读取文件与下载文件关系如何呢？我理解在后端都是以读取文件方式获得目标文件内容。最后将文件流内容传给浏览器，并在header头中添加浏览器解析方式和文件名，比如：文件下载到本地实现方法可以使用响应头 `Content-disposition` 来控制。

`Content-Disposition` 响应头：指示回复的内容该以何种形式展示，是以 `内联` 的形式（即网页或者页面的一部分），还是以 `附件` 的形式下载并保存到本地。

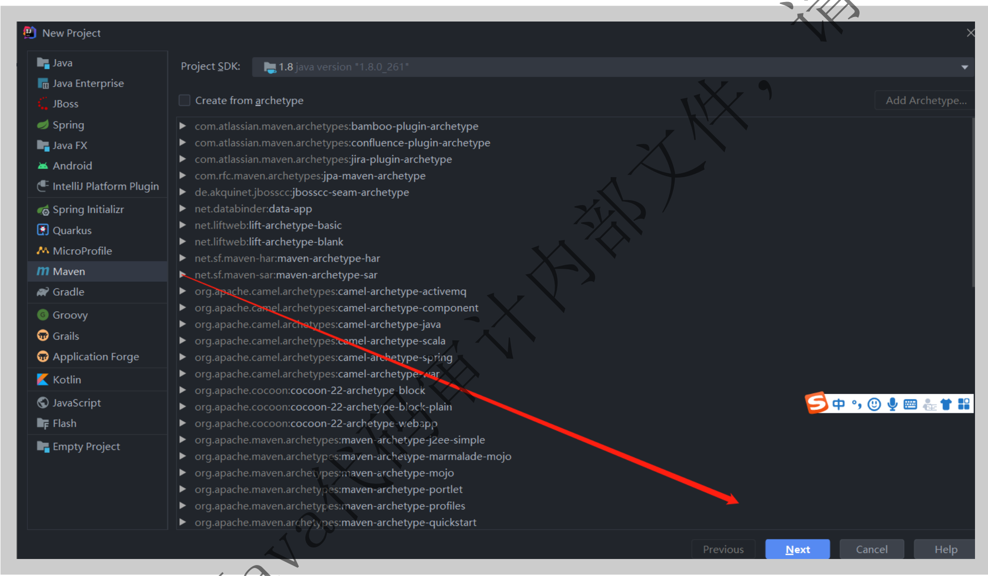
但从漏洞挖掘角度来看，我们使用下载或读取功能的目的是获得目标敏感文件中的数据。

一、新建Java基础工程

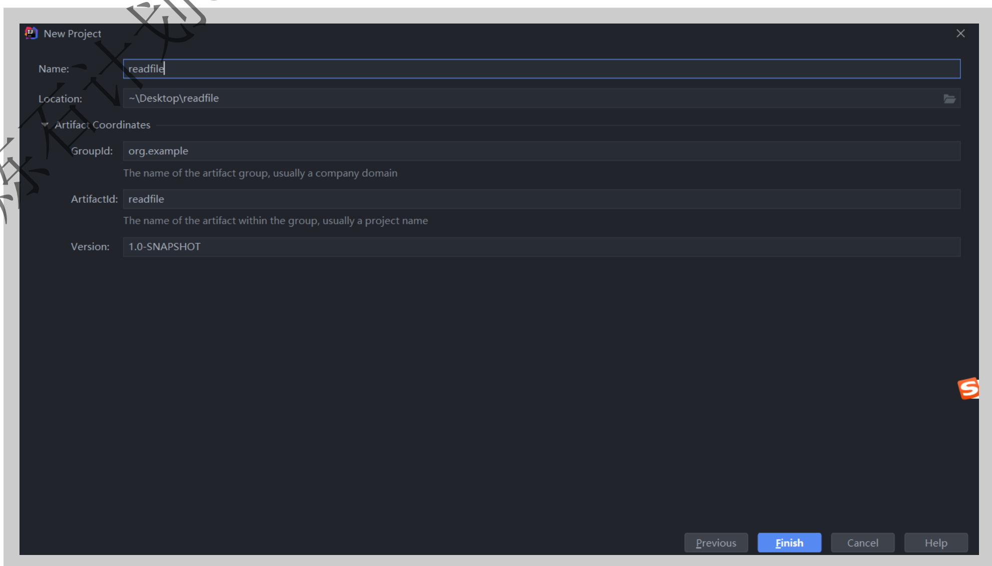
使用IDEA创建一个基础项目工程，用于下面几种文件读取/下载实现方法的调试。

①、双击IDEA启动，点击 `Create New Project` 。

