The James Hutton Institute
Information & Computational Sciences

Germinate 3

Documentation

Version 3.4.0

March, 2017

Developers:

Paul Shaw
Sebastian Raubach
Iain Milne
Gordon Stephen
David Marshall

Contents

1 Introduction

Germinate 3 is a generic platform for the storage and dissemination of multiple data types associated with genetic resource collections. Examples of the types of data that Germinate 3 currently supports are passport, phenotypic, field trial, pedigree, genetic and geographic location data. Our aim is to keep Germinate as flexible as possible and we will add support for additional data types over time.

Germinate not only acts as storage for experimental data but offers a user-friendly interface into the data and acts as a backend for analysis tools such as Helium for pedigree visualization, our graphical genotyping application Flapjack [1] and CurlyWhirly for the simple display of xyz coordinate data such as PCO and PCA. All these tools are available from `https://ics.hutton.ac.uk`. We have also prioritised the development of tools to allow users to export data in a variety of formats for analysis in external applications such as R.

Germinate 3 builds on the existing Germinate 2 platform an adds additional tools, functionality and introduces a much more simple installation process over its predecessor. These changes also allow us to deploy Germinate both within server and desktop environments which was difficult with Germinate 2. We have also removed the limitation of requiring Linux, Germinate 3 is now compatible with any environments that have Apache Tomcat. Germinate 3 takes the Germinate platform away from its Perl roots and has been completely rewritten using the Google Web Toolkit (GWT) from Google. The underlying database is as it was and while we recommend using MySQL should be compatible with a number of relational database management systems with minor changes.

The move from Perl to Java and GWT has allowed us to introduce a number of new features. Examples include full internationalization and localization support which allows us to offer Germinate in multiple languages if suitable translations exit as well as offering a more advanced and responsive web-interface and user access control.

We hope that you find Germinate 3 useful and our vision is that Germinate forms platform on to which additional tools can be added over time. The common platform means that any additional functionality can be rolled out to all other Germinate installations which makes it both a flexible and continually evolving platform to help meet the needs of the genetic resources community.

If you encounter any problems, have ideas for features that would be useful to include in Germinate or just want to chat about the system then we would love to hear from you and we can be contacted in a number of ways. By email on `germinate@hutton.ac.uk`, or you can write to us at:

Germinate c/o Paul Shaw,
Information and Computational Sciences,
The James Hutton Institute,
Invergowrie,
Dundee,
DD2 5DA, UK.

We also have a website (`https://ics.hutton.ac.uk`) that we keep up to date with current developments of Germinate and all our other analysis and visualization tools and you can follow us on Twitter `@cropgeeks`.

2 Setup

In this section, we will explain how to get Germinate 3 up and running. First, we will explain the requirements and the necessary steps to deploy a self contained WAR (**W**eb application **AR**chive) file. The second part contains a detailed description for the advanced user, where we explain how to deploy the application from the source code.

2.1 Requirements

We try to keep the requirements of Germinate 3 as basic as possible. In order to run Germinate 3 you will need to have the following applications available on your server:

- *Apache Tomcat* (7.0.28 or above) to run the web application

- *Java* (8 or above) to run Apache Tomcat and the Germinate 3 server side code

- *MySQL* (or *MariaDB*) (5.6.1 or above) database to hold the data
  (if you're using 5.7.5 or above, make sure it's not set to use `ONLY_FULL_GROUP_BY`).

If you are planning to compile Germinate 3 from source (required if you do not have `.war` files) you will also need to have the following applications available on the system you're compiling Germinate 3 on:

- *Apache Ant* (1.9.1 or above) to compile to `.war` files

Please make sure that you have the required applications installed before continuing.

2.2 Configuration of Apache Tomcat

If you already have a Tomcat user account, skip this step.

Once Apache Tomcat is installed, you will need to create a user account. This user account will be used to deploy Germinate 3 to Tomcat. To add a user account, open the file `<Tomcat Directory>/conf/tomcat-users.xml` and add the following line:

```
<user username="<Username>" password="<Password>" roles="manager-gui,manager-script"/>
```

Just replace the username and password placeholders with the credentials you want to use. Tomcat needs to be restarted after these two steps.

Remember the username and password, we will need them later.

2.3 Configuration of MySQL

If you already have a MySQL user account that has permissions to query the Germinate 3 and Germinate Gatekeeper databases, skip this step.

After the installation of MySQL, you need to create a user account for Germinate 3, so that the web application can query the database. Remember the username and password, we will need them later.

Once the user is created, you can grant permissions to the newly created user. It is save to grant this user all available permissions for the Germinate 3 and Germinate Gatekeeper database. The minimal set of required permissions are: `select`, `insert`, `delete`, `update`, `create view`, `create routine` and `execute`.
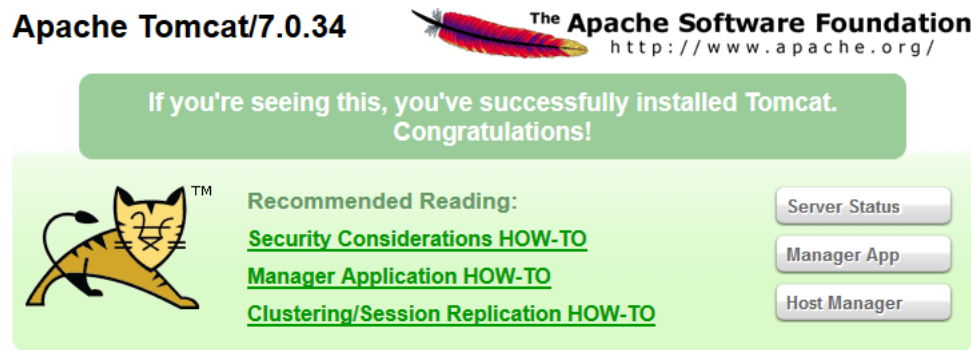
Figure 1: Tomcat welcome screen

2.4 Basic setup

In the best case, you already have a compiled version of Germinate 3 and Germinate Gatekeeper in the form of a `.war` file each as well as their database counterparts in your MySQL installation. To deploy these applications, navigate to the following URL:

<div align="center">

`http://<Your web server>:8080`

</div>

You should see something similar to Figure 1. Click on the button labelled "Manager App" and you will be prompted to enter your credentials. Use the username and password defined in the file `tomcat-users.xml`. The following page will show all the applications running on Tomcat right now. Below this table, there is a section called "Deploy" with subsection "WAR file to deploy". Simply select each of the two WAR files by clicking on the "Browse" button and then click on "Deploy". Germinate 3 and Gatekeeper should now be listed in the applications table at the top of the page.

Clicking on the `Path` link in the first column of the overview table should redirect you to Germinate 3 and Germinate Gatekeeper respectively.

2.5 Advanced setup

In this section, we will explain how you compile Germinate 3 (Germinate Gatekeeper analogously) from source. This is required if you do not have pre-compiled versions of the applications. We will also explain how you can create empty versions of the Germinate 3 and Germinate Gatekeeper databases in your database server.

2.5.1 Download of source code

First, download the Germinate 3 and Germinate Gatekeeper source code from our SVN:

- Germinate 3

    - `https://ics.hutton.ac.uk/svn/germinate3`

- Germinate Gatekeeper

    - `https://ics.hutton.ac.uk/svn/germinate-gatekeeper`

2.5.2 Configuration of source code

Both applications contain a file called `build.xml` which is the Apache Ant build script as well as an associated `build.properties` file. These files are used to compile and deploy the application.

Edit the `build.properties` file and replace the placeholder username and password with your Apache Tomcat username and password. Replace the placeholder for your web server as well.

```
# Ant properties for building the GWT app
project.name=germinate-template
project.root=jhi.germinate.Germinate

instance.files=./instance-stuff/template

tomcat.manager.url=http://<Your web server>:8080/manager/text
tomcat.manager.username=<Username>
tomcat.manager.password=<Password>
```

The next file that needs editing is the `config.properties` file. In the case of Germinate Gatekeeper this file is located in the root directory. In the case of Germinate 3, you can find this file in the `instance-stuff/<your germinate instance>` folder.

Configure Germinate Gatekeeper in the following way:

```
database.server=<server holding germinate>
database.name=<database name>
database.useport=<is a port necessary?>
database.port=<port number>
database.username=<username, e.g. germinate3>
database.password=<password>
[...]
```

Finally, configure Germinate 3 in the following way:

```
Germinate.Database.Server=<server holding germinate>
Germinate.Database.Name=<database name>
Germinate.Database.UsePort=<is a port necessary?>
Germinate.Database.Port=<port number>
Germinate.Database.Username=<username, e.g. germinate3>
Germinate.Database.Password=<password>

Gatekeeper.URL=<base url of gatekeeper>
Gatekeeper.Database.Name=<name of gatekeeper database>
Gatekeeper.Database.Server=<server holding gatekeeper>
Gatekeeper.Database.UsePort=<is a port necessary?>
Gatekeeper.Database.Port=<port number>
[...]
```

2.5.3 Configuration of the database
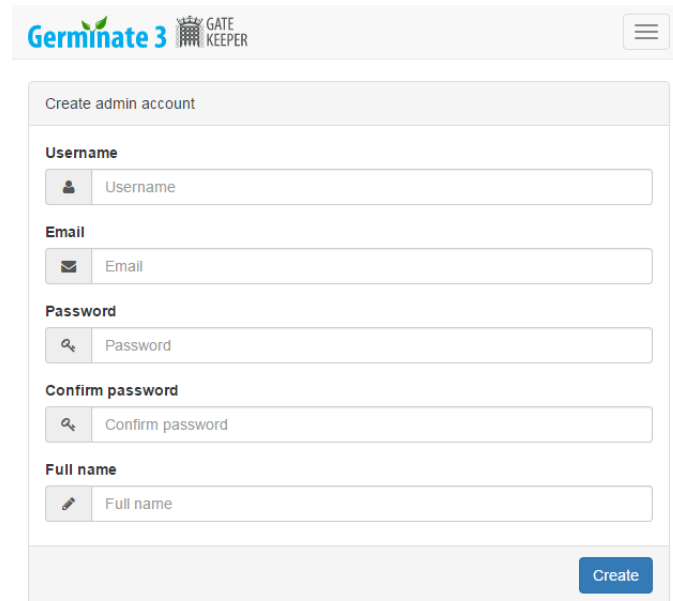
We've explained how to create a MySQL user account in Section 2.3. Now we will explain how to create empty versions of the Germinate 3 and Germinate Gatekeeper databases on your server.

Both application source folders contain a sub-folder called "database". This folder contains a `.sql` file that can be used to create an empty database (apart from default and required entries) for Germinate 3 and Germinate Gatekeeper respectively. Before you execute the scripts, make sure that you have an empty database per application that you can run the script against.

After running the two scripts, you should now have two template databases on your server. Grant your Germinate user access to those two databases. Make sure to grant at least the permissions listed in Section 2.3.

2.5.4 Building the source code

To build the application, simply run `ant` from the command line in the root directory of Germinate 3 and Germinate Gatekeeper. Apache Ant will then run the build script and compile the source code and finally deploy it to your Apache Tomcat installation.

Figure 2: Create admin account page

Make sure that the machine you're compiling the source on can communicate with the web server via HTTP (required to deploy the application).

After the build finishes successfully, you should be able to view Germinate in your browser. The address is based on your configuration. It should have this structure: `http://<Your web server>:8080/<project.name>`.

2.5.4.1 Making code changes

You are now free to change the code and add new pages. For examples of how to add new pages and other handy code snippets see Section 6.

