[GK] 6a.2.1: RBAC

Done: View To do: Make a submission

Opened: Sunday, 5 March 2023, 12:00 AM
Due: Friday, 17 March 2023, 12:00 AM

Role Based Access Control

GitHub Classroom

Erstelle für deine Aufgabe ein Git Repository und nutze folgenden GitHub Classroom Link: https://classroom.github.com/a/glxSzO-L

Hintergrund

Du bist Teil eines Softwareentwickler:innen-Teams, das gerade eine größere Serverapplikation entwickelt, das in Zukunft von verschiedensten Gruppen von Nutzer:innen verwendet werden soll.

Da in dem System auch sensible Daten gespeichert werden, soll sichergestellt werden, dass Nutzer:innen nur jeweils auf jene Teile der Anwendung Zugriff haben, die sie für ihre Arbeit benötigen. Es liegt nun an dir, für diesen Zweck ein einfaches System für die Rollenbasierte Zugriffskontrolle (engl. Role Based Access Controll; RBAC) zu implementieren.

Das Zugriffssystem

Das Zugriffssystem besteht aus drei Hauptkomponenten:

- **User**: Sind die Nutzer:innen des Systems. Für der Zugriffskontrolle möchte also der:die Nutzer:in Zugriff auf eine Ressource erhalten.
- **Resource**: Ressourcen sind eine Abstraktion aller Funktionen des Systems. Je nach konkreter Anwendung könnte das eine Datei, ein Verzeichnis oder ein Dienst sein.
- **Role**: Anstatt Nutzer:innen direkt zu Ressourcen zuzuweisen (was bei großen Anwendungen mit einer vielzahl an Nutzer:innen und Ressourcen zu einem unhandhabbaren Organisationsaufwand führt), stellen in der Rollenbasierte Zugriffskontrolle Rollen das Bindeglied zwischen Nutzer:innen und Ressourcen dar. Dabei hat ein:e Nutzer:in ein oder mehrere Rollen, in dessen Rahmen er:sie mit der Anwendung interagieren kann. Umgekehrt hat eine Ressource potentiell mehrere Rollen, die einen Zugriff auf ebendiese Ressource zulassen. Beispiele für Rollen sind z.B. Gast, Mitarbeiter und Administrator.

Umsetzung

Setze das oben beschriebene Zugriffssystem in Java um. Beachte bei der Implementierung folgendes:

- `User` und `Resource` sollen als Java-Klasse umgesetzt werden. Die sollen jeweils Methoden zum Hinzufügen, Entfernen und Auflisten der Rollen haben. Außerdem soll `Resource` eine Methode haben, welche überprüft, ob ein:e bestimmte:r Nutzer:in (mittels ihrer:seiner Rollen) Zugriff auf die jeweilige Ressource hat.
- Nutze für die Speicherung der Rollen in `User` und `Resource` ein geeignetes Set. Beim Abgabegespräch solltest du dazu erklären können, warum für diesen Fall ein Set die richtige Datenstruktur ist!
- `Role` soll als Interface umgesetzt werden und Methoden zur Rückgabe eines Namens und einer textuellen Beschreibung der Rolle haben. Implementiere zusätzlich mindestens drei konkrete Rollen (also Klassen, die das `Role` Interface implementieren). Rollen werden durch ihren Namen eindeutig identifiziert. Vergiss nicht, die `hashcode` und `equals` Methoden richtig zu implementieren!
- Teste dein Programm anschließend in mit den verschiedensten Konstellationen (Zugriff erlaubt / verweigert).

Anforderungen

- Vollständiges UML-Klassendiagramm ist vorhanden (erstelle dieses idealerweise **bevor** du mit dem Programmieren beginnst!)
- Sourcecode ist sinnvoll dokumentiert
- Entwicklung findet im Git-Repository statt (sinnvolle Commitstruktur)
- Anforderungen wie oben beschrieben umgesetzt
- Sets und deren Methoden sinnvoll eingesetzt

Die Abgabe erfolgt durch Hochladen des Source Codes als zip **und** Angabe des GitHub-Links.

EK

Dieses einfache Rollensystem soll nun so erweitert werden, dass auch komplexere Strukturen abgebildet werden können. Neben einer Menge an Rollen, die alternativ erfüllt sein können (dh. der:die Benutzer:in muss eine der Rollen besitzen, damit ein Zugriff erlaubt wird), soll es auch möglich sein, kumulative Voraussetzungen abzubilden (der:die Benutzer:in muss alle der angegebenen Rollen besitzen).

Ein Lösungsansatz dafür wäre, dass eine Ressource statt einer Menge an Rollen als Zugriffsvoraussetzung eine Menge mit Mengen an Rollen hat (also `Set<Set<Role>>` statt `Set<Role>`), wobei die inneren Mengen jeweils die kumulativ vorausgesetzten Rollen abbilden.

Es steht dir allerdings frei, deinen eigenen Lösungsweg zu wählen, sei in diesem Fall allerdings beim Abgabegespräch darauf vorbereitet, die Vor- bzw Nachteile deiner Lösung zu argumentieren.

Submission status

Submission status	No submissions have been made yet
Grading status	Graded
Time remaining	Assignment is overdue by: 329 days 17 hours
Last modified	-
Submission comments	Comments (0)

Grading criteria

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *2 points*	Funktion vollständig *4 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
Set Verwendung	Grobe Fehler *0 points*	Leichte Fehler bei Verwendung von Sets *2 points*	Sets korrekt benutzt *4 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben *2 points*

Feedback

Grade	14.00 / 14.00
Graded on	Thursday, 30 March 2023, 8:27 AM
Graded by	HD Haselberger David
Feedback comments	Gut gemacht!

Grade breakdown

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *2 points*	Funktion vollständig *4 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
Set Verwendung	Grobe Fehler *0 points*	Leichte Fehler bei Verwendung von Sets *2 points*	Sets korrekt benutzt *4 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben *2 points*