②、左侧选择Maven，模板就不用选择了，点击Next。如下图所示：



③、随便起个名字，最后点击Finish即可。如下图所示：



④、在桌面随便创建一个txt文本文件，里面键入任意内容，用于下面读取文件。

到此项目创建完成，下面根据每一个实现方法编写对应实现代码。

二、文件读取/下载实现方法

下面介绍几种我学到的Java读取/下载文件的几种实现方法。

方法一：使用java.nio.file.Files读取文本

1、简述

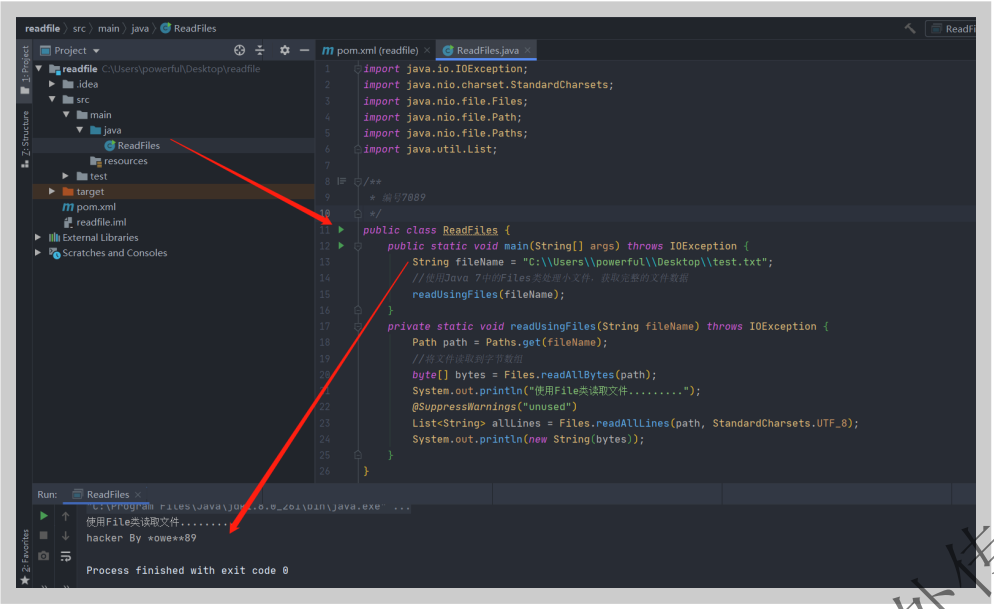
使用 `Files` 类将文件的所有内容读入字节数组。 `Files` 类还有一个方法可以读取所有行到字符串列表。 `Files` 类是在Java 7中引入的，如果想加载所有文件内容，使用这个类是比较适合的。只有在处理小文件并且需要加载所有文件内容到内存中时才应使用此方法。