2.6 Germinate Gatekeeper Admin

The first time you go to the Germinate Gatekeeper website, you'll see the form shown in Figure 2. Fill it in to create the initial admin account. After this, you will be able to log in to Gatekeeper and create other users, database systems and set their permissions.

2.7 Logging

Depending on your configuration (ref. `Germinate.Server.Logging.Enabled` in Section 3), Germinate will log all server-side exceptions. This can be useful when you need to debug your version of Germinate.

The log files are stored to this location: `<Tomcat Directory>/temp/logs/`. Each instance of Germinate uses its own log files to avoid mix-ups.

2.8 Docker

Since version 3.3.1, a Germinate Demo Application is available as a Docker image. Please consult our website for more details:

> `https://ics.hutton.ac.uk/germinate/germinate-docker-image`

3   Configuration

This section will highlight some of the configurations of Germinate 3. The configurations are stored in the file `instance-stuff/<project.name>/config.properties`.

Currently the following settings are available:

```
Germinate.Database.Username=<username for the database>
Germinate.Database.Password=<password for the database

Germinate.Database.Server=<server of the germinate database>
Germinate.Database.Name=<name of the germinate database>
Germinate.Database.UsePort=<use a port for the connection?>
Germinate.Database.Port=<port for database connection>

Gatekeeper.URL=<base url of gatekeeper>
Gatekeeper.Database.Server=<server of the gatekeeper database>
Gatekeeper.Database.Name=<name of the gatekeeper database>
Gatekeeper.Database.UsePort=<use a port for the connection?>
Gatekeeper.Database.Port=<port for database connection>

Gatekeeper.BCrypt.Rounds=<number of rounds for the bcrypt hashing algorithm>
Gatekeeper.Registration.Enabled=<allow user registration?>
Gatekeeper.Registration.Needs.Approval=<user registration needs manual approval?>

Germinate.UseAuthentication=<users have to log in to see data?>
Germinate.CookieLifespanMinutes=<the lifespan of cookies in minutes>
Germinate.Debug=<enable to see sql queries run on the server>
Germinate.KeepTemporaryFileForHours=<how long should temporary files be kept?>
Germinate.UploadSizeLimitMB=<file size limit of uploads in MB>
Germinate.AvailablePages=<list of comma separated pages that are available for this instance of germinate>

Germinate.AccessionDisplayColumn=<the germinatebase column used for display purposes>
Germinate.BaseSearchColumn=<the germinatebase column used for the search and groups widget>
Germinate.CollectingsiteTreemapColumn=<the collecting sites column to group by for the treemap>
Germinate.ShowHomeOnLogin=<show the content of the home page on the login page as well?>

Germinate.Server.Logging.Enabled=<should exceptions be logged on the server?>

Germinate.IsUnderMaintenance=<is the server under maintenance?>

Germinate.IsReadOnly=<is the database in read-only mode?>

Germinate.ExternalDataFolder=<optional external data directory outside tomcat>

Germinate.HideIdColumns=<should internal id columns be hidden from tables?>

Germinate.Gallery.Images.Per.Page=<determines the number of images per page>

GoogleAnalytics.Enabled=<should google analytics be enabled?>
GoogleAnalytics.TrackingId=<the google analytics tracking id>

GoogleMaps.Api.Key=<the google maps api key to use>

CookieNotifier.Enabled=<should the cookie notifier for EU cookie law be enabled?>

Germinate.Template.CustomMenu=<the custom menu structure in XML format>

Germinate.Template.HighlightColor=<main color of all highlights on the interface>
Germinate.Template.CategoricalColors=<colors used for some of the charts>
Germinate.Template.Social.ShowFacebook=<show facebook share button?>
Germinate.Template.Social.ShowTwitter=<show twitter share button?>
Germinate.Template.Social.ShowGooglePlus=<show google+ share button?>
Germinate.Template.UseToggleSwitches=<show toggle switches?>
Germinate.Template.Show.Search=<show search in main menu?>
Germinate.Template.Logo.Contains.Link=<does the svg logo contain links?>
Germinate.Template.Show.Parallax.Banner=<show parallax image banner?>

Germinate.Template.EmailAddress=<contact email email address>

Germinate.Template.Title=<title of the germinate instance>
```

```
Germinate.Template.DatabaseName=<name of the germinate instance>
Germinate.Template.TwitterLink=<twitter link>
Germinate.Template.Copyright=<copyright information>

Path.Java=<path to the java installation>
Path.R=<path to the r installation>
```

All items starting with `Germinate.Database` have already been discussed in Section 2.5.2. Items starting with `Gatekeeper.Database` can be configured analogously.

We will now explain the meaning and possible settings of all the items. Non-optional properties are marked with "∘" whereas "∗" marks properties that are non-optional if you want to use the associated functionality. Unmarked properties either have a default value shown in **bold** or aren't crucial for Germinate to work properly.

Germinate.Database.Username$^{\circ}$ [String]
  The MySQL username used for the Germinate 3 database.

Germinate.Database.Password$^{\circ}$ [String]
  The MySQL password associated with the username.

Germinate.Database.Server$^{\circ}$ [String]
  The server MySQL is running on.

Germinate.Database.Name$^{\circ}$ [String]
  The name of the Germinate 3 database.

Germinate.Database.UsePort   [true | **false**]
  Set to `true` if a (non-standard) port should be used to connect to the database.

Germinate.Database.Port* $[x \in \{0, \ldots, 49151\}]$
  The actual port number.

Gatekeeper.URL* [URL]
  The URL of Germinate Gatekeeper. This is required if the registration feature is enabled. Refer to property `Germinate.Registration.Enabled` and Section 4.5.1.

Gatekeeper.Database.Server* [String]
  The server MySQL is running on.

Gatekeeper.Database.Name* [String]
  The name of the Germinate Gatekeeper database.

Gatekeeper.Database.UsePort   [true | **false**]
  Set to `true` if a (non-standard) port should be used to connect to the database.

Gatekeeper.Database.Port* $[x \in \{0, \ldots, 49151\}]$
  The actual port number.

Gatekeeper.BCrypt.Rounds $[x \in \{4, \ldots, \mathbf{10}, \ldots, 31\}]$
  Gatekeeper uses BCrypt to hash passwords. The number of rounds determines how long the hashing will take. A larger number will make it harder to use brute-force to find the password, but at the same time it will influence how long users will have to wait for their password to be verified. The default is a value of 10 which results in 2^{10} rounds. Increasing this to 11 will **double the runtime** to 2^{11}.

Gatekeeper.Registration.Enabled [true | **false**]

Germinate and Gatekeeper support user registration. If this property is set to `true`, users will be able to register for access to your instance of Germinate. The registration will be accessible from the login page. See Section 4.5.1 for more details.

Gatekeeper.Registration.Needs.Approval [**true** | false]

If registration is enabled, there are two ways how users are approved for access to Germinate. If this property is set to `false`, users will automatically be approved and can start using Germinate right away. If this is not what you want, set this to `false` which will require you to approve requests manually through the Gatekeeper interface. Users will be notified with your decision via mail.

Germinate.UseAuthentication [true | **false**]

If set to `true`, the user will have to log in using the credentials stored in Germinate Gatekeeper, otherwise no login procedure is required.

Germinate.CookieLifespanMinutes [$x \in \mathbb{N}^+$; default: **1440**]

The lifespan of all cookies given in minutes.

Germinate.Debug [true | **false**]

Set to `true` if you want to see the SQL queries that are run against the database on each page. This is very handy when debugging or searching for errors. However, this should **never** be used in production use, since it exposes the database internals.

Germinate.KeepTemporaryFileForHours [$x \in \mathbb{N}^+$; default: **24**]

The amount of hours temporary files should be kept for. Temporary files are generated whenever the user wants to download parts of the database. They are kept locally at least for the given amount of hours and after this amount of time they will be deleted the next time a user requests any temporary file.

Germinate.UploadSizeLimitMB [Float; default **0.5**]

The file size limit of uploads in MB.

GoogleAnalytics.Enabled [true | **false**]

Germinate supports Google Analytics. Page navigation as well as user interactions will be tracked if this property is set to `true`.

GoogleAnalytics.TrackingId[*] [String]

The tracking id provided by Google Analytics.

GoogleMaps.Api.Key[*] [String]

The Google Maps API key. Starting from mid-2016 Google Maps requires websites to use an API key when using Google Maps. The key is free and can be acquired from this website:

        `https://developers.google.com/maps/documentation/javascript/get-api-key`

CookieNotifier.Enabled [true | **false**]

In accordance with the EU Cookie Law [2] we provide a notify banner at the bottom of the page that informs users that we use cookies. Set this to `false` to disable this banner.

Germinate.AvailablePages[○] [CSV]

Not all pages are useful for all instances of Germinate. List the names of those pages that should be available for this instance in a comma-separated fashion, e.g. `climate, megaEnvironments, gallery`.

Germinate.AccessionDisplayColumn   [String; default: **name**]
> Whenever accessions are displayed in charts, we need to show an identifier or a name of the accession. We allow the user to define which of the germinatebase table columns they want to use for this purpose.

Germinate.BaseSearchColumn   [String; default: **name**]
> On many parts of the page, the user is able to search for accessions based on a given column of the germinatebase table (e.g. `name`, `number`, etc.). Specify this column name here to search in this column.

Germinate.CollectingsiteTreemapColumn   [String; default: **country_name**]
> The column used to group the collecting sites for the treemap (e.g. `region`, `country_name`, etc.).

Germinate.ShowHomeOnLogin   [true | **false**]
> If set to `true`, the login page will show the content of the home page as well. If set to `false`, the login page will only contain the text fields for the username and password.

Germinate.Server.Logging.Enabled   [true | **false**]
> If set to `true`, exceptions will be logged on the server side. Otherwise, they will only be forwarded to the client and the client decides what to do with them.

Germinate.IsUnderMaintenance   [true | **false**]
> If set to `true`, the web interface will be completely disabled, just showing a notification that the system is under maintenance.

Germinate.IsReadOnly   [true | **false**]
> If set to `true`, the web interface will not allow the user to perform changes to the database, i.e., the generation/modification of user-created content will be disabled.

Germinate.ExternalDataFolder[*]   [Path]
> If this property is set, Germinate will use the given path to look for files like images, data files and external applications. If the property is not set, Germinate will use the internal folders. This option can be useful if your data is huge and you don't want to include it in the generated war file, but rather want to store it in a different location on the server. Make sure that Germinate has read and write access to this folder.

Germinate.HideIdColumns   [true | **false**]
> If set to `true`, the internal id column will be hidden from all tables.

Germinate.Gallery.Images.Per.Page   [$x \in \mathbb{N}^+$; default: 16]
> This setting determines how many images are shown per page on the gallery page.

Germinate.Gallery.Make.Thumbnails.Square   [true | **false**]
> If set to `true`, the automatically generated thumbnails of images will be square to keep a uniform look across all thumbnails. The default is `false` to keep it consistent with previous versions.

Germinate.Template.CustomMenu   [XML]
> This allows you to customize the main menu of Germinate. Please consult Section 3.1 for more details and an example.

Germinate.Template.HighlightColor   [HEX; default: **#00ACEF**]
> The color used for all highlights on the Germinate interface.

Germinate.Template.CategoricalColors$^\circ$ [CSV(HEX)]
    A list of comma separated HEX color values (including the hash) that are used to color
    categories in some of the charts.

Germinate.Template.Social.ShowFacebook [true | **false**]
    Set to true if a Facebook share button should appear on the site.

Germinate.Template.Social.ShowTwitter [true | **false**]
    Set to true if a Twitter share button should appear on the site.

Germinate.Template.Social.ShowGooglePlus [true | **false**]
    Set to true if a Google+ share button should appear on the site.

