Remote Debugging Steps for all

Thursday, August 25, 2022   7:04 PM

Java Remote Debugging

1. Create a new mount point directory on your machine for the remote directory:

Code:
```
$ sudo mkdir /mnt/debug
```
2. Remote mount the directory of the remote application on your machine (to avoid having to copy all of the remote files to your machine):

Code:
```
$ sudo sshfs -o allow_other,default_permissions user@machine:/path/to/program
/mnt/debug
```
3. Decompile the application Jar (or the most interesting one out of several) with JD-GUI.

4. Save the decompiled JD-GUI source file and unzip it. The decompiled Java code won't be in the correct format for Visual Studio and breakpoints won't be hit if you try and debug right now. You'll have to format the output of the files and folders in the unzipped file that you get from JD-GUI, but once you have this format, debugging should work with visual studio code.

You can also use the following command to extract the jar files. This doesn't decompile them, but you'll be able to see the general file structure and get a sense of the size of the application:

Code:
```
jar xvf file.jar
```
5. The file and directory layout that has worked for me has been as follows:

Pasted image 20210326212537.png

- *src*: Base directory where everything is stored.
- *application.properties*: Contain the properties to run the application in a different environment.
- *com*: Directory that contains the actual application java files
- *lib*: Directory that contains all the third party jar files used in the application.
- *META-INF*: Holds the `MANIFEST.MF` file that is used to define extension and package related data.
- *org*: Holds the Spring Framework model for Java enterprise applications.
- *static*: Holds the application styling files such as css and webfont files.
- *templates*: Holds the application template pages.

6. Open the mounted folder with Visual Studio code:

Code:
```
$ cd /mnt/debug/src
$ code .
```
7. You'll have to rename some of the *package* lines at the top of the *.java* files in the *com* directory to match the new directory layout.

8. Create a Java remote debugging *launch.json* file:

Example:

Code:
```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "java",
            "name": "Debug (Attach) - Remote",
            "request": "attach",
            "hostName": "remoteIP",
            "port": 8000
        }
    ]
}
```
9. Start the debugger and set breakpoints to start debugging.

10. When finished debugging, unmount the directory mount created in step 1:

Code:
```
$ sudo umount /mnt/debug
```
NodeJS Remote Debugging

1. Create a new mount point directory on your machine for the remote directory:

Code:
```
$ sudo mkdir /mnt/debug
```
2. Remote mount the directory of the remote application on your machine (to avoid having to copy all of the remote files to your machine):

Code:
```
$ sudo sshfs -o allow_other,default_permissions user@machine:/path/to/program
/mnt/debug
```
3. Open the mounted folder with Visual Studio code:

Code:
```
$ cd /mnt/debug
$ code .
```
4. Use SSH local port forwarding to get the remote node port on your machine:

Code:
```
$ ssh -fNL 9229:127.0.0.1:9229 user@machine
```
5. Create a NodeJS debugging launch.json file and add remote debugging configuration:
Example:

Code:
```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "address": "127.0.0.1",
            "localRoot": "${workspaceFolder}",
            "name": "Attach to Remote",
            "port": 9229,
            "remoteRoot": "/path/to/remote/application/folder",
            "request": "attach",
            "skipFiles": [
                "<node_internals>/**"
            ],
            "type": "pwa-node"
        }
    ]
}
```
6. Start the debugger and set breakpoints to start debugging. **Note** Breakpoints might appear as a gray circle for a few seconds after you initially place them if the remote debugger is still in the process of attaching, but they'll fill in red once that's complete.

7. When finished debugging, unmount the directory mount created in step 1:

Code:
```
$ sudo umount /mnt/debug
```
PHP Remote Debugging

1. Make sure that XDebug is installed on the remote server (https://xdebug.org/docs/install)
Installing it can be a bit finicky, so try and install it with PECL if you are on Linux.

Code:
```
$ pecl install xdebug
```
On Windows, downloading and compiling the Source is the best way.
Make sure the version of XDebug you are downloading is compatible with the version of PHP you are running.

2. On the remote server, add the following code to the bottom of your php.ini file:

Code:
```
[XDebug]
;; Only Zend OR (!) XDebug
zend_extension=/path/to/xdebug.so
xdebug.mode=debug
xdebug.start_with_request=yes
xdebug.discover_client_host=true
xdebug.client_host=localhost
xdebug.client_port=9001
xdebug.log=/tmp/xdebug.log
xdebug.log_level=7
xdebug.connect_timeout_ms=1000
```
You can use the following commands to see if XDebug is loaded and where the php.ini files are located:

Code:

```
php -v
php --ini
```
3. Restart PHP and Apache (or whatever web server you are running) on the remote server.

4. Create a new mount point directory on your machine for the remote directory:

Code:
```
$ sudo mkdir /mnt/debug
```
5. Remote mount the directory of the remote application (or directory that contains the files and sub-folders of for the application) on your machine (to avoid having to copy all of the remote files to your machine):

Code:
```
$ sudo sshfs -o allow_other,default_permissions user@machine:/path/to/program
/mnt/debug
```
6. Use SSH remote port forwarding to get access to the XDebug port:

Code:
```
$ ssh -fNR 9001:localhost:9001 user@machine
```
7. Open the mounted folder with Visual Studio code:

Code:
```
$ cd /mnt/debug
$ code .
```
8. Install the PHP Debug extension for Visual Studio code and create a "Listen for Xdebug" launch.json file configuration:
Example:

Code:
```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Listen for Xdebug",
            "type": "php",
            "request": "launch",
            "port": 9001,
            "pathMappings": {
                "/path/to/remote/application/folder": "${workspaceFolder}",
            },
        }
    ]
}
```
9. Start the debugger and set breakpoints to start debugging. **Note** After walking through a breakpoint and stepping out of it, you may have to stop and restart the debugger to set a new one.

10. When finished debugging, unmount the directory mount created in step 5:

Code:
```
$ sudo umount /mnt/debug
```
Edit:
If you run into an issue where you can't write to files in the the remote mounted directory, try removing the 'default_permissions' flag when running SSHFS:

Code:
```
$ sshfs -o allow_other user@machine:/path/to/program /mnt/debug
```
Last edited by SecVoid; 2021-04-02 at 21:12.