Coarse Grain Access Control Framework – CGA

Minor Project

IBM Career Education

Disclaimer

This Software Requirements Specification document is a guideline. The document details all the high level requirements. The document also describes the broad scope of the project. While developing the solution if the developer has a valid point to add more details being within the scope specified then it can be accommodated after consultation with IBM designated Mentor.

INTRODUCTION

The purpose of this document is to define scope and requirements of a Coarse Grain Access Control Framework (CGA) for the developers of a software services company, engaged in developing Java Servlet/JSP based web applications. The proposed system will eliminate the need to develop specific coarse grain access control module from scratch every time. CGA framework will provide a loosely coupled and easy to use “access control mechanism” driven by access control lists (ACLs).

This document is the primary input to the development team to architect a solution for this project.

System Users

The entire team of developers of the software services company is expected to benefit from the coarse grain access control framework, CGA.

Assumptions

1. A valid “user object” is available in the session and it contains the logged-in user-id and the group membership(s) that the logged-in user has.
2. The tables, GROUPS and GROUP_MEMBERSHIP are already available in the database. GROUPS table has only one field viz. “group name” (PK). GROUP_MEMBERSHIP table has 2 fields viz. (a) “group name” and (b) “user-id”. The PK in this case is “group name” + “user-id”. GROUPS and GROUP_MEMBERSHIP tables enjoy a parent child relationship. Each authorized user of the application will be a member of one or more groups. For testing purpose, you may populate these 2 tables with some meaningful data from the backend directly.
3. It is assumed that the standard Java EE Filter Mechanism will be used for the development.

REQUIREMENTS

CGA will provide a very simple and loosely coupled mechanism to add coarse grain access control capabilities to any new web application, especially RESTful web applications. Learn more about REST at http://en.wikipedia.org/wiki/Representational_state_transfer URL.

The access rights will be assigned to a user through groups. It will be accomplished by mapping a group to the set of URIs that the members of that group can access. Therefore, for a user to be able to access an URI, s/he must be a member of at least one of the corresponding group(s). This mapping will be maintained in a table called “ACL”. A well-designed form and view will be required to allow management of ACL table.

CGA will leverage the Java EE standard “filters” to build this framework. The overall flow is outlined below:

1. The logged-in user attempts to access a page (URI) in the web application;
2. CGA “filter” intercepts the logged-in user’s request;
3. CGA filter now retrieves the logged-in user’s “user object” from the session; and obtains the list of all the group(s) that the user is a member.
4. A lookup using the URI is performed in ACL table to extract the list of mapped group(s). If no groups are found, a HTTP 401 error is raised. Otherwise (i.e. one or more groups found) the logged-in user is allowed to access the requested page.

The CGA framework will consists of 2 key artifacts viz. (a) a filter to intercept requests to pages of the web application, and (b) ACL table. The ACL table will contain “group name” and “application URI” pairs.

Note CGA can control much more finer access control for a RESTful web application compared to an old-fashioned web application.

As an optional requirement, a complete user interface to manage GROUPS and GROUP_MEMBERSHIP tables may be created.

Testing Guidelines

Create a simple test bed consisting of simple RESTful looking web pages to test the framework.

DEVELOPMENT ENVIRONMENT

CGA will be developed as a web application using Java/JSP/Servlets/Filters and DB2 database. Eclipse will be used as the IDE for the same. You may consider using a JavaScript framework like jQuery/Prototype/ Scriptaculous.

Details about Java EE 6 Filters can be found at http://docs.oracle.com/javaee/6/tutorial/doc/bnagb.html URL.