

JAVA.LANG.SECURITYMANAGER.CHECKACCESS METHOD

http://www.tutorialspoint.com/java/lang/securitymanager_checkaccess.htm

Copyright © tutorialspoint.com

Description

The **java.lang.SecurityManager.checkAccessThread** method throws a `SecurityException` if the calling thread is not allowed to modify the thread argument. This method is invoked for the current security manager by the `stop`, `suspend`, `resume`, `setPriority`, `setName`, and `setDaemon` methods of class `Thread`.

If the thread argument is a system thread *belongstothreadgroupwithanullparent* then this method calls `checkPermission` with the `RuntimePermission "modifyThread"` permission. If the thread argument is not a system thread, this method just returns silently. Applications that want a stricter policy should override this method. If this method is overridden, the method that overrides it should additionally check to see if the calling thread has the `RuntimePermission "modifyThread"` permission, and if so, return silently. This is to ensure that code granted that permission *suchastheJDKitself* is allowed to manipulate any thread.

If this method is overridden, then `super.checkAccess` should be called by the first statement in the overridden method, or the equivalent security check should be placed in the overridden method.

Declaration

Following is the declaration for **java.lang.SecurityManager.checkAccess** method

```
public void checkAccess(Thread t)
```

Parameters

- **t** -- the thread to be checked.

Return Value

This method does not return a value.

Exception

- **SecurityException** -- if the calling thread does not have permission to modify the thread.
- **NullPointerException** -- if the thread argument is null.

Example

Our examples require that the permissions for each command is blocked. A new policy file was set that allows only the creating and setting of our Security Manager. The file is in `C:/java.policy` and contains the following text:

```
grant {  
    permission java.lang.RuntimePermission "setSecurityManager";  
    permission java.lang.RuntimePermission "createSecurityManager";  
    permission java.lang.RuntimePermission "usePolicy";  
};
```

The following example shows the usage of `lang.SecurityManager.checkAccess` method.

```
package com.tutorialspoint;  
  
public class SecurityManagerDemo extends SecurityManager {  
  
    // check access needs to overridden  
    @Override  
    public void checkAccess(Thread t) {
```

```
throw new SecurityException("Not allowed.");
}

public static void main(String[] args) {

// set the policy file as the system security policy
System.setProperty("java.security.policy", "file:/C:/java.policy");

// create a security manager
SecurityManagerDemo sm = new SecurityManagerDemo();

// set the system security manager
System.setSecurityManager(sm);

// check if accepting access for thread is enabled
sm.checkAccess(Thread.currentThread());

// print a message if we passed the check
System.out.println("Allowed!");
}
}
```

Let us compile and run the above program, this will produce the following result:

```
Exception in thread "main" java.lang.SecurityException: Not allowed.
Loading [MathJax]/jax/output/HTML-CSS/jax.js
```