Germinate.Template.UseToggleSwitches [**true** | false]
    Set to true if you want to use toggle switches instead of two radio buttons whenever the
    user has to decide between a "yes" and a "no" option.

Germinate.Template.Show.Search [true | **false**]
    Set to true if you want a dedicated "search" menu item in the main Germinate menu.

Germinate.Template.Logo.Contains.Link [true | **false**]
    Set to true if the main website logo SVG file contains links. In that case, Germinate will
    disable the default logo link to "home" and prioritize the SVG-internal links.

Germinate.Template.Show.Parallax.Banner [**true** | false]
    Determines if the parallax scrolling image banner is shown at the top of selected pages or
    not.

Germinate.Template.EmailAddress [E-Mail]
    The email address that will be displayed on the page as a contact address.

Germinate.Template.Title [String]
    The text to show in the browser title.

Germinate.Template.DatabaseName [String]
    The text to show in the "featured banner" on the page.

Germinate.Template.TwitterLink [URL]
    The link to the associated Twitter account.

Germinate.Template.Copyright [String]
    The copyright information shown at the bottom of the page.

Path.Java [String]
    The path to the Java installation. This is optional, since Germinate can get it from the
    JVM it's running in. However, if required, Germinate can be pointed to a different Java
    version and will use this to run all internal Java calls.

Path.R* [String]
    The path to the R installation.

Any change to these properties will automatically be picked up by Germinate, i.e. no reload of
the web interface on Tomcat is required.

    There is also a configuration page directly in Germinate. This page is only visible to administrators and can be reached by going to:

        `http://<Your web server>:8080/<project.name>/#admin-config`

An example of this page can be seen in Figure 3.

Figure 3: The Germinate configuration page available to administrators.

3.1 Germinate Menu

Germinate uses a predefined menu structure by default. This default menu setup is based on what we think is a reasonable structure. However, even we are not infallible, so we decided to make the menu customizable.

We will now explain how you can define your own, custom menu structure and we will then give an example of how this can be used.

3.1.1 Structure

The menu of Germinate can be defined in XML format. We will explain this format using the example below:

```xml
<menu>
    <item key="home">
        <label key="en_GB">Home</label>
        <label key="de_DE">Home</label>
    </item>
    <item key="category.data">
        <label key="en_GB">Data</label>
        <label key="de_DE">Daten</label>
        <item key="browse-accessions">
            <label key="en_GB">Accessions</label>
            <label key="de_DE">Muster</label>
        </item>
        <item key="category.molecular">
            <label key="en_GB">Molecular data</label>
            <label key="de_DE">Molekulare Daten</label>
            <item key="genotype-datasets">
                <label key="en_GB">Genotypes</label>
                <label key="de_DE">Genotypen</label>
            </item>
            <item key="map-details">
                <label key="en_GB">Maps</label>
                <label key="de_DE">Molekulare Karten</label>
            </item>
        </item>
```

```
    </item>
    <item key="category.environment">
        <label key="en_GB">Environment</label>
        <label key="de_DE">Umwelt</label>
        <item key="geographic-search">
            <label key="en_GB">Geographic search</label>
            <label key="de_DE">Geografische Suche</label>
        </item>
    </item>
</menu>
```

This will result in the following menu structure:

British English:

- Home

- Data

    - Accessions

    - Molecular data

        ∗ Genotypes
        ∗ Maps

- Environment

    - Geographic search

German:

- Home

- Daten

    - Muster

    - Molekulare Daten

        ∗ Genotypen
        ∗ Genetische Karten

- Umwelt

    - Geographische Suche

We will now explain the individual elements of the XML file:

menu The root element of the XML file.

item This is used for any menu element. This can either be a link to an actual page, or a submenu.

item key If this item represents an actual Germinate Page (cf. Section 4.5) then this has to be the name of the page. If this item is a sub-menu, then this has to be a unique id.

label Labels are used as the text content of the menu item, i.e. what the user will see. Every item has to provide this information for ALL of the supported languages (cf. Section 6.2).

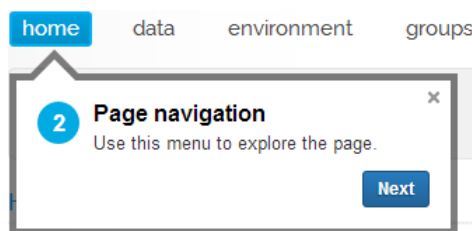**label key** The locale of the supported language.

Figure 4: Example step of a tour

4 Features

In this section, we will highlight the main features of Germinate. This section will expand as we add new features.

4.1 Internationalization and Localization

Germinate fully supports internationalization for an unlimited number of languages. Every text that you can see on the web interface can be localized. The language used on start-up is chosen based on the browser settings, but the user can easily switch to a different language by selecting it from the combo box at the bottom of the page. Have a look at Section 6.1 for usage details.

4.2 Help

Every page can include a help text. This text will be shown when the user clicks on the help icon. The main purpose of this help is to explain features of the current page to the user. As a result, the page itself does not look cluttered, but the user is still able to find help if the workflow of the page is not obvious.

4.2.1 Tours

In addition to a simple help text, we also offer the ability to add interactive tours to the pages. An interactive tour can be described as a sequence of help texts with visual feedback on the page itself. These steps can guide the user through the page and explain every step in detail. An example of a tour step can be seen in Figure 4.

4.3 News

The Germinate web interface contains a place holder to show the latest news of the project. On the database side, these news are stored in two tables and can be updated at any time. The web interface will check the availability of news and show the three latest entries at the bottom of the page. Changes on the database side will immediately be visible on the website.

4.4 Notifications

Germinate offers a set of essential notification types. The notification system is based on Toastr [3]. All notifications fully support internationalization as well as custom styling. The currently available notification types can be seen in Figure 5. Section 6.5 contains an example showing how to use the notification system.
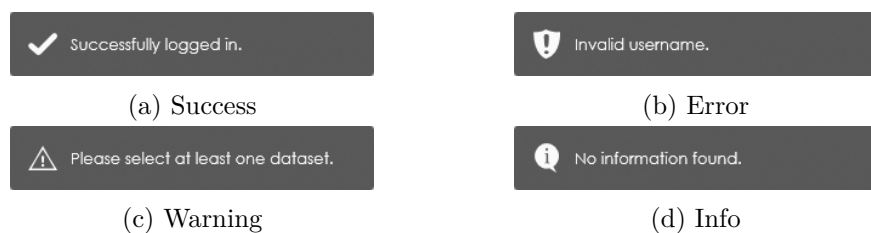
(a) Success

(b) Error

(c) Warning

(d) Info

Figure 5: Available notifications

4.5   Available pages

This section gives an overview of all the pages that are available for Germinate. Based on your requirements and the data that is available, you can decide which pages should actually be available on the web interface and hide all others. The set of available pages can be changed dynamically without having to re-deploy the application. This gives you the freedom to customize your Germinate experience just as you please.

A table with the available pages can be found in Table 1.

Page name	Description	Required data
about-germinate	Shows information about Germinate	-
about-project	Shows information about the project	-
accessions-for-collsite	Shows available accessions for the selected collecting site	germinatebase, locations, locationtypes, countries
acknowledgements	Shows acknowledgements	-
allele-freq-dataset	Shows the datasets containing allele frequency data	datasets, datasetstates, datasetpermissions, experiments
allele-freq-export	Page showing the settings used for the export of allele frequency data	allelefrequencydata, datasets, datasetstates, datasetpermissions, germinatebase, groups, groupdatasets grouptypes, maps
allele-freq-result	Page showing the result of the allele frequency data export	allelefrequencydata, datasets, datasetstates, datasetpermissions, groups, maps

Page name	Description	Required data
browse-accessions	A paginated table containing all the accessions in the database.	countries, germinatebase, locations, taxonomies, subtaxa
cart	The cart contains marked accessions.	germinatebase
categorical-datasets	Shows the datasets containing categorical data	datasets, datasetstates, datasetpermissions, experiments
categorical-export	Page showing the settings used for the export of categorical data	datasets, datasetstates, datasetpermissions, germinatebase, groups, groupdatasets, grouptypes phenotypes
climate	Page showing the settings used for the export of categorical data	climatedata, climates, groups, units, groups, grouptypes, phenotypes
collsite-treemap	Page showing a treemap of the collectingsites	countries, germinatebase, locations
cookie	Page showing the cookie policy	-
dataset-overview	Page showing all the available datasets	datasets, datasetstates, datasetpermissions, experiments, experimenttypes
gallery	A gallery of images	-
gbs-export	List of exported genotypic data files	-
genotype-datasets	Shows the datasets containing genotypic data	datasets, datasetstates, datasetpermissions, experiments

Page name	Description	Required data
genotype-export	Page showing the settings used for the export of genotypic data	datasets, datasetstates, datasetpermissions, genotypes, germinatebase, groups, groupdatasets grouptypes, mapdefinitions, maps, markers
geographic-search	Page used to query for collecting sites by distance to a query location	countries, locations
geography	Page that shows different visualizations of collecting sites on maps	countries, locations
groups	Page listing defined groups of accessions, markers, ...	groupmembers, groups, grouptypes
help	Page with general help information	-
home	Welcome page of the web interface	-
login	The login page of Germinate	-
logout	The logout page of Germinate	-
map-details	Page listing the details about a selected map	mapdefinitions, mapfeaturetypes, maps, markers
marker-details	Page listing the details about a selected marker	datasets, genotypes, markers
mega-environments	Page with the mega environments	countries, germinatebase, locations, megaenvironmentsdata, megaenvironments, megaenvironmentsource, soils
news	List of the latest news	news, newstypes
passport	Page listing the details about a selected accession	countries, germinatebase, locations, subtaxa, taxonomies
pco-coordinates	List of exported pco data files	-
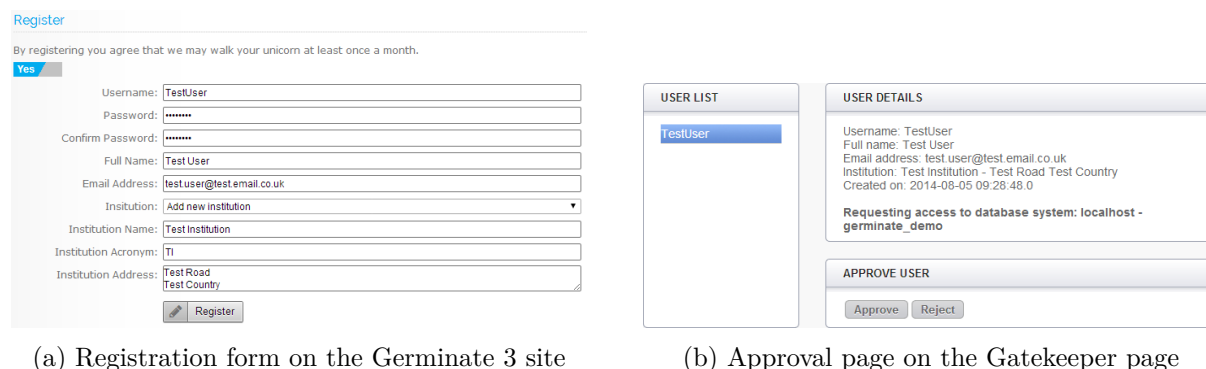
(a) Registration form on the Germinate 3 site   (b) Approval page on the Gatekeeper page

Figure 6: User registration

Page name	Description	Required data
search	Shows search results	countries, germinatebase, locations, markers, mapdefinitions, mapfeaturetypes
trials	Page with different visualizations of trials data	germinatebase, phenotypes, trialsdata
trials-datasets	Shows the datasets containing trials data	datasets, datasetstates, datasetpermissions, experiments
trials-individual	Shows visualizations of trials data for an accession and a phenotype	trialsdata, trialseries, trialsites

4.5.1   User registration

Germinate can be used with and without authentication. This means that you can decide if you want to restrict access to your Germinate instance to registered users. If you decide to do so, you'll need people to be able to register. The registration form of Germinate (Figure 6a) provides a concise and easy way to register. If you choose to use a disclaimer that potential new users have to accept, this will appear before the form is visible. Once the user completes the form, they will either get access right away or you need to approve the new users manually. This is based on the setting of the property `Gatekeeper.Registration.Needs.Approval` (see Section 3 for help). If you choose to use the approval approach, you'll see the screen shown in Figure 6b after selecting "Approve users" in Gatekeeper. The user will be notified with your decision.

