University of York optional module allocation software: documentation

Alex Muller (07525 370 388)

November 2011–March 2012

Contents

1 Introduction

This web application was created to allocate optional modules automatically to students at the University of York.

The software was originally written by Alex Muller (`http://alex.mullr.net/`) as his final year Computer Science and Maths BSc project.

2 Members of staff in academic departments

2.1 Spring 2012 pilot by Archaeology and History departments

The following notes are applicable during the Spring 2012 pilot of the application. Please note that the application is quite frail at this point, and may at times display unintelligible errors.

2.1.1 Accessing the application

The application will only be accessible when on campus (either using an IT Services PC or the student's own device). "Campus" denotes anywhere IT Services provides access, including the library, King's Manor and student bedrooms. We're being cautious to limit the effect of any bugs in the software.

If you like, you can let your students know that they can use the Virtual Private Network (VPN) to access the application while off campus: `http://www.york.ac.uk/it-services/connect/vpn/` or `https://webvpn.york.ac.uk`. The hope is that students should be on campus at least once during the module allocation period, so this won't be too inconvenient.

2.1.2 Administration

During the pilot period, there may be some parts of the administrative interface where staff in a department can view or modify information belonging to another department.

While the application is restricted to just the Archaeology and History departments, I don't see this as being a problem - please don't edit each others' modules or students! This will obviously need to be improved before the application can be used more widely.

2.1.3 Setting up the system

Data should be constructed locally by departmental administrators. For example, an Excel file (or similar) should be created with the following data:

- Sheets: Route code, stage, help text
- Modules: Module code, module name, minimum size, maximum size, URL (can be blank), credits
- Groups: Route code, stage, group title, help text, credits to allocate, modules (a space-separated list of module codes)
- Students: Username, route code, stage

This data should be exported to four separate CSV files. The structure of the CSV files is quite important (e.g. do not include any blank lines in the document – this is what I mean by the software being quite frail).

You will want to remove curly quotes from your Excel document - these will not break the system, but will not display correctly to users.

2.1.4 Limitations of the system

Uniqueness of (student, module) pairs

Currently, it is not possible to display the same module to the same student in two different groups. Modifications will need to be made to the software to accommodate this.

Joint students where both departments use this system

If a student is on a joint course and both departments are using this software, please consider the following:

- Only one department should add a "sheet" for that group of students
- Both departments can upload their respective modules and module groups
- Only one department should upload details of that student (their username)

2.1.5 Ranking on behalf of students

You can rank modules on behalf of a student by visiting the administration section of the application and selecting "Make a ranking on behalf of a student".

An example link to rank `am639`'s modules would be `/admin/rank/am639`.

2.1.6 The allocation

To make an allocation, view the table of all allocations – at the bottom is a button which will delete all existing allocations and re-allocate modules.

If you receive an error, please check your students' elective credit values. These are what is most likely to break the allocation.

If these all look correct (no odd numbers of credits), consult Alex and he'll try to diagnose the problem.

3 IT Services

This is a Java application, written using Spring 3. It can be deployed to Apache Tomcat 7, using Apache Maven 3.0.3 (see `pom.xml` in the root).

The scaffolding was originally created using Spring Roo 1.1.5. Roo can be removed fairly easily – see the reference docs at `http://static.springsource.org/spring-roo/reference/html/index.html`.

- Views: Apache Tiles
- Persistence: Hibernate/JPA
- Security: Spring Security 3.0.5

3.1 Data in and out

When the application is first deployed, no data is present. The application does not connect to external sources to gather data (such as the Data Warehouse).

Firstly, a system administrator (an IT Services developer) will add the two pilot departments to the database, and the usernames of one or more members of staff from each department.

Data is uploaded into the system by members of administrative staff in academic departments. Before the application is made accessible to students, departments upload or otherwise input:

1. **A list of students required to use the system:** username, route code, stage
2. **A list of modules available through the system:** module code, module name and various other module attributes
3. **"Sheets" and "groups":** these refer to the structure of a specific course (a sheet contains several groups and a group contains several modules)

None of the module information is sensitive - it is published externally on departmental websites. The student information is incredibly sensitive and will only be disclosed through the application to departmental administrators.

