Project Information

It is possible to create course and test timetables, manage modifications to these timetables, share rooms with other events, and schedule students to particular sessions using UniTime, a comprehensive educational scheduling system. It is a distributed system that enables various university and departmental schedule administrators to coordinate efforts to create and change a schedule that satisfies their various organisational needs while minimising student course conflicts. It can be used independently or in conjunction with an existing student information system to develop and maintain a school's course and/or exam schedule. Faculty, students, and staff from universities in North America and Europe worked together to design the system at first. . In the hopes that other colleges and universities would use the programme to improve their students' scheduling or wish to contribute to ongoing study in this field, it is made available for free under an open-source licence. In March 2015, the [Apereo Foundation] [apereo] began funding the UniTime initiative. Since we were initially unfamiliar with the system, we employed the bottom-up strategy. In this method, we first fixed errors and added minor system components to better understand the system as a whole.

Classes whose features we've added:

1)ExternalUidLookup

prior characteristics

search only using the searchID

We added new features like:

allows users to be looked up using their username and email, which is helpful in situations when the username can be used to identify a user specifically. If the user cannot be located or if there is a lookup issue, the procedure throws an exception instead of returning a UserInfo object for the relevant user.Jira

2) Security messages:

The optimized code adds explicit keys to each message string in the SecurityMessages interface. This is useful for localization purposes, as it enables translation of messages into different languages and cultures.

By providing a key for each message, the code can easily map the message string to a translated version of that message in the appropriate language. This makes the code more flexible and adaptable to different users and contexts.

It's also worth noting that the optimized code is identical to the original code in terms of functionality, and only differs in the addition of explicit keys to each message string.

3) ExternalUidLookup

The optimized version of the code has several changes to improve its readability, maintainability, and performance.

First, it adds three new lookup methods that allow searching for users by external ID, email, or username.

Second, it renames the doLookup method to lookup. This follows the standard naming convention for such methods.

Third, it simplifies the UserInfo class by using more descriptive variable names and removing redundant code. It also removes unnecessary null assignments in the constructor.

Fourth, it removes the Exception declaration from the lookup method's signature. Instead, any exceptions thrown by the method are handled by the calling code.

Overall, these changes make the code easier to understand, more flexible, and more efficient.

Github:

https://github.com/Beshoymourad/Maintenance.git