4.5.2   Groups

Germinate allows you to define groupings of various items which makes exporting or visualizing data more comfortable. An intuitive example of groups are accession groups. A group of

(a) Accession search by item criteria

(b) Collecting site search by climate criteria

Figure 7: Examples of group searches

accessions can contain accessions that are similar to each other or a list of accessions that you want to export against a map for visualization in Flapjack.

A new group can be created with a couple of clicks: First, select the type of group that you want to create, e.g. accession group, collecting site group, marker group etc. Afterwards, create a new group by just entering the name of the new group.

To delete group members, just select the group that you want to modify and check the items to remove in the table. Afterwards click on the delete button below the table. To delete a complete group, click on the delete icon next to the group selection box.

4.5.2.1 Searching by item criteria

Searching for the items you want to add to the group is very straight forward. Germinate supports searching on almost all columns of the table containing the data. Examples are: "country of origin", "region", "state, "latitude", "longitude" and "elevation" for collecting sites. You can restrict your search by specifying multiple criteria. The criteria currently have the following structure:

<center><column> {less than, greater than, equal} <value></center>

Multiple criteria can be combined using logical conjunction (∧, AND) and disjunction (∨, OR). Note that conjunction binds stronger than disjunction making the order of criteria important. An example can be seen in Figure 7 (a). The example requests all accessions with "mexico" as their country of origin **or** those accessions with a name starting with "A".

4.5.2.2 Searching by climate criteria

If climate data is available, Germinate allows to select subsets of items based on a combination of climate criteria. The process is identical to the one seen above. Figure 7 (b) shows an example. The example requests collecting sites having a maximal average temperature of less than 30 **and** having a minimal average temperature of more than 20.

4.5.3 Data

The main purpose of Germinate is to store multiple data types, let the user query them and create visualizations of the data. In this section, we will give an overview of the different data types and visualizations Germinate supports.

4.5.3.1 Browse accessions and Passport

This page lists all accessions stored in Germinate in a tabular format. The table is sortable and supports pagination, i.e. it will split into multiple pages if necessary to ensure readability. Clicking on an accession will take you to the passport page. The passport page contains all the meta information that is available for the selected accession such as name, genus, breeder information, images and annotations.

Passport data tends to be quite fragmented so its perfectly possible that we have limited data for some lines while others have a full complement.

4.5.3.2 Maps and Markers

The map page shows a list of available maps defined in Germinate. Clicking on a map will show its member markers with detailed information about chromosome and position on the chromosome. In turn clicking on a marker shows information about the molecular and genotype information about this marker as well as a list of the datasets the marker is contained in. Click on the dataset to reveal the genotypes for the selected combination of marker and dataset.

4.5.3.3 Genotype data export

The genotype information held in Germinate can be exported as tab separated values as well as in a format that is compatible with the genotype data viewer *Flapjack*. The process of exporting the data is structured like a wizard dialog.

First you need to select the dataset containing the data that you want to export. The second wizard page shows the accession and marker groups as well as the maps that are visible to you. Just select the items that you want to export. The *Crap Data Filter* (CDF) can be enabled to prune markers from the export that exceed a certain threshold for missing or heterozygous values. Finally, you need to select the file containing the raw data. The third page finally contains your exported data in both already mentioned formats. If the CDF was enabled, this page will show the list of deleted markers for your convenience.

Note: Make sure that all the files that contain your genotype data are encoded with UTF-8.

4.5.3.4 Allele frequency data export

Exporting allele frequency data is almost identical to the genotype data export. The first two wizard pages are again used to select datasets and accession and marker groups. The third page differs from the genotype export, because this is the page where you actually can decide how you want your data to be binned. The binning step is necessary for visualization in Flapjack. Currently, there are three types of binning mechanisms: *Equal-width binning*, *Split point binning* and *automatic binning*.

Equal-width binning will result in a specified number of bins all having the same width. Split point binning is an extension of this principle. It lets you specify a split point and a number of bins to either side of the split point. The parts to the left and the right of the split point are binned according to the equal-width binning principle. Automatic binning is more sophisticated. It will automatically determine the best splitting points given a number of bins. The resulting binning will contain approximately the same number of items in each bin. Another name for this principle is consequently equal-size binning.

4.5.3.5 Phenotype data export

The phenotype export process is structured similarly to the genotype export. In the first step you can select the dataset containing your data. The next page will show the available phenotypes and accession groups that can be selected for export.

Now you can select if you just want to have a look at the data or if you want to be able to download it as well. Check the checkbox at the bottom of the page to create a downloadable file containing the data. Pressing continue will start the export process. Once the date is returned from the server, it will be displayed at the bottom of the page. A download link will be attached below the table if you decided to select data download in the previous step.

4.5.4 Environment

Environmental data can include climate data, GIS (Geographic information system) data, soil data and many more. Germinate can store all that information and use it to create meaningful visualizations and overviews.

4.5.4.1 Geography

The geography page visualizes the geographic data contained in Germinate on Google Maps. So far the only instance that holds geographic information are the collecting sites, for which we store latitude, longitude and altitude information.

There are three types of maps that are currently supported. The first type is a simple map with a marker for each data point in the database. This can get quite cluttered when the number of items increases. The two other types of maps pose solutions to this issue.

The first of them is a *Heatmap Layer* [4] for the map. A heatmap is a data visualization technique where the data points are not represented individually, but instead they are used to calculate a density function which is then plotted on top of the Google Map. The density function assigns a high value to areas where the probability of observing data points is high and a low value to areas where this probability is low. To visualize this principle, the heatmap is color-coded.

The other alternative to individual markers is the *MarkerClusterer* [5]. The MarkerClusterer uses a very basic grid-based clustering algorithm to group clusters located in the same area into clusters. This solution massively reduces the number of visible markers on the screens since each cluster is now represented by a marker instead of the hundreds of actual markers it contains. The clusters will be re-generated on every zoom level resulting in approximately the same number of visible clusters independent of the zoom level.

Figure 8 shows an example dataset containing about 1000 markers. Figure 8(a) shows the individual markers and we can see that this is very cluttered. Figure 8(b) and 8(c) show the HeatmapLayer and MarkerClusterer respectively.

4.5.4.2 Geographic Search

The geographic search can be used to find collecting sites with the help of a Google Map. You can either enter coordinates (latitude and longitude) directly, or scroll to the position of interest on the map. A click on the "Continue" button will list all collecting sites ordered by their distance (in km) to your query location.

4.5.4.3 Geographic Treemap

Treemapping (`http://en.wikipedia.org/wiki/Treemapping`) is a method for displaying hierarchical data by using nested rectangles. In our case, we visualize the hierarchy of locations in a treemap. The highest level of abstraction can be defined using the `Germinate`
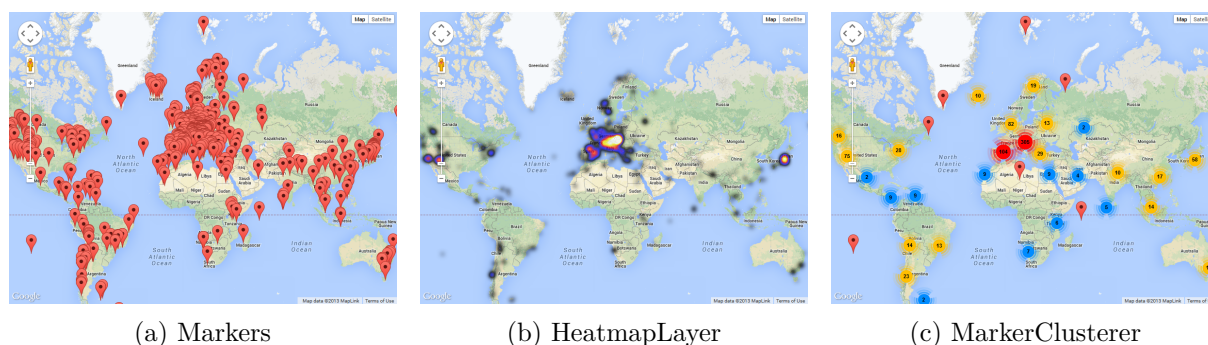
(a) Markers   (b) HeatmapLayer   (c) MarkerClusterer

Figure 8: Different representations of the same geographic information

Figure 9: Color-coded climate table (average temperature in ℃)

.CollectingsiteTreemapColumn property described in Section 3. Clicking on one of the top level rectangles will zoom in and show the actual collecting sites. Clicking on those will take you to a page listing all the accessions collected at this site.

4.5.4.4 Mega Environments

A mega environment can be defined as a set of areas sharing the same or very similar attributes ranging from biotic and abiotic stresses to customer preferences.

The mega environments page lists all recorded mega environments with the collecting sites located within them. Selecting a collecting site takes you to a page showing the accessions found at this site.

The data contained in both the mega environment page as well as the collecting sites page can be downloaded in KML (**K**eyhole **M**arkup **L**anguage) format allowing viewing in Google Earth [6].

4.5.4.5 Climate

Climate data is very diverse. Everything that can be measured by sensors can be considered as climate information. Examples are precipitation, hours of daylight, temperature, frost frequency and atmospheric pressure. Germinate can store all kinds of climate data. The climate page of the Germinate web interface visualizes the contained data in a compact and understandable way. Figure 9 shows an example visualization using color-coding of the average measured temperature for the given collecting sites.

If you have climate layer images that go along with your climate data, Germinate is able to display them on top of a Google Map below the climate chart.

4.5.5 Image gallery

The image gallery is an elegant way to display images that do not fit on any of the other pages. The images will be held in a table structure and can be browsed in a pagewise fashion. New

images can easily be added by copying them into the following directory:

```
<Apache Directory>/webapps/<project.name>/WEB-INF/classes/download
```

There are two sub-folders called `images` and `thumbnails`. Copy the full size images into the `images` directory and small thumbnails into the `thumbnails` folder. For optimal visual appearance make sure that all the thumbnails have the same height. A height of 150px is a good reference value.

The images will be sorted by the "Last Modified Date" making sure that the newest images appear first.

4.5.6 Search

Every website should have a basic search feature. The search feature of Germinate allows to search for accessions, collecting sites, markers etc. by their features. You know the name of an accession? Excellent, just search for it and tell Germinate that it's the name you're searching for. Only know the ID? No problem, just use the ID instead and mark it as the ID. We use `%` as the wildcard character. A wildcard character can be used to substitute for arbitrary other characters. As an example, `%king%` will match `United Kingdom`, whereas `king%` will only match `Kingdom`. Note that we currently don't match cases during the search.