After the allocation is performed, the following data is exported from the system:

1. **A list of allocations:** username, route code, stage, module code, various other module attributes
2. **A list of choices:** a table mapping username to module, with every ranking allocated by a student to a module

A file containing the list of allocations is given by departments to SSDT, and they perform a bulk import of the data into SITS. The list of choices is maintained by the departments so that they have a record of students' choices when the application is no longer accessible. This data is held in line with the department's retention policies.

Once departments have exported all their data, the application no longer needs to be accessible online.

3.2 OWASP Top Ten

Threat: an attacker penetrates the system in order to compromise, disclose or destroy potentially sensitive data held by the system.

The application is restricted by the firewall such that it can only be accessed from a University IP address. It sits behind Shibboleth, so a valid University username and password are required to access any part of the application.

3.2.1 A1: Injection

All interactions with the persistent storage are carried out through Hibernate (HQL). Database queries use `TypedQuery<T>.setParameter()`.

3.2.2 A2: Cross-Site Scripting (XSS)

The only information persisted from the student interface is a number indicating their rank for that module. If a user attempts to submit a value not of type `short`, an error is raised and that data is not persisted.

Every variable displayed to students is passed through `<c:out />` to escape characters such as `<` and `>`.

Note: I would like to escape everything globally on output, but haven't found a way yet.

3.2.3 A3: Broken Authentication and Session Management

The application is served over SSL, so session cookies cannot be hijacked by a malicious third party. Authentication is handled by Shibboleth, in the same way as other SSO applications.

When the `optmodsdev.york.ac.uk` cookies are removed from the browser, the user is kicked back to `shib.york.ac.uk`. If the Shibboleth session is still live, the user is automatically logged in – otherwise, they have to re-authenticate with the University.

3.2.4 A4: Insecure Direct Object References

There are no insecure direct object references – for example, URLs are managed using Spring Security and are checked globally for the application.

All administrative functionality is kept under `/admin`, and all the student interface is kept under `/rank`. As such, administrative staff cannot access the student UI and students cannot access the administrative UI.

3.2.5 A5: Cross-Site Request Forgery (CSRF)

[I'm a little unsure on this one]

TODO: Add a CSRF token to at least the student `POST` request. Perhaps use `http://info.michael-simons.eu/2012/01/11/creating-a-csrf-protection-with-spring-3-1/` to automatically deal with all `POST` requests?

3.2.6 A6: Security Misconfiguration

OS and related security is handled by IT Services. The application uses the same main framework and version (Spring 3.0.5) as the Student Portal.

3.2.7 A7: Insecure Cryptographic Storage

Data is stored in an IT Services Oracle database.

3.2.8 A8: Failure to Restrict URL Access

URLs are restricted using Spring Security, with the following configuration (or similar):

```
<!-- HTTP security configurations -->
<http auto-config="true" use-expressions="true">
  <form-login default-target-url="/"/>
  <!-- Configure these elements to secure URIs in your application -->
  <intercept-url pattern="/admin/**" access="hasRole('ROLE_STAFF')"/>
  <intercept-url pattern="/rank/**" access="hasRole('ROLE_STUDENT')" />
  <intercept-url pattern="/resources/**" filters="none" />
  <intercept-url pattern="/**" access="authenticated" />
  <custom-filter ref="ShibbolethHeaderFilter" position="PRE_AUTH_FILTER"/>
</http>
```

This indicates that everything under `/admin` requires the user to be a staff member in this application's database. Everything under `/rank` requires the user to be a student in this application's database. Everything under `/` is accessible to every Shibboleth'd user.

As long as the application follows this pattern when designing URLs (which it does), there are no failures to restrict URL access.

3.2.9 A9: Insufficient Transport Layer Protection

The application is served over SSL. Google Chrome reports that all cookies set by this application are marked `Send For: Secure connections only`.

3.2.10 A10: Unvalidated Redirects and Forwards

User parameters are never used when calculating redirects or forwards (that is, all redirects in the application are to exactly known locations).

4 Contacts

- Laura Crossley (Academic Support Office, project manager)
- James Cussens (Computer Science, dissertation project supervisor)
- Matt Hanson (SSDT, SITS import)