java.security

Class AccessControlContext

java.lang.Object
     java.security.AccessControlContext

```
public final class AccessControlContext
extends Object
```

An AccessControlContext is used to make system resource access decisions based on the context it encapsulates.

More specifically, it encapsulates a context and has a single method, `checkPermission`, that is equivalent to the `checkPermission` method in the AccessController class, with one difference: The AccessControlContext `checkPermission` method makes access decisions based on the context it encapsulates, rather than that of the current execution thread.

Thus, the purpose of AccessControlContext is for those situations where a security check that should be made within a given context actually needs to be done from within a *different* context (for example, from within a worker thread).

An AccessControlContext is created by calling the `AccessController.getContext` method. The getContext method takes a "snapshot" of the current calling context, and places it in an AccessControlContext object, which it returns. A sample call is the following:

```
AccessControlContext acc = AccessController.getContext()
```

Code within a different context can subsequently call the `checkPermission` method on the previously-saved AccessControlContext object. A sample call is the following:

```
acc.checkPermission(permission)
```

See Also:

    AccessController

Constructor Summary

Constructors

Constructor and Description
AccessControlContext(**AccessControlContext** acc, **DomainCombiner** combiner) Create a new AccessControlContext with the given AccessControlContext and DomainCombiner.
AccessControlContext(**ProtectionDomain**[] context) Create an AccessControlContext with the given array of ProtectionDomains.

Method Summary

Methods

Modifier and Type	Method and Description
void	**checkPermission**(**Permission** perm)

		Determines whether the access request indicated by the specified permission should be allowed or denied, based on the security policy currently in effect, and the context in this object.
boolean	**equals**(**Object** obj) Checks two AccessControlContext objects for equality.	
DomainCombiner	**getDomainCombiner**() Get the DomainCombiner associated with this AccessControlContext.	
int	**hashCode**() Returns the hash code value for this context.	

Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, toString, wait, wait, wait

---

Constructor Detail

AccessControlContext

public AccessControlContext(ProtectionDomain[] context)

Create an AccessControlContext with the given array of ProtectionDomains. Context must not be null. Duplicate domains will be removed from the context.

Parameters:

context - the ProtectionDomains associated with this context. The non-duplicate domains are copied from the array. Subsequent changes to the array will not affect this AccessControlContext.

Throws:

NullPointerException - if context is null

AccessControlContext

public AccessControlContext(AccessControlContext acc,
                            DomainCombiner combiner)

Create a new AccessControlContext with the given AccessControlContext and DomainCombiner. This constructor associates the provided DomainCombiner with the provided AccessControlContext.

Parameters:

acc - the AccessControlContext associated with the provided DomainCombiner.

combiner - the DomainCombiner to be associated with the provided AccessControlContext.

Throws:

NullPointerException - if the provided context is null.

SecurityException - if a security manager is installed and the caller does not have the "createAccessControlContext" SecurityPermission

Since:

1.3

Method Detail

getDomainCombiner

public `DomainCombiner` getDomainCombiner()

Get the `DomainCombiner` associated with this `AccessControlContext`.

Returns:

the `DomainCombiner` associated with this `AccessControlContext`, or `null` if there is none.

Throws:

`SecurityException` - if a security manager is installed and the caller does not have the "getDomainCombiner" `SecurityPermission`

Since:

1.3

checkPermission

public void checkPermission(`Permission` perm)
                       throws `AccessControlException`

Determines whether the access request indicated by the specified permission should be allowed or denied, based on the security policy currently in effect, and the context in this object. The request is allowed only if every ProtectionDomain in the context implies the permission. Otherwise the request is denied.

This method quietly returns if the access request is permitted, or throws a suitable AccessControlException otherwise.

Parameters:

`perm` - the requested permission.

Throws:

`AccessControlException` - if the specified permission is not permitted, based on the current security policy and the context encapsulated by this object.

`NullPointerException` - if the permission to check for is null.

equals

public boolean equals(`Object` obj)

Checks two AccessControlContext objects for equality. Checks that *obj* is an AccessControlContext and has the same set of ProtectionDomains as this context.

Overrides:

`equals` in class `Object`

Parameters:

`obj` - the object we are testing for equality with this object.

Returns:

true if *obj* is an AccessControlContext, and has the same set of ProtectionDomains as this context, false otherwise.

See Also:

`Object.hashCode()`, `HashMap`

hashCode

```
public int hashCode()
```

Returns the hash code value for this context. The hash code is computed by exclusive or-ing the hash code of all the protection domains in the context together.

Overrides:

hashCode in class Object

Returns:

a hash code value for this context.

See Also:

Object.equals(java.lang.Object), System.identityHashCode(java.lang.Object)

Java™ Platform Standard Ed. 7

Overview   Package   Class   Use   Tree   Deprecated   Index   Help

Prev Class   **Next Class**   Frames   No Frames   All Classes

Summary: Nested | Field | Constr | Method   Detail: Field | Constr | Method

Submit a bug or feature
For further API reference and developer documentation, see Java SE Documentation. That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.