4.6 Color Themes

Germinate is highly customizable. We have already seen a lot of configuration options in Section 3. In this section we want to highlight one of the customization features in particular: color themes. The configuration properties responsible for the color theme of Germinate are:

```
Germinate.Template.HighlightColor=<main color of all highlights on the interface>
Germinate.Template.CategoricalColors=<colors used for some of the charts>
```

We will now show how the look of Germinate can be completely changed by just setting these two properties. The default color theme of Germinate is a composition of different shades of blue (see Figure 10a). With the following property changes, we get the completely changed theme seen in Figure 10b.

```
Germinate.Template.HighlightColor=#e74c3c
Germinate.Template.CategoricalColors=#e74c3c,#d35400,#e67e22,#f39c12,#f1c40f
```

As you can see, the color theme affects almost all elements of the website: links, headings, backgrounds, icons, highlight effects, chart colors, etc.

Although we do our best to take care of all elements, we can't simply change the images you include, i.e. the included images do not change colors based on these settings. An example is the Germinate 3 logo in both screenshots. As can be seen, it stays blue. However, all other elements that are included in the default installation of Germinate will change color according to the defined properties.

4.7 Security

As already mentioned earlier, Germinate is equipped with a secure login system[1]. This feature is completely optional, but it allows you to protect your data from any unauthorized access. In this section, we will explain how the security system works and we will show what is necessary to use it properly.

---

[1]Germinate is only as secure as the connection between client and server. Use a secure connection (TLS/SSL) to prevent password snooping.

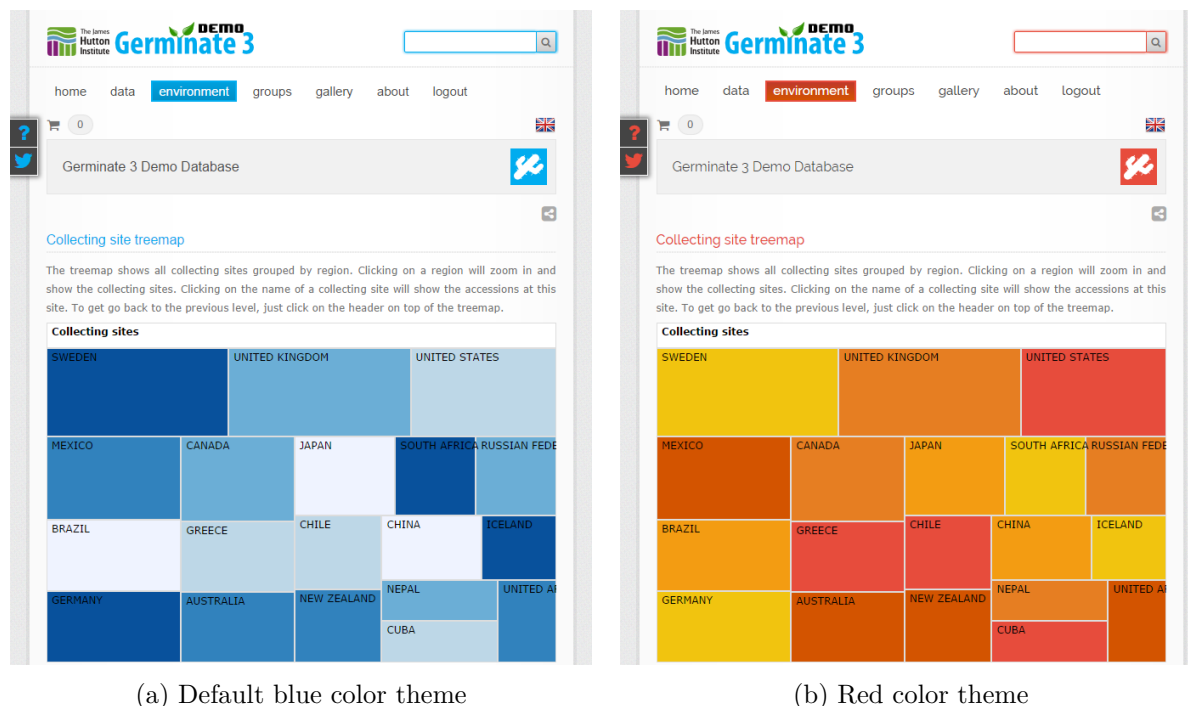(a) Default blue color theme   (b) Red color theme

Figure 10: Different color themes

Figure 11 shows the authentication process of Germinate. The figure contains two major parts. The upper part visualizes the login process, which is initialized by the user entering his/her credentials. These are sent to the server alongside any session id that is still stored in the client from previous sessions. The server will then check the session id that is received. If the id is still valid, the server will store the session id and return it to the client, which, in turn, will create a cookie containing this id.

If the session id is invalid, the server will check if the username password combination is genuine. This is done by encrypting the password and checking it against the entry in the database. If this check fails, the user will be logged out. Otherwise, the server will create a new session id, store it in the HTTP session and return it to the client, which will create a new cookie using this id. The login process is now complete and both the server (via HTTP session) and the client (via cookie) know the current session id.

For each new request that is made from the client, it will need to send the session id as the payload, i.e. each remote procedure call (RPC) has to request the session id as a parameter to ensure the security of the communication. As a consequence if this, the server will receive the session id three times per request, namely via the HTTP session, via the cookie and via the payload. If all of these ids match up, the server can fulfil the client's request and return the data. However, if it fails, the user will be logged out.

As a final remark: Even if you currently do not want to enable the security feature, you should still write your code in a way that ensures it will work properly even when the security feature is enabled. Otherwise you might end up with a security leak.
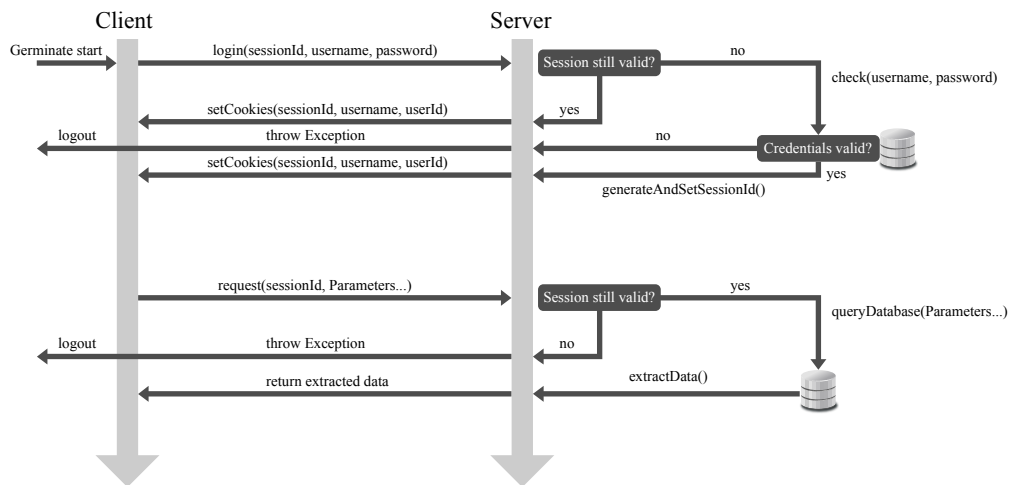
Figure 11: Authentication procedure of Germinate

5   Structure

In Section 5.1, we will explain the structure of the Germinate project. This includes the folder structure of the subversion project as well as the structure of the deployed product. Section 5.2 contains information about the structure of the database itself.

5.1   Subversion project

(a) Overall structure   (b) Web structure
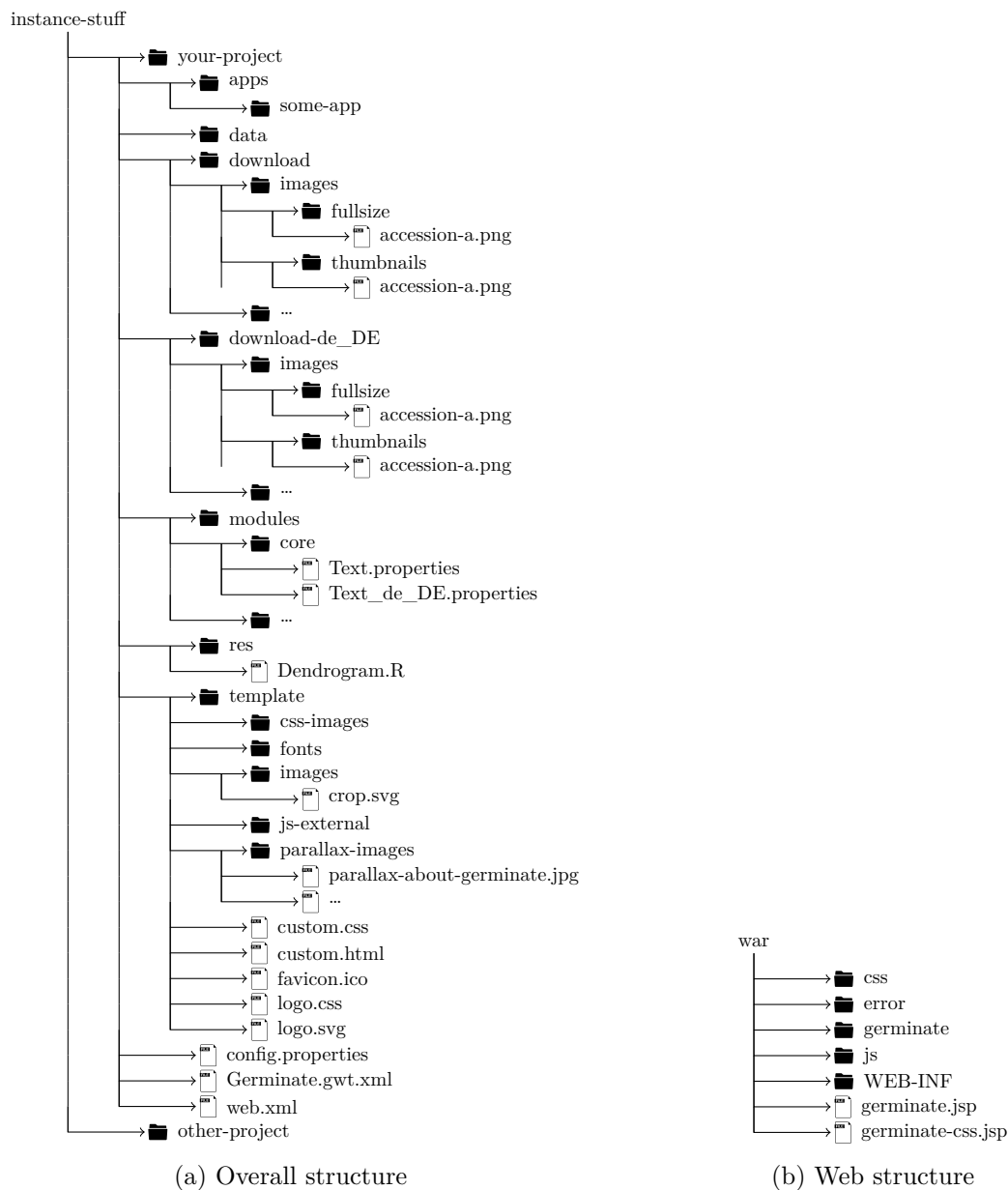
Figure 12: Available notifications

The actual subversion project of Germinate is structured as shown in Figure 12. Figure 12a shows the overall structure of the whole project whereas Figure 12b shows the structure of the files and folders related to the web side of things.

5.1.1　Project structure

Germinate is designed to allow custom look and feel for different projects. This means you are able to have multiple configurations of Germinate, each tailored specifically for the special needs of the individual projects. As an example we show the structure of the project "your-project" which in reality would be named after your project.

