Source: <http://www.codejava.net/java-se/file-io/file-change-notification-example-with-watch-service-api>

Java 7 adds a new feature for its NIO package called Watch Service API which allows applications monitoring directories and files for change events such as creation, deletion and modification. The Watch Service API is fairly simple to use, and relieves programmers from using third party libraries for files change monitoring. In this article, we’ll see how to use this API in the simplest form with a simple demo program. All interfaces and classes in the Watch Service API can be found in the package java.nio.file.

First, create a new WatchService object like this:

|  |  |
| --- | --- |
| 1 | WatchService watcher = FileSystems.getDefault().newWatchService(); |

Then register this WatchService for a given directory like the following:

|  |  |
| --- | --- |
| 1  2 | Path dir = Paths.get("Path/To/Watched/Directory");  dir.register(watcher, ENTRY\_CREATE, ENTRY\_DELETE, ENTRY\_MODIFY); |

The register() method of the Path class takes a WatchService object and a varargs list of event types which the application needs to get notified. The supported event types are:

* + **ENTRY\_CREATE**: indicates that a directory or file is created.
  + **ENTRY\_DELETE**: indicates that a directory or file is deleted.
  + **ENTRY\_MODIFY**: indicates that a directory or file is modified.
  + **OVERFLOW**: indicates that the event might have been lost or discarded. This event is always implicitly registered so we don’t need to explicitly specify it in the register() method.

All these event types are declared in the java.nio.file.StandardWatchEventKinds class.

**NOTES:** The Watch Service API does not allow registering an individual file. We’ll get a java.nio.file.NotDirectoryException if trying to do so.

Next, write an infinite loop to repeatedly poll the events fired by the watcher and process the events accordingly. For example:

while (true) {

    WatchKey key;

    try {

        // wait for a key to be available

        key = watcher.take();

    } catch (InterruptedException ex) {

        return;

    }

    for (WatchEvent<?> event : key.pollEvents()) {

        // get event type

        WatchEvent.Kind<?> kind = event.kind();

        // get file name

        @SuppressWarnings("unchecked")

        WatchEvent<Path> ev = (WatchEvent<Path>) event;

        Path fileName = ev.context();

        System.out.println(kind.name() + ": " + fileName);

        if (kind == OVERFLOW) {

            continue;

        } else if (kind == ENTRY\_CREATE) {

            // process create event

        } else if (kind == ENTRY\_DELETE) {

            // process delete event

        } else if (kind == ENTRY\_MODIFY) {

            // process modify event

        }

    }

    // IMPORTANT: The key must be reset after processed

    boolean valid = key.reset();

    if (!valid) {

        break;

    }

}

**NOTES:**It’s very important that the key must be reset after the events have been processed. If not, the key won’t receive further events. If the reset() method returns false, the directory is inaccessible (might be deleted), so the loop exits.

The following is complete code of a demo program that watches a directory for all changes, and pays attention to the modification of a file named DirectoryWatchDemo.java:

package net.codejava.io;

import static java.nio.file.StandardWatchEventKinds.ENTRY\_CREATE;

import static java.nio.file.StandardWatchEventKinds.ENTRY\_DELETE;

import static java.nio.file.StandardWatchEventKinds.ENTRY\_MODIFY;

import java.io.IOException;

import java.nio.file.FileSystems;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.WatchEvent;

import java.nio.file.WatchKey;

import java.nio.file.WatchService;

/\*\*

 \* This program demonstrates how to use the Watch Service API to monitor change

 \* events for a specific directory.

 \* @author www.codejava.net

 \*

 \*/

public class DirectoryWatchDemo {

    public static void main(String[] args) {

        try {

            WatchService watcher = FileSystems.getDefault().newWatchService();

            Path dir = Paths.get("E:/Test/Download");

            dir.register(watcher, ENTRY\_CREATE, ENTRY\_DELETE, ENTRY\_MODIFY);

            System.out.println("Watch Service registered for dir: " + dir.getFileName());

            while (true) {

                WatchKey key;

                try {

                    key = watcher.take();

                } catch (InterruptedException ex) {

                    return;

                }

                for (WatchEvent<?> event : key.pollEvents()) {

                    WatchEvent.Kind<?> kind = event.kind();

                    @SuppressWarnings("unchecked")

                    WatchEvent<Path> ev = (WatchEvent<Path>) event;

                    Path fileName = ev.context();

                    System.out.println(kind.name() + ": " + fileName);

                    if (kind == ENTRY\_MODIFY &&

                            fileName.toString().equals("DirectoryWatchDemo.java")) {

                        System.out.println("My source file has changed!!!");

                    }

                }

                boolean valid = key.reset();

                if (!valid) {

                    break;

                }

            }

        } catch (IOException ex) {

            System.err.println(ex);

        }

    }

}

This program would produce the following output when running:

Watch Service registered for dir: Download

ENTRY\_CREATE: joke\_fails.gif

ENTRY\_MODIFY: joke\_fails.gif

ENTRY\_MODIFY: joke\_fails.gif

ENTRY\_MODIFY: DirectoryWatchDemo.java

My source file has changed!!!

ENTRY\_MODIFY: DirectoryWatchDemo.java

My source file has changed!!!

Delete the monitored directory or kill the process to stop the program.