

JAVA.LANG.SECURITYMANAGER.CHECKWRITE METHOD

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

Copyright © tutorialspoint.com

Description

The **java.lang.SecurityManager.checkWriteStringfile** method throws a `SecurityException` if the calling thread is not allowed to write to the file specified by the string argument. This method calls `checkPermission` with the `FilePermissionfile`, "write" permission. If you override this method, then you should make a call to `super.checkWrite` at the point the overridden method would normally throw an exception.

Declaration

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

```
public void checkWrite(String file)
```

Parameters

- **file** -- the system-dependent file name.

Return Value

This method does not return a value.

Exception

- **SecurityException** -- if the calling thread does not have permission to access the specified file.
- **NullPointerException** -- if the file 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.checkWrite` method.

```
package com.tutorialspoint;  
  
import java.io.FileDescriptor;  
  
public class SecurityManagerDemo extends SecurityManager {  
  
    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);  
    }  
}
```

```
// perform the check
sm.checkWrite("test.txt");

// 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.security.AccessControlException: access denied
(java.io.FilePermission test.txt write)
Loading [MathJax]/jax/output/HTML-CSS/jax.js
```