There are several sub-directories. The "apps" directory includes external applications that ship with Germinate. In this example, we include an application called "some-app". The "data" directory contains raw data files that are used during the export processes. The "download" directory contains all the data files that you want to provide as downloads on Germinate. Those can include images for the gallery, pdfs, etc. The "download-de_DE" directory contains internationalized versions of the files found in "download". See Section 6.3 for further details. The "modules" directory contains the internationalization property files for each module. In this example, there is only one module, namely the "core" module which is the basic Germinate module. Internationalized text goes here. The "res" folder can be used to host several files that are used as resources. The content of this folder is copied to the server and accessible from the server side code. In this example we have an R file that is used on the server to generate a dendrogram.

Finally, the "template" folder contains custom styling files for your individual instance of Germinate. Those include custom logos, custom css, custom html, custom JavaScript libraries, custom parallax images, fonts, a favicon, etc.

The other files in your project folder are the most important ones. They define how Germinate works. We'll give a detailed description of each file below.

5.1.1.1　Config properties

`config.properties` is the file you have already read about in Section 3. It is used to define how to connect to the database, where to find Flapjack and R, which title to use for Germinate and so on.

5.1.1.2　Germinate xml

`Germinate.gwt.xml` is a very technical file. It should be left alone unless you have a good working knowledge of the Google Web Toolkit. There is, however, one thing in here that is easily to configure which are the supported languages.

An example of the internationalization properties can be found below:

```xml
<!-- Internationalizations -->
<extend-property name="locale" values="en_GB" />
<extend-property name="locale" values="de_DE" />
<set-property-fallback name="locale" value="en_GB" />
```

The `extend-property` entries list the supported language, or more exact, the locales. In this case, we support German and British English. You can add any locale you want to this file as long as you also provide a language properties file of the form `Text_xx_XX.properties`.

The `set-property-fallback` entry defines the default locale as well as the locale that is used if an unsupported locale is requested. It has to be one of the previously defined locales. In this case we use British English.

5.1.1.3　Web xml

Every Java web developer will be familiar with this file. We will give a short explanation of the content here.

```xml
<display-name>Germinate 3 Template Database</display-name>
```

```
<!-- LISTENERS -->
<listener>
    <listener-class>jhi.germinate.server.util.ApplicationListener</listener-class>
</listener>

<!-- Default page to serve -->
<welcome-file-list>
    <welcome-file>germinate.jsp</welcome-file>
</welcome-file-list>

<!-- CUSTOM ERROR PAGES -->
<error-page>
    <error-code>404</error-code>
    <location>/error/error.jsp</location>
</error-page>
```

The code above shows part of the template file. `welcome-file` is the default file that is served to the browser on the first visit. The `error-page` entries define the error pages that Germinate will take care of. All other error pages are handled by Tomcat instead.

5.1.2 Web structure

The most important file in this structure is `germinate.jsp`. This file contains the basic web skeleton of the website, i.e. the overall structure. In this file, you should define external JavaScript files and external css files you want to load in addition to the already contained ones. `germinate-css.jsp` contains the custom Germinate style sheet. Most of what you find in this file takes care of how the content of the page looks. To change the styling of the overall template (not the content), use this file: `css/style-css.jsp`.

As you might already have notices, both files are not typical `.css` files, but rather `.jsp` files. However, the first line in all of these files lets the browser know, that in fact they are css stylesheets. The reason for using jsp instead is based on the fact, that we have access to the `PropertyReader` in the jsp files. This allows us to get properties from the server and change the stylesheets before they are sent to the browser.

We use this feature to get the so called "highlight color" from the properties file. This color is used for most of the header elements on the web site as well as several other things including links. If we change this color in the properties file, the stylesheets will pick it up and use this color from now on. This centralized mechanism reduces errors and makes it easier to change things quickly.

As mentioned earlier, you can simply add external css and js files to Germinate. There are predestined folders for these here as well. The `germinate` folder is created by Eclipse when you compile the project. It contains all the JavaScript source files that are generated and served to the browser. You don't need to worry about this folder at all. The `error` folder contains resources for the error pages. At the moment Germinate provides humorous error pages for the most common errors: 404, 403 and 401. Feel free to add further error pages.

Finally, the `WEB-INF` folder should look familiar to every web developer. It contains the libraries (in form of `.jar` files that are necessary to run Germinate on the server as well as the compiled Java source.

5.2 Database

We moved the documentation of the database to an online resource. You can find an overview of all the tables with detailed descriptions at the following location:

> `https://ics.hutton.ac.uk/resources/germinate/model/germinate_3_4_0/`

6 Example code

Possible changes to Germinate may range from basic formatting changes to the addition of completely new content. The following sections contain detailed examples for the most common scenarios of extension. Each example consists of the necessary code and accompanying explanations.

6.1 Internationalizing your pages

As already mentioned in Section 4.1, Germinate supports internationalization for as many languages as you want. Internationalizing your custom code is really easy:

1. Define a method that returns your localized text in `jhi.germinate.client.i18n.Text`.

2. Add the localized values for this new variable to the `.properties` files located at `instance-stuff/<project.name>/modules/core/`. The file naming convention is: `Text_<Language>_<Region>.properties`.

A list of supported locales is available at [7].

As an example, we will now add a new localized String to the application. To get the localized text, we used the method `Text.LANG.menuData()` which returned "data" for the English version of the page. Now let's assume we want to add a menu item that has the text "my fancy page". The first thing we have to do is create a new method in the interface `jhi.germinate.client.i18n.Text`:

```
String menuMyFanyPage();
```

As we can see, this method will return a `String` which we can use in our menu. You can regard this method as a template for the localization.

The second step is to provide a translation of this text for each of the supported languages:

```
menuMyFanyPage=my fancy page
```

Add this to the files mentioned in the second bullet point above.

GWT will substitute the template calls with the appropriate localized text during compile time.

6.1.1 Parametrization and Plural Forms

Instead of just returning static text from the properties files, Germinate is able to substitute place holders with parameters allowing for a dynamic usage of localization. One specific case of parameter usage is the case of plural forms. The text you want to display might differ based on the number of menus it should represent. As an example, consider the sentence: "I can see 3 trees". The sentence changes for just one tree: "I can see a tree". The method returning this text could look like this:

```
String iSeeTrees(@PluralCount(DefaultRule_en.class) int number);
```

The annotation `@PluralCount` tells the method that the parameter called `number` is a countable variable. The associated properties entries could look like this:

```
iSeeTrees=I can see {0} trees.
iSeeTrees[one]=I can see a tree.
iSeeTrees[many]=I can see many trees.
```

Table 2: Available plural form rules in GWT

Category	Rules	Examples
zero	x is 0	0
one	x is 1	1
two	x is 2	2
few	$x \bmod 100 \in \{3, ..., 10\}$	3-10, 103-110, …
many	$x \bmod 100 \in \{11, ..., 99\}$	11-99, 111-199, …
other	Everything else	100-102, 200-202, 11.68, …

As you can see, the plural form is handled by the first line, while the singular form is handled by the second line. A list of available plural forms can be seen in Table 2. For more information, please refer to [8].

Another case of parametrization is simple substitution. As an example we will create a method returning a welcome message for the user and the day of the week.

```
String welcomeMessage(String username, String day);
```

The method is defined just the way a normal method taking two parameters would be defined. The localized entry in the properties file could look like this:

```
welcomeMessage=Hello {0}. Welcome to Germinate 3. Today is {1}.
```

Calling `welcomeMessage("Joe Bloggs", "Wednesday");` will result in "Hello Joe Bloggs. Welcome to Germinate 3. Today is Wednesday".

6.1.2 Notes

- It is possible to include HTML tags in the localized strings. Consequently, you can change the font style (size, bold, italic,...), add hyperlinks, include images, etc.

- Use two single quotes instead of just one (e.g. "It"s done" instead of "It's done").

- The encoding of the properties files can be either UTF-8 or ISO-8859-1. If you choose the latter, make sure to encode all non-Latin-1 characters by using Unicode escape characters, e.g. \uXXXX where XXXX is the hexadecimal index of the character in the Unicode character set. A list of these characters can be found at [9].

6.2 Adding a new language

We have seen how to add content to existing languages in the previous section. Now we will show how to add a completely new language to the application. To ensure neutrality, we selected Switzerland and will use the language of Swiss German. The locale id of Swiss German is `de_CH`. The first part represents the language (de = Deutsch which is German for "German") and the second part represents the country (CH = Confoederatio Helvetica which is Latin for "Swiss Confederation").

Navigate to your configuration folder located at `instance-stuff/<project.name>` and create a new file called `Text_de_CH.properties`. Now copy the whole content of one of the fully translated other languages into this file and substitute all of the text with its appropriate translation.

Next, you'll need to make a change to this file: `war/css/language-selector-css.jsp`. Search for the line containing the 2-digit country code of the country associated with the new language. In our example, that would be `CH` and the line looks something like this:

```
span.country.ch { background-position: 0px -1440px; }
```

Add a new line with your new locale before this line. Mind the comma:

```
.language-selector span.de_CH,
span.country.ch { background-position: 0px -1440px; }
```

Finally, open the file `instance-stuff/<project.name>/Germinate.gwt.xml` and add this line:

```
<extend-property name="locale" values="de_CH" />
```

After compiling and deploying the project, you should be able to select the new localization from the language selector at the top of the page.

6.2.1 Notes

- The default locale is determined by the browser settings. A fall-back solution can be specified by this entry in the `Germinate.gwt.xml` file:

  ```
  <set-property-fallback name="locale" value="en_GB" />
  ```

- Not all localization files have to be complete. If translations are missing, they will be substituted by their respective value from the default locale.

6.3 Internationalized files and images

In some cases it may be necessary to supply a download file or image in more than just one language. We decided to adopt a concept called *resource qualifiers*. This concept is also used in the popular Android platform [10].

The main idea of this concept is to provide resources in alternate forms which will be used under certain circumstances. On Android one example of a resource qualifier are language and region. Valid examples are *de* for German resources or *en-rGB* for British English.

The advantage of this concept is that you only have to copy your files/images to the appropriate folder and the system will choose the correct file based on the current configuration. If a file does not exist in the folder for the current configuration, the system will fall back to the default file.

We adopted this idea for Germinate. We allow internationalization of files in the directories "download", "data", and "res" (see Section 5 for details). To provide internationalized versions of your files, you need to create a new directory of the type you want to extend (e.g. "download") and then append the locale separated by a dash. In the previous sections we used the local `de_CH` as an example. The internationalized download directory for this locale would have to be named `download-de_CH`. The structure within this directory has to be identical to the base-directory ("download"). Please note, that this concept only works if the files in the different directories have the same file name. Otherwise, Germinate won't be able to find them.

The result for the user is a seamless internationalization of all the content within Germinate. When changing the locale on the web interface, the server backend will serve the internationalized files for this locale, if available, and fall back to the default locale otherwise.

6.4 Using the `ParameterStores`

The `TypedParameterStore` is an easy and convenient way to maintain a session state while navigating Germinate. This means that you can store selections the user makes on one page and reuse them on another page without having to pass them around. There is a typed version of the `TypedParameterStore` for each supported data type (`Long`, `String`, etc.). The important methods of the `ParameterStores` are:

```
public static T get(Parameter key);
public static T set(Parameter key, T value);
```

The first method will return the saved values for the given parameter (if available) whereas the second method stores a new value for the given parameter. Note that the return type of `get()` is `T`, meaning that each sub-class returns the data in the suitable data type.

The list of available parameters is defined as the enum `jhi.germinate.shared.enums.Parameter`. New parameters can be added by simply adding a new value to the enum.

