EnumSet Efficiency

EnumSet is Specialized Set which only takes Enum elements. Look at the signature of this class.

***public abstract class EnumSet<E extends Enum<E>>  
extends AbstractSet<E>  
implements Cloneable,******Serializable***

The Beauty of EnumSet is we can perform any collection operations like (add,remove etc. ) on Enum as Enumset is a collection furthermore it provides some Static methods by which we can do union,range check,none etc. operation.

So by using Enumset we can dynamically add Enum element remove an element or perform union operation . we can use it in a efficient way and use it’s power but still Developer are not use this they put Enum in a Set or any other collection if required.

**So where we can Use EnumSet?**

Ummm a probable use case comes in my mind is ,

***Role and operation implementation not only that say from UI we can manipulate it and can create a new Role with new operation Set.***

To be specific say I have Three types of role say **Self,Friend and Guest.**

As you can guess Owner of the profile got Self role , Friends of owner got friend role and any unknown person got Guest role. Now Owner wants to create a new role for few guests , those can perform Friend operation as well as guest operations both.

How we will implement that,

Obvious one can think of Bit representation for operations and do an anding operation to check is particular operation is valid for a role or not, the same way he can create a new role I don’t go for that implementation that will hard to maintain. rather I will try to solve it through EnumSet.

Code:

1. I create a Operation Enum where I store all the possible Operations

package com.example.enumtest;

public enum Operation {

VIEW\_PROFILE,EDIT\_PROFILE,DELETE\_PROFILE,VIEW\_ALBUM,EDIT\_ALBUM,DELETE\_ALBUM,COMMENT,RATE\_PROFILE

}

2. Then I create a Role Enum and associate permitted operations with those role.

package com.example.enumtest;

import java.util.EnumSet;

import com.sun.xml.internal.ws.policy.privateutil.PolicyUtils.Collections;

public enum Role {

SELF(EnumSet.of(Operation.VIEW\_PROFILE,Operation.VIEW\_ALBUM,Operation.EDIT\_PROFILE,Operation.EDIT\_ALBUM,Operation.DELETE\_PROFILE,

Operation.DELETE\_ALBUM,Operation.COMMENT)),

FRIEND(EnumSet.of(Operation.VIEW\_ALBUM,Operation.VIEW\_PROFILE,Operation.COMMENT)),

GUEST(EnumSet.of(Operation.VIEW\_PROFILE,Operation.RATE\_PROFILE));

EnumSet<Operation> operationSet;

Role(EnumSet<Operation> operationSet){

this.operationSet=operationSet;

}

}

Look in Role Constructor I pass EnumSet<Operation>, and when we define the Enum element I called a Static method ***of on EnumSet.*** Which will perform an union operation.

3. Now each role has its set of operations and we can easily print them by traversing on operationSet ,easily check an operation is permitted or not by using

**operationSet.contains(operation)**

4. Now I want to add a new Role which can perform both Friend and guest operations.

package com.example.enumtest;

import java.util.EnumSet;

public class ProfileManager {

public static void printOperationForCustomRole(EnumSet<Role>roles)

{

EnumSet<Operation> mergeSet = EnumSet.noneOf(Operation.class);

for(Role role: roles)

{

for(Operation op : role.operationSet)

mergeSet.add(op);

}

System.out.println(mergeSet);

}

public static void main(String[] args) {

ProfileManager.printOperationForCustomRole(EnumSet.of(Role.FRIEND,Role.GUEST));

}

}

Look , how easily we can do it just pass Role.Friend and Role.Guest in printOperationForCustomRole() role method and create a new Operation set which holds all the operations of friend and Guest.

Output:

[VIEW\_PROFILE, VIEW\_ALBUM, COMMENT, RATE\_PROFILE]