Java - Try with Resources

Try with Resources

The Java try-with-resources statement is a try statement that is used for declaring one or more resources such as streams, sockets, databases, connections, etc. These resources must be closed while the program is being finished. The try-with-resources statement closes the resources at the end of the statement.

The try-with-resources feature was introduced in Java7. The try-with-resources can also be a replacement for try-catch-finally with resources objects.

The try-with-resources statement is also referred as "Automatic Resource Management” and it was introduced in Java7. This statement is a replacement of try-catch-finally-statement if you are working resources objects.

Syntax of Try with Resources

To use this statement, you simply need to declare the required resources within the parenthesis, and the created resource will be closed automatically at the end of the block. Following is the syntax of try-with-resources statement:

```
try(resources declarations) {
    // use of the resources
}
catch(Exception e) {
    // exception handling
}
```

For example, opening a file with try-with-resources:

```
try(FileReader fr = new FileReader("file path")) {
   // use the resource
   } catch () {
      // body of catch
   }
}
```

Following is the program that reads the data in a file using try-with-resources statement.

Learn **Java** in-depth with real-world projects through our **Java certification course**. Enroll and become a certified expert to boost your career.

Example: Try with Resources in Java

In this program, we're creating the FileReader object within try with resources statement. FileReader fr, reference is declared within the try statement and we need not to remember to close it in finally block as it will be closed automatically by JVM so that there is no memory leak or loose connection possibility.

```
import java.io.FileReader;
import java.io.IOException;

public class Try_withDemo {

   public static void main(String args[]) {
      try(FileReader fr = new FileReader("E://file.txt")) {
         char [] a = new char[50];
         fr.read(a);   // reads the contentto the array
         for(char c : a)
         System.out.print(c);   // prints the characters one by one
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Try with Resources having Multiple Resources

You can also declare multiple resource inside a try block. Consider the below-given example,

```
// This example is to use Try with Resources
// with multiple  Resources
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class Main {
  public static void main(String[] args) {
    // try block with multiple resources
    try (
      FileReader fileReader = new FileReader("file1.txt");
      BufferedReader bufferedReader = new BufferedReader(fileReader);
      FileWriter fileWriter = new FileWriter("file2.txt");
      PrintWriter printWriter = new PrintWriter(fileWriter)
    ) {
      String line;
      while ((line = bufferedReader.readLine()) != null) {
        // Read content line by line and write it
        // to the output (file2.txt) file
        printWriter.println(line);
      }
      System.out.println("Content copied.");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Example: Without Try with Resources

In the following program, we are reading data from a file using **FileReader** and we are closing it using finally block. In this program, we're creating the FileReader object within try block. FileReader fr, reference is declared outside the try block so that it is accessible outside the try block and we need to remember to close it in finally block or before program exits so that there is no memory leak or loose connection possibility.

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class ReadData_Demo {

   public static void main(String args[]) {
      FileReader fr = null;
      try {
         File file = new File("file.txt");
         fr = new FileReader(file); char [] a = new char[50];
         fr.read(a);   // reads the content to the array
         for(char c : a)
         System.out.print(c);   // prints the characters one by one
      } catch (IOException e) {
         e.printStackTrace();
      }finally {
         try {
            fr.close();
         } catch (IOException ex) {
            ex.printStackTrace();
         }
      }
   }
}
```

Try with Resources: Points to Remember

Following points are to be kept in mind while working with try-with-resources statement.

- To use a class with try-with-resources statement it should implement **AutoCloseable** interface and the **close()** method of it gets invoked automatically at runtime.
- You can declare more than one class in try-with-resources statement.
- While you declare multiple classes in the try block of try-with-resources statement these classes are closed in reverse order.
- Except the declaration of resources within the parenthesis everything is the same as normal try/catch block of a try block.
- The resource declared in try gets instantiated just before the start of the try-block.
- The resource declared at the try block is implicitly declared as final.

try with resources improvement with Java 9

Prior to Java 9, resources are to be declared before try or inside try statement as shown below in given example. In this example, we'll use BufferedReader as resource to read a string and then BufferedReader is to be closed.

Before Java 9

</> Open Compiler

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StringReader;

public class Tester {
   public static void main(String[] args) throws IOException {
      System.out.println(readData("test"));
   }
   static String readData(String message) throws IOException {
      Reader inputString = new StringReader(message);
      BufferedReader br = new BufferedReader(inputString);
      try (BufferedReader br1 = br) {
         return br1.readLine();
      }
   }
}
```

Output

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

```
test
```

Here we need to declare a resource br1 within try statment and then use it. In Java9, we don't need to declare br1 anymore and following program will give the same result.

Java 9 onwards

</> Open Compiler

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StringReader;

public class Tester {
   public static void main(String[] args) throws IOException {
      System.out.println(readData("test"));
   }
   static String readData(String message) throws IOException {
      Reader inputString = new StringReader(message);
      BufferedReader br = new BufferedReader(inputString);
      try (br) {
         return br.readLine();
      }
   }
}
```

Output

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

```
test
```