2、实现代码

①、在刚才创建的工程下的 `src/main/java` 目录下创建一个名为 `ReadFiles` 的Java Class。并键入以下代码。

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

public class ReadFiles {
    public static void main(String[] args) throws IOException {
        String fileName = "C:\\Users\\powerful\\Desktop\\test.txt";
        //使用Java 7中的Files类处理小文件，获取完整的文件数据
        readUsingFiles(fileName);
    }
    private static void readUsingFiles(String fileName) throws
IOException {
        Path path = Paths.get(fileName);
        //将文件读取到字节数组
        byte[] bytes = Files.readAllBytes(path);
        System.out.println("使用File类读取文件.........");
        @SuppressWarnings("unused")
        List<String> allLines = Files.readAllLines(path,
StandardCharsets.UTF_8);
        System.out.println(new String(bytes));
    }
}
```

②、启动运行项目，观察结果，如下图所示：



方法二：使用java.io.FileReader类读取文本

1、简述

可以使用 `FileReader` 获取 `BufferedReader` ，然后逐行读取文件。 `FileReader` 不支持编码并使用系统默认编码，因此它不是一种Java中读取文本文件的非常有效的方法。

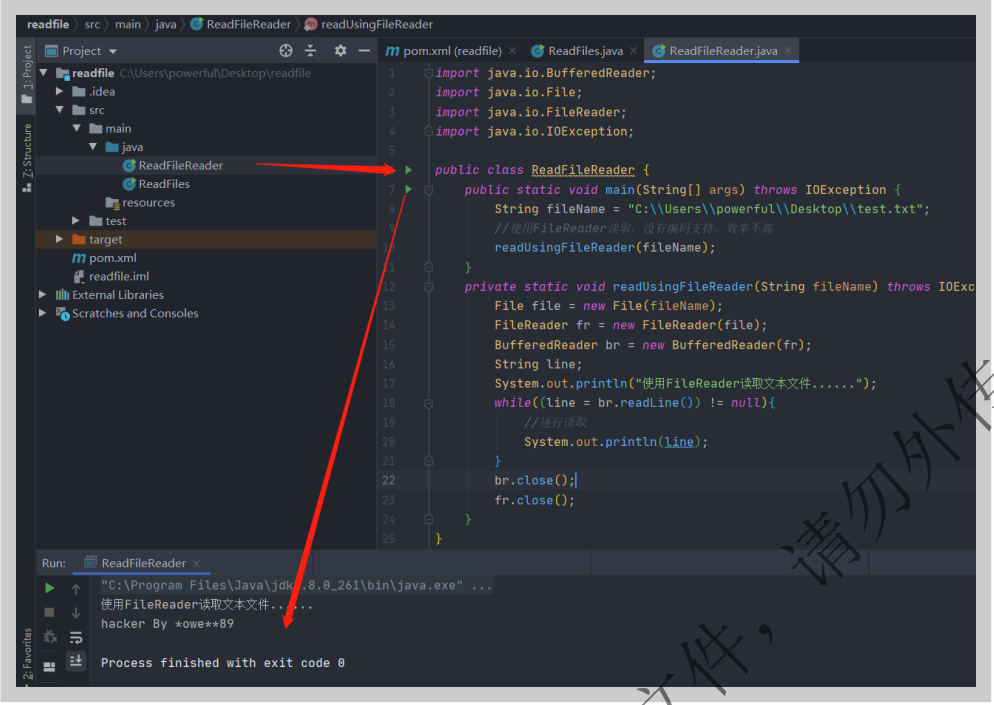
2、实现代码

①、在刚才创建的工程下的 `src/main/java` 目录下创建一个名为 `ReadFileReader` 的Java Class。并键入以下代码。

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

public class ReadFileReader {
    public static void main(String[] args) throws IOException {
        String fileName = "C:\\Users\\powerful\\Desktop\\test.txt";
        //使用FileReader读取，没有编码支持，效率不高
        readUsingFileReader(fileName);
    }
    private static void readUsingFileReader(String fileName) throws
IOException {
        File file = new File(fileName);
        FileReader fr = new FileReader(file);
        BufferedReader br = new BufferedReader(fr);
        String line;
        System.out.println("使用FileReader读取文本文件......");
        while((line = br.readLine()) != null){
            //逐行读取
            System.out.println(line);
        }
        br.close();
        fr.close();
```

```
    }
}
```

②、启动运行项目，观察结果，如下图所示：



方法三：使用java.io.BufferedReader读取文本

1、简述

如果想逐行读取文件并对它们进行处理，那么 `BufferedReader` 是非常合适的。它适用于处理大文件，也支持编码。

`BufferedReader` 是同步的，因此可以安全地从多个线程完成对 `BufferedReader` 的读取操作。`BufferedReader` 的默认缓冲区大小为： `8KB` 。

