.drupal.org/project/gin>

Beautiful and sleep admin theme.

##### Gin Login

<https://www.drupal.org/project/gin_login>

Gin Login provides a beautiful login page with a splashdown image.

##### Gin Toolbar

<https://www.drupal.org/project/gin_toolbar>

Helper module that brings Gin theme to the admin toolbar.

### Useful Modules

Useful Modules are any Modules that provide useful features but are not yet used on the Humber site. These can be considered in future updates if it is deemed that they are necessary for a prospective feature.

**TODO:** Add modules from [Building a Drupal site with Git | Installing Drupal | Drupal Wiki guide on Drupal.org](https://www.drupal.org/docs/installing-drupal/building-a-drupal-site-with-git)

#### Album Photos

Needs to be compared to core photos module to see if this is necessary or not.

<https://www.drupal.org/project/photos>

#### Video

Allows you to host and play videos on the platform.

<https://www.drupal.org/project/video>

#### Field Group

Allows you to group fields so that they are easier to organize.

<https://www.drupal.org/project/field_group>

#### Field Permissions

Provides field level permissions for different roles.

<https://www.drupal.org/project/field_permissions>

#### Delete Unused Files

This module allows unused files to be safely deleted. Default Drupal behavior archives the files so that a copy always exists on the site.

<https://www.drupal.org/project/delete_unused_files>

# Administration

## One time login

**Note:** drush is required in order to follow these steps.

If a user is unable to login for some reason, it is possible to send them a one-time login using drush uli:

drush uli --name=username --uri=website-url

If successful, it will output a URL that the user can use to login and change their password.

# Additional Info

## Documentation

<https://www.drupal.org/documentation>

## Glossary

<https://www.drupal.org/docs/user_guide/en/glossary.html>

# Further Reading

## React for Drupal

<https://reactfordrupal.com/tutorials/react-for-drupal-developers>