6.4.1 Passing parameters via the URL

When Germinate is first loaded by the browser, it will parse the given URL parameters and save the respective parameters in the `ParameterStores`. As a result, you can create URLs that take the user to a specific page for a specific combination of parameters. As an example, the URL

```
http://<your_server>:8080/<project.name>/?accessionId=1#passport
```

will take you to the passport page of Germinate showing the accession with id 1. Alternatively, the URL

```
http://<your_server>:8080/<project.name>/?searchString=Baz#search
```

will show the search results for the given search string. Multiple parameters can be specified by combining them with an ampersand (&).

For security reasons, we only save valid parameters, i.e. parameters that are part of the enum `Parameter`. Moreover, it is very important that the parameters are located **before** the fragment identifier (#).

6.5 Using the notification system

The Notification class (`jhi.germinate.client.util.Notification`) contains the following methods to create notifications:

```
Notification.notify(Type.SUCCESS, "Message");

Notification.notify(Type.INFO, "Message");

Notification.notify(Type.WARNING, "Message");

Notification.notify(Type.ERROR, "Message");
```

Methods taking two parameters allow using an arbitrary title, whereas methods with just one parameter will use the default title defined in the localized properties files.

6.6 Using D3.js

D3.js [11] (D3 for Data-Driven Documents) is a JavaScript library used to create interactive data visualizations in the browser backed by Scalable Vector Graphics (SVG). D3.js is a highly complex framework and creating new charts can be tricky at times. This is why we integrated reusable versions of a set of frequently used chart types into Germinate. Those include: scatter plots, histograms, bar charts, line charts, treemaps, etc.

Using these reusable charts is straightforward. In this section, we will create a scatter plot as an example. The following code snippet is used to create the chart:

Table 3: Input data

id	name	x	y
3	CACTUAR-000005	4.08695	0.465969
3	CACTUAR-000005	3.55	0.451295
3	CACTUAR-000005	4	0.452107
3	CACTUAR-000005	3.7	0.474437
⋮	⋮	⋮	⋮

```
1  d3.tsv('scatter-plot.tsv', function (data) {
2      d3.select('#scatter-plot')
3          .datum(data)
4          .call(scatterPlot()
5              .margin(margin)
6              .width(width)
7              .height(height)
8              .x(function (d) { return parseFloat(d.x); })
9              .y(function (d) { return parseFloat(d.y); })
10             .colorKey(function (d) { return d.name; })
11             .tooltip(function (d) { return d.name + '<br/>(' + d.x + ', ' + d.y + ')'; })
12             .onClick(function (d) { console.log(d); })
13             .showDistribution(true)
14             .xLabel('weight [g]')
15             .yLabel('diameter [mm]')
16         );
17 });
```

Now, let's explain what this code actually does. In line 1, we tell D3.js to load a file called
`scatter-plot.tsv` (see Table 3). A `tsv` file is a file of tab-separated values (like a `csv` only
with tabs instead of commas). The result is passed to the following function as the variable
`data`.

In line 2, we select the HTML element with the id "`scatter-plot`", then use our data on
this element and call the function `scatterPlot`. The code so far would be enough to create a
scatter plot if our data was formatted in the format expected by the reusable chart. Lines 5 to
7 define the margin, width and height of the chart. Please note, that we omitted the definition
of these variables here. Width and height are integer values, whereas margin is an object con-
taining the values `top`, `right`, `bottom` and `left`. Lines 8 and 9 tell the chart how to extract
the values for x and y axis respectively. As you can see, they use `d.x` and `d.y` as the values are
defined in the x and y columns of the input file. Line 10 is used to define which column of our
input data should be used to determine the color of the data point. In this example, we want
all the points of the same accession to have the same color. The accession name is defined in
column `name`, so we return it. Line 11 extracts the information to show inside the tooltip when
the user hovers over the data point. Line 12 can be used to implement on-click behaviour. This
could be a simple popup message or a completely new page with additional information about
the clicked element. Line 13 enables the distribution visualization along the sides. Lines 14 and
15 define the axis titles for x and y axis respectively.

Using just these 17 lines, we can create the interactive scatter plot shown in Figure 13. Just
to put this number into context: The implementation of the reusable `scatterPlot` function is
roughly 350 lines long. This, however, includes a multitude of configuration options that aren't
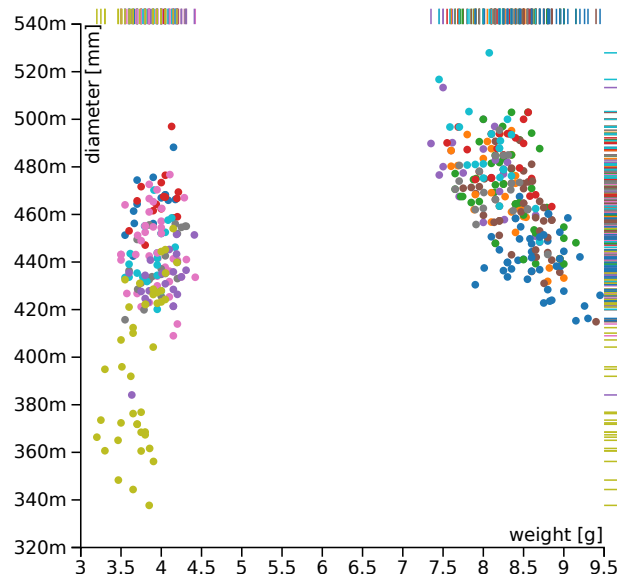used in this example.

Figure 13: D3.js scatter plot

6.7 Important notes

We try our best to make Germinate as secure as possible. The code you download from our page has been tested intensively to ensure that your data and your users are safe and secure.

That said, Germinate is only as secure as its individual parts. Should you decide to extend Germinate and add your own code to it, it is in your responsibility to ensure the security of your version of Germinate.

In this section, we will explain what types of security/safety issues can arise when developing with GWT and JavaScript and we will cover solutions for the most common issues. Please refer to the official documentation [12] for a complete list.

6.7.1 XSS and GWT

Cross-site scripting (XSS) is a security vulnerability that enables attackers to inject malicious scripts into the web page. GWT prevents XSS very well **if used correctly**. XSS can be avoided if you follow good JavaScript coding practices. Here is a list of things to keep in mind:

Avoid using `new HTML(String)` and `HTML#setText(String)`
> The class `com.google.gwt.user.client.ui.HTML` is a popular way of adding HTML code in the form of a `String` to your page. This is perfectly fine if you know where the `String` came from and what it contains, or in other words, if you are certain that the `String` doesn't contain malicious code. However, if the `String` is passed down to your code as a parameter and you don't know it's origin, this may entail a security issue.

There are ways to prevent this, however:

- Prefer plain-text widgets, e.g. `Label`. If your text does not contain HTML tags, it's preferable to use a `Label` instead.

- Change your parameter type from `String` to `SafeHtml`. `SafeHtml` represent XSS-Safe HTML and it's therefore safe to use it. The easiest way to use `SafeHtml` is to change the return type of the internationalization methods seen in Section 6.1 from `String` to `SafeHtml`. Refer to the official documentation to find out how to create instances of `SafeHtml` without this approach.

Avoid using the JavaScript function `eval()` in JSNI methods
The JavaScript Native Interface (JSNI) allows you to define JavaScript code embedded in Java code. This can be useful in cases where the default GWT implementation just doesn't provide the JavaScript functionality you require. When using JSNI, you of course can't rely on GWT to take care of XSS prevention. Consequently, be very careful when using JSNI. One of the most common mistakes is calling the JavaScript function `eval()`. This function takes a `String` and executes it as if it were regular JavaScript. This already screams "Security Issue".

Avoid code that sets `innerHTML` of GWT widgets
This is related to the first item in this list. Manipulating `innerHTML` of GWT widgets circumvents the XSS security mechanisms of GWT and therefore is not recommended. Setting `innerText` (plain-text) on the other hand is safe.

6.7.2   XSRF and GWT

Cross-Site Request Forgery (XSRF) is a type of attack where the attacker can perform actions on behalf of a verified user without the user's knowledge. The attacker uses the valid user session to start these requests to the server. Without any further measures, JavaScript and therefore GWT are vulnerable to XSRF attacks.

However, we included a mechanism in Germinate that prevents XSRF attacks. This mechanism requires you to send the session id with **every** request sent to the server. The server can then verify this session id (token) against the actual id of the session and the cookie. Only if all three ids are identical will Germinate return the requested data or execute the query. This security mechanism has already been covered in detail in Section 4.7.
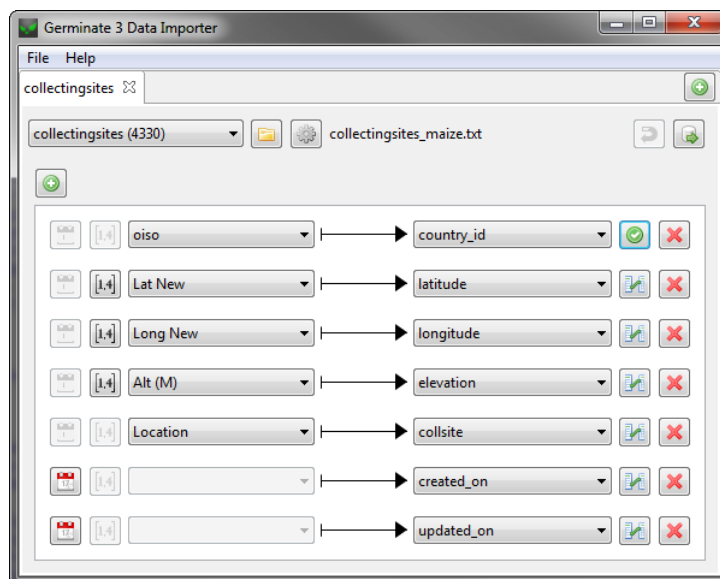
Figure 14: G3DI - Column mapping

7 Data Import

There are several ways to import data into Germinate. Some people may prefer to write their own scripts to import the data whereas others may prefer database management tools like *Navicat* [13], *MySQL Workbench* [14] etc.

However, we believe that we can make the import process even easier and so we developed a Java based desktop application to aid in the upload of experimental datasets into Germinate. The aim of this tool is to simplify the process of entering data into the system. *Germinate 3 Data Importer* (G3DI) allows permitted MySQL system users to upload data in text file format into a specified Germinate installation. We have also included tools to revert the most recent change should any problems arise in the data upload process.

The Germinate 3 Data Importer can be downloaded here:

<p style="text-align:center">http://ics.hutton.ac.uk/g3di/</p>

Figure 14 shows the main GUI of G3DI with an example mapping. The current tab is used to import data from a text file with the name `collectingsites_maize.txt` into the database table `collectingsites`. Each row represents a mapping between column of the input file and column of the database. G3DI is aware of the data types of the database columns and provides functionality to accommodate dates, number formats and foreign keys. The first row of the mapping maps `oiso` (which is a 3-digit country code) to `country_id`. G3DI knows that `country_id` is a foreign key column and requests the user to define a lookup table and column. This lookup is used to find the foreign id based on the value in the `oiso` column.

As an example, let's assume the lookup table is `countries` and the lookup column is `country_code3` and the value in the column `oiso` is "GBR". G3DI will then query the database and determine the id of the country with the 3-digit code "GBR" (which is the United Kingdom) and use this id instead of the actual value.

In addition you can define a number format (or to be exact, a locale) for decimal columns. This format is used to parse your input data before inserting it into the database.

Finally, you can define date formats for columns that represent dates. As an example, let's say your input data is in the form `dd/MM/yyyy` (2-digit day, 2-digit month, 4-digit year). G3DI

will then parse your dates before inserting them into the database. This is necessary, since the database uses its own date format which may differ from the format you used for your data.

