java.security

Class PermissionCollection

java.lang.Object
    java.security.PermissionCollection

All Implemented Interfaces:

Serializable

Direct Known Subclasses:

Permissions

```
public abstract class PermissionCollection
extends Object
implements Serializable
```

Abstract class representing a collection of Permission objects.

With a PermissionCollection, you can:

- add a permission to the collection using the `add` method.
- check to see if a particular permission is implied in the collection, using the `implies` method.
- enumerate all the permissions, using the `elements` method.

When it is desirable to group together a number of Permission objects of the same type, the `newPermissionCollection` method on that particular type of Permission object should first be called. The default behavior (from the Permission class) is to simply return null. Subclasses of class Permission override the method if they need to store their permissions in a particular PermissionCollection object in order to provide the correct semantics when the `PermissionCollection.implies` method is called. If a non-null value is returned, that PermissionCollection must be used. If null is returned, then the caller of `newPermissionCollection` is free to store permissions of the given type in any PermissionCollection they choose (one that uses a Hashtable, one that uses a Vector, etc).

The PermissionCollection returned by the `Permission.newPermissionCollection` method is a homogeneous collection, which stores only Permission objects for a given Permission type. A PermissionCollection may also be heterogeneous. For example, Permissions is a PermissionCollection subclass that represents a collection of PermissionCollections. That is, its members are each a homogeneous PermissionCollection. For example, a Permissions object might have a FilePermissionCollection for all the FilePermission objects, a SocketPermissionCollection for all the SocketPermission objects, and so on. Its `add` method adds a permission to the appropriate collection.

Whenever a permission is added to a heterogeneous PermissionCollection such as Permissions, and the PermissionCollection doesn't yet contain a PermissionCollection of the specified permission's type, the PermissionCollection should call the `newPermissionCollection` method on the permission's class to see if it requires a special PermissionCollection. If `newPermissionCollection` returns null, the PermissionCollection is free to store the permission in any type of PermissionCollection it desires (one using a Hashtable, one using a Vector, etc.). For example, the Permissions object uses a default PermissionCollection implementation that stores the permission objects in a Hashtable.

Subclass implementations of PermissionCollection should assume that they may be called simultaneously from multiple threads, and therefore should be synchronized properly. Furthermore, Enumerations returned via the `elements` method are not *fail-fast*. Modifications to a collection should not be performed while enumerating over that collection.

See Also:

`Permission`, `Permissions`, Serialized Form

Constructor Summary

Constructors

Constructor and Description
`PermissionCollection()`

Method Summary

Methods

Modifier and Type	Method and Description
`abstract void`	`add(Permission permission)` Adds a permission object to the current collection of permission objects.
`abstract Enumeration<Permission>`	`elements()` Returns an enumeration of all the Permission objects in the collection.
`abstract boolean`	`implies(Permission permission)` Checks to see if the specified permission is implied by the collection of Permission objects held in this PermissionCollection.
`boolean`	`isReadOnly()` Returns true if this PermissionCollection object is marked as readonly.
`void`	`setReadOnly()` Marks this PermissionCollection object as "readonly".
`String`	`toString()` Returns a string describing this PermissionCollection object, providing information about all the permissions it contains.

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Detail

PermissionCollection

```
public PermissionCollection()
```

Method Detail

add

```
public abstract void add(Permission permission)
```

Adds a permission object to the current collection of permission objects.

Parameters:

`permission` - the Permission object to add.

Throws:

`SecurityException` - - if this PermissionCollection object has been marked readonly

`IllegalArgumentException` - - if this PermissionCollection object is a homogeneous collection and the permission is not of the correct type.

implies

```
public abstract boolean implies(Permission permission)
```

Checks to see if the specified permission is implied by the collection of Permission objects held in this PermissionCollection.

Parameters:

`permission` - the Permission object to compare.

Returns:

true if "permission" is implied by the permissions in the collection, false if not.

elements

```
public abstract Enumeration<Permission> elements()
```

Returns an enumeration of all the Permission objects in the collection.

Returns:

an enumeration of all the Permissions.

setReadOnly

```
public void setReadOnly()
```

Marks this PermissionCollection object as "readonly". After a PermissionCollection object is marked as readonly, no new Permission objects can be added to it using `add`.

isReadOnly

```
public boolean isReadOnly()
```

Returns true if this PermissionCollection object is marked as readonly. If it is readonly, no new Permission objects can be added to it using `add`.

By default, the object is *not* readonly. It can be set to readonly by a call to `setReadOnly`.

Returns:

true if this PermissionCollection object is marked as readonly, false otherwise.

toString

```
public String toString()
```

Returns a string describing this PermissionCollection object, providing information about all the permissions it contains. The format is:

```
 super.toString() (
   // enumerate all the Permission
```

```
   // objects and call toString() on them,
   // one per line..
 )
```

`super.toString` is a call to the `toString` method of this object's superclass, which is Object. The result is this PermissionCollection's type name followed by this object's hashcode, thus enabling clients to differentiate different PermissionCollections object, even if they contain the same permissions.

Overrides:

`toString` in class `Object`

Returns:

information about this PermissionCollection object, as described above.