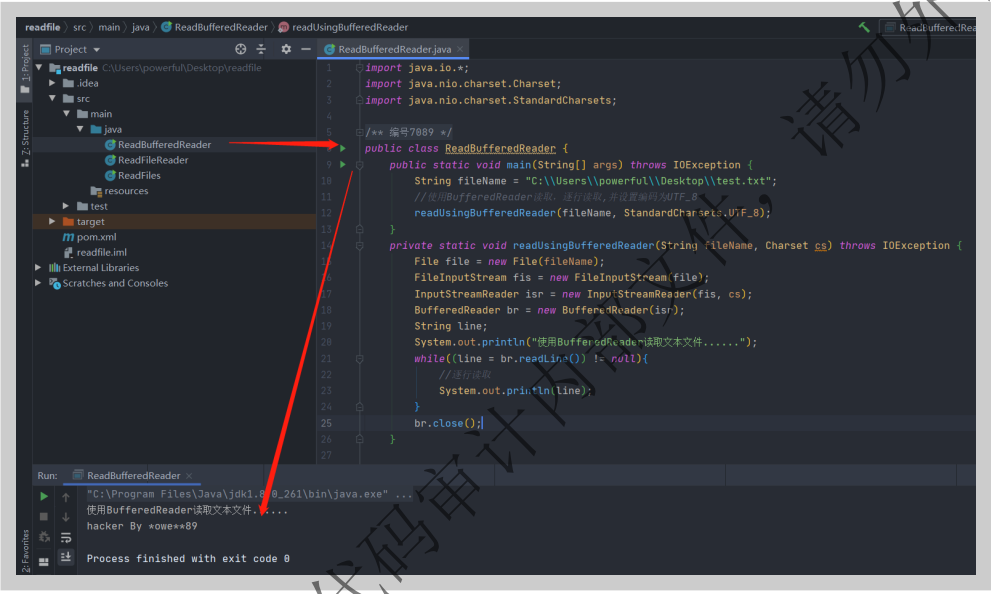
2、实现代码

①、在刚才创建的工程下的 `src/main/java` 目录下创建一个名为 `ReadBufferedReader` 的Java Class。并键入以下代码。

```
import java.io.*;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;

public class ReadBufferedReader {
    public static void main(String[] args) throws IOException {
        String fileName = "C:\\Users\\powerful\\Desktop\\test.txt";
        //使用BufferedReader读取，逐行读取,并设置编码为UTF_8
        readUsingBufferedReader(fileName, StandardCharsets.UTF_8);
    }
    private static void readUsingBufferedReader(String fileName,
Charset cs) throws IOException {
        File file = new File(fileName);
```

```
        FileInputStream fis = new FileInputStream(file);
        InputStreamReader isr = new InputStreamReader(fis, cs);
        BufferedReader br = new BufferedReader(isr);
        String line;
        System.out.println("使用BufferedReader读取文本文件.....");
        while((line = br.readLine()) != null){
            //逐行读取
            System.out.println(line);
        }
        br.close();
    }

}
```

②、启动运行项目，观察结果，如下图所示：



方法四：使用Scanner读取文本

1、简述

如果要逐行读取文件或基于某些java正则表达式读取文件，则可使用 `Scanner` 类。

`Scanner` 类使用分隔符模式将其输入分解为标记，分隔符模式默认匹配空格。然后可以使用各种下一种方法将得到的标记转换成不同类型的值。 `Scanner` 类不同步，因此不是线程安全的。

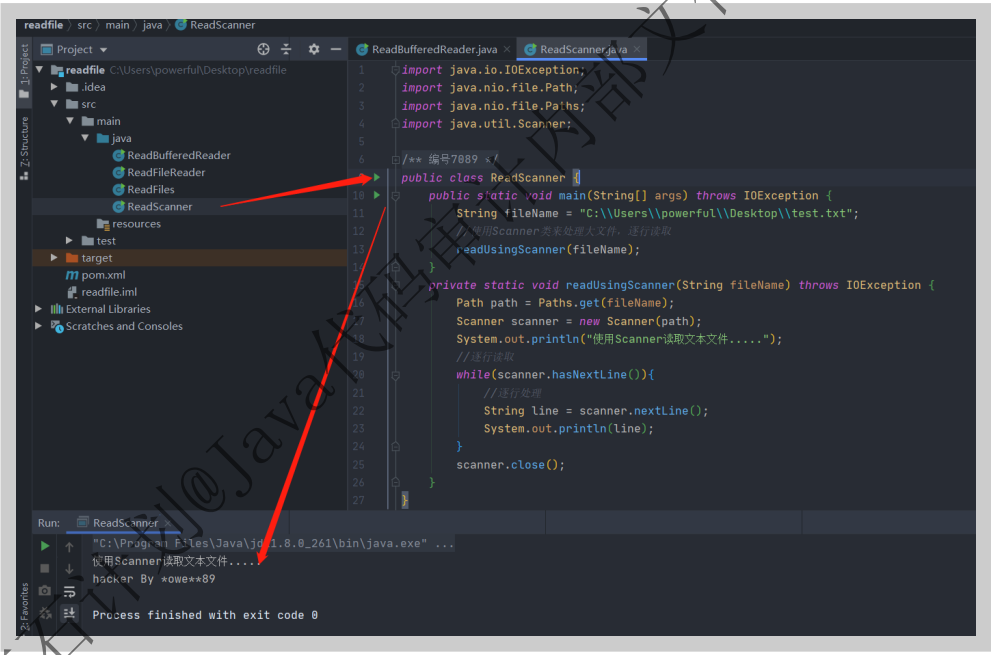
2、实现代码

①、在刚才创建的工程下的 `src/main/java` 目录下创建一个名为 `ReadScanner` 的 Java Class。并键入以下代码。

```
import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Scanner;
```

```java
public class ReadScanner {
    public static void main(String[] args) throws IOException {
        String fileName = "C:\\Users\\powerful\\Desktop\\test.txt";
        //使用Scanner类来处理大文件，逐行读取
        readUsingScanner(fileName);
    }
    private static void readUsingScanner(String fileName) throws
IOException {
        Path path = Paths.get(fileName);
        Scanner scanner = new Scanner(path);
        System.out.println("使用Scanner读取文本文件.....");
        //逐行读取
        while(scanner.hasNextLine()){
            //逐行处理
            String line = scanner.nextLine();
            System.out.println(line);
        }
        scanner.close();
    }
}
```


②、启动运行项目，观察结果，如下图所示：



方式五：使用RandomAccessFile断点续传读取文本

1、简述

随机流（RandomAccessFile）不属于IO流，支持对文件的读取和写入随机访问。

首先把随机访问的文件对象看作存储在文件系统中的一个大型 byte 数组，然后通过指向该 byte 数组的光标或索引（即：文件指针 FilePointer）在该数组任意位置读取或写入任意数据。

断点续传是在下载或上传时，将下载或上传任务（一个文件或一个压缩包）人为的划分为几个部分，每一个部分采用一个线程进行上传或下载，如果碰到网络故障，可以从已经上传或下载的部分开始继续上传或者下载未完成的部分，而没有必要从头开始上传或者下载。

断点续传实现原理：

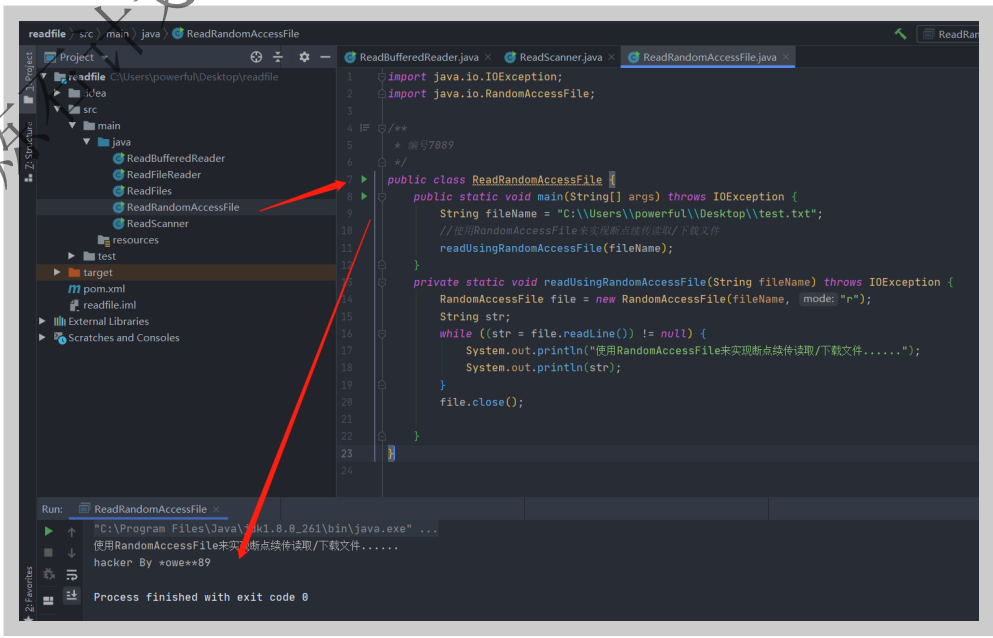
1. 下载断开的时候，记录文件断点的位置position；
2. 继续下载的时候，通过RandomAccessFile找到之前的position位置开始下载

2、实现代码

①、在刚才创建的工程下的 `src/main/java` 目录下创建一个名为 `ReadRandomAccessFile` 的Java Class。并键入以下代码。

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class ReadRandomAccessFile {
    public static void main(String[] args) throws IOException {
        String fileName = "C:\\Users\\powerful\\Desktop\\test.txt";
        //使用RandomAccessFile来实现断点续传读取/下载文件
        readUsingRandomAccessFile(fileName);
    }
    private static void readUsingRandomAccessFile(String fileName)
throws IOException {
        RandomAccessFile file = new RandomAccessFile(fileName, "r");
        String str;
        while ((str = file.readLine()) != null) {
            System.out.println("使用RandomAccessFile来实现断点续传读取/下载
文件......");
            System.out.println(str);
        }
        file.close();
    }
}
```

②、启动运行项目，观察结果，如下图所示：



方式六：使用外部库org.apache.commons.io.FileUtils.readFileToString()读取文本