After you imported your data, you might realize that your mapping was incorrect. G3DI can undo the most recent import step with one click of a button, to allow you to do your work without worrying that you might mess up the database. If you want to be on the safe side, you can still make a backup of the whole database with your favourite database management tool.

Database	Accessions	Markers	Collectingsites	Genotypes	Phenotypes
Germinate Maize	22022	37812	9714	0	33
Germinate Wheat	9811	5652	109	0	262
Germinate Breeders	829	4621	1	4866048	262
Germinate Demo	100	241	76	24100	7
Germinate CPC	1499	0	457	0	23
Germinate Grasses	167	3884	28	644744	3
Germinate Pea	3693	1469	213	244496	0

Figure 15: Statistics overview table

8 Statistics

We maintain a Germinate statistics page on our server. It shows an overview of all Germinate instances that we currently maintain. It's available at:

<div align="center">

`http://ics.hutton.ac.uk/germinate/germinate-statistics`

</div>

Figure 15 shows one of the visualizations we provide. We're planning to add more visualizations in the future. Now you might think: "What does that have to do with me?".

The answer is simple: We can add your instance of Germinate to our statistics website. That way, people browsing our website may stumble across your site which can broaden your potential target audience. In addition, access to statistical data from more Germinate instances allows us to improve Germinate and the statistics page. So it's basically a win-win situation.

If you are interested and want us to add your Germinate instance to our statistics page, or if you have any questions, drop us an email: `germinate@hutton.ac.uk`.

Just in case you are wondering what kind of data we would collect from your Germinate instance, let me go into detail here: When you start Germinate on your server, it creates a couple of views. These views contain the aggregated data. You can examine these views with your favourite MySQL tool. Let me stress, that these views by themselves are not visible to anyone but you, so there is nothing to worry about.

For us to be able to use these views, you would need to create a new database user that has `select` and `show view` privileges on the views. Please note, that this user will not have select access to the database tables, so your actual data is safe. We would then store the user credentials in our database and run a query against your views once a day. Consequently, the data we show on our statistics page is always up to date.

9 Troubleshooting

9.1 Germinate links in download files are not HTTPS

Files downloaded from Germinate sometimes contain links back to certain pages back on Germinate. This is used so that external tools can come back to Germinate for additional information.

In case you're running Germinate over HTTPS (which you should) and are also using a reverse proxy to forward requests to Tomcat, it might happen that these links don't point to `https://your-server.com/germinate` but rather to `http://your-server.com/germinate`. In most cases, this will be fine as the user will be redirected to HTTPS and won't even notice that this happens. Some external tools might not be able to handle this redirect though.

The problem here is that Tomcat doesn't know that the request came to an HTTPS resource originally (before it got proxied) and Germinate therefore doesn't have sufficient information to generate the links correctly.

To fix this, open the `server.xml` file under Tomcat's `conf` folder and add the following `Connector` to the `Server`:

```
<Connector port="8081" protocol="HTTP/1.1"
    connectionTimeout="20000"
    proxyName="<proxy server name, e.g. baz.hutton.ac.uk>"
    proxyPort="443"
    scheme="https"
    secure="true"/>
```

In this case, `proxyName` is the name of the proxy server. In addition, you need to change the "proxy reverse" settings of Apache. Open the file containing your configuration and identify the entries responsible for Germinate. They may look something like this:

```
ProxyPass          /germinate-template http://tomcat-server:8080/germinate-template
ProxyPassReverse   /germinate-template http://tomcat-server:8080/germinate-template
```

Change the port from 8080 to 8081 (as configured on the `server.xml`) so that it looks like this:

```
ProxyPass          /germinate-template http://tomcat-server:8081/germinate-template
ProxyPassReverse   /germinate-template http://tomcat-server:8081/germinate-template
```

Restart both Apache and Tomcat for this fix to work.

10   Funding

Germinate 3 was developed with CIMMYT [15] as part of the MasAgro Seeds of Discovery Project [16].

11 Technologies

Germinate uses third-party software libraries to extend its functionality. The following sections will list these libraries and explain their purpose:

11.1 JavaScript libraries

CookieCuttr
Because of the EU "Cookie Law", websites have to let the user know when they're using cookies. CookieCuttr displays a banner notifying the user of just that.
`https://github.com/weare2ndfloor/cookieCuttr`

D3.js
We us D3.js to generate dynamic and interactive data visualizations. Almost all charts in Germinate are created using this library.
`https://github.com/mbostock/d3`

d3Pie
This is a plugin for D3.js that we use to easily create pie charts.
`https://github.com/benkeen/d3pie`

D3.tip
This is a plugin for D3.js that allows easy creation of tooltips for SVG elements.
`https://github.com/Caged/d3-tip`

FancyBox
FancyBox is used to create popups of images and text.
`https://github.com/fancyapps/fancyBox`

html2canvas
html2canvas is used to convert the d3.js chart legend to an image and download it.
`https://github.com/niklasvh/html2canvas`

jQuery
jQuery is the most popular JavaScript library out there. It allows easy selection, traversal and manipulation of DOM elements.
`https://github.com/jquery/jquery`

jQuery.freezeheader
We use a highly customized version of this library to make our table headers "freeze", i.e. when you scroll down the page, the table header will stick to the top of the window once you scroll past it.
`https://github.com/laertejjunior/freezeheader`

lasso.js
This is a plugin for D3.js that allows freehand drawing of selections within D3.js charts.
`https://github.com/skokenes/D3-Lasso-Plugin`

MarkerClustererPlus
We us this library to group large numbers of markers on a Google Maps into a smaller number of clusters. This emphasizes regions with many markers without cluttering the map with individual pins.
`https://github.com/googlemaps/js-marker-clusterer`

OverlappingMarkerSpiderfier

This spiderfier is used to allow easy selection of pins on a Google Map with a high density of markers. When clicked, the pins will be spread out so that the user can then select the marker of interest.
`https://github.com/jawj/OverlappingMarkerSpiderfier`

prettify.js

Prettify is used to make code on website more pretty, e.g. it adds basic syntax highlighting. The main use of this library is for the SQL debug messages that appear when Germinate is run in debug mode.
`https://github.com/google/code-prettify`

saveSvgAsPng.js

This library allows the user to save a SVG image in the browser (usually generated by D3.js) to a PNG file on their computer.
`https://github.com/exupero/saveSvgAsPng`

Spectrum

Spectrum allows us to easily add color-pickers to the web-interface. HTML5 introduced the color-type input element, but many current browsers still don't support it. We also always have to consider users with outdated browsers, so unfortunately we have to fall back to a third-party library for this purpose.
`https://github.com/bgrins/spectrum`

toastr

Toastr is a library that allows us to show notifications similar to Gnome / Growl notifications on the desktop.
`https://github.com/CodeSeven/toastr`

11.2 Font frameworks

Font Awesome

This is a truly awesome framework containing loads of pictographic icons that we use throughout Germinate. They are fully scalable which makes them a perfect replacement for raster images.
`https://github.com/FortAwesome/Font-Awesome`

11.3 Java libraries

Apache Commons FileUpload

This library is used to make the upload of files from the browser to the server easier.
`https://github.com/apache/commons-fileupload`

Apache Commons IO

This is a widely used utility library.
`https://github.com/apache/commons-io`

Apache Commons Logging

A popular logging library.
`https://github.com/apache/commons-logging`

Apache HttpComponents Client

This library is used to easily communicate with Gatekeeper via HTTP requests.
`https://github.com/apache/httpclient`

Apache HttpComponents Core

Required by Apache HttpComponents Client.
`https://github.com/apache/httpcore`

Flapjack

Flapjack is a graphical genotype viewer developed at The James Hutton Institute. Germinate uses it on the server side to export genotypic data and allele frequency data.
`https://ics.hutton.ac.uk/flapjack`

Flyway

Flyway is a database migration tool that Germinate uses to update its database between releases.
`https://flywaydb.org`

GWT

GWT is the main building stone of Germinate. It's the web development framework we chose to use.
`https://github.com/gwtproject/gwt`

GWT-Charts

This is a GWT implementation of the JavaScript Google Charts API.
`https://code.google.com/p/gwt-charts`

GWT-Maps-V3-Api

This is a GWT implementation of the JavaScript Google Maps API.
`https://github.com/branflake2267/GWT-Maps-V3-Api`

GwtQuery

We use this library to use jQuery-like code in Java.
`https://github.com/ArcBees/gwtquery`

GWT-Tour

This is a GWT API for the hopscotch framework. It's used to generate "tours" for users, i.e. step-by-step instructions for using the web-interface.
`https://github.com/eemi2010/gwt-tour`

jBCrypt

jBCrypt is a Java implementation of the OpenBSD's Blowfish password hashing algorithm.
`https://github.com/jeremyh/jBCrypt`

JAK

This "Java API for KML" is used when exporting geographic information to KML format.
`https://github.com/micromata/javaapiforkml`

JAXB

Required by JAK.
`https://jaxb.java.net/`

MySQL Connector/J

This is a library that allows us to easily communicate with a MySQL database from Java code.
`https://github.com/mysql/mysql-connector-j`

Thumbnailator

This is a utility library used to create thumbnails of images. Whenever a new image is copied to the full-size image folder of Germinate, we will use this library to automatically generate a thumbnail for it.

`https://github.com/coobird/thumbnailator`

References

[1] I. Milne, P. Shaw, G. Stephen, M. Bayer, L. Cardle, W. T. Thomas, A. J. Flavell, and D. Marshall. Flapjack - graphical genotype visualization. *Bioinformatics*, 26(24):3133–3134, 2010.

[2] The EU cookie law (e-Privacy Directive). `http://ico.org.uk/for_organisations/privacy_and_electronic_communications/the_guide/cookies`. Last accessed on 2014-08-05.

[3] Toastr - Simple javascript toast notifications. `http://toastrjs.com`, 2012. Last accessed on 2013-09-24.

[4] HeatmapLayer for Google Maps v3. `https://developers.google.com/maps/documentation/javascript/3.exp/reference#HeatmapLayer`. Last accessed on 2013-09-19.

[5] MarkerClusterer for Google Maps v3. `http://google-maps-utility-library-v3.googlecode.com/svn/trunk/markerclusterer/docs/reference.html`, 2010. Last accessed on 2013-09-19.

[6] Google Earth. `http://www.google.com/earth/`, 2005. Last accessed on 2013-09-25.

[7] JDK 7 and JRE 7 Supported Locales. `http://www.oracle.com/technetwork/java/javase/javase7locales-334809.html`. Last accessed on 2013-09-20.

[8] I18n Plural Forms. `http://www.gwtproject.org/doc/latest/DevGuideI18nPluralForms.html`. Last accessed on 2013-09-23.

[9] List of Unicode characters. `http://en.wikipedia.org/wiki/List_of_Unicode_characters`, 2007. Last accessed on 2013-09-23.

[10] Android Resource Qualifiers. `http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources`. Last accessed on 2014-09-07.

[11] D3.js - Data-Driven Documents. `http://d3js.org/`, 2011. Last accessed on 2014-09-15.

[12] GWT Security Documentation. `http://www.gwtproject.org/doc/latest/DevGuideSecurity.html`. Last accessed on 2014-08-18.

[13] Navicat. `http://www.navicat.com/`, 2002. Last accessed on 2014-07-23.

[14] MySQL Workbench. `http://mysqlworkbench.org/`, 2005. Last accessed on 2014-07-23.

[15] CIMMYT: International Maize and Wheat Improvement Center. `http://www.cimmyt.org/`. Last accessed on 2014-06-26.

[16] Seeds of Discovery: Unlocking the genetic potential of maize and wheat. `http://seedsofdiscovery.org/`, 2012. Last accessed on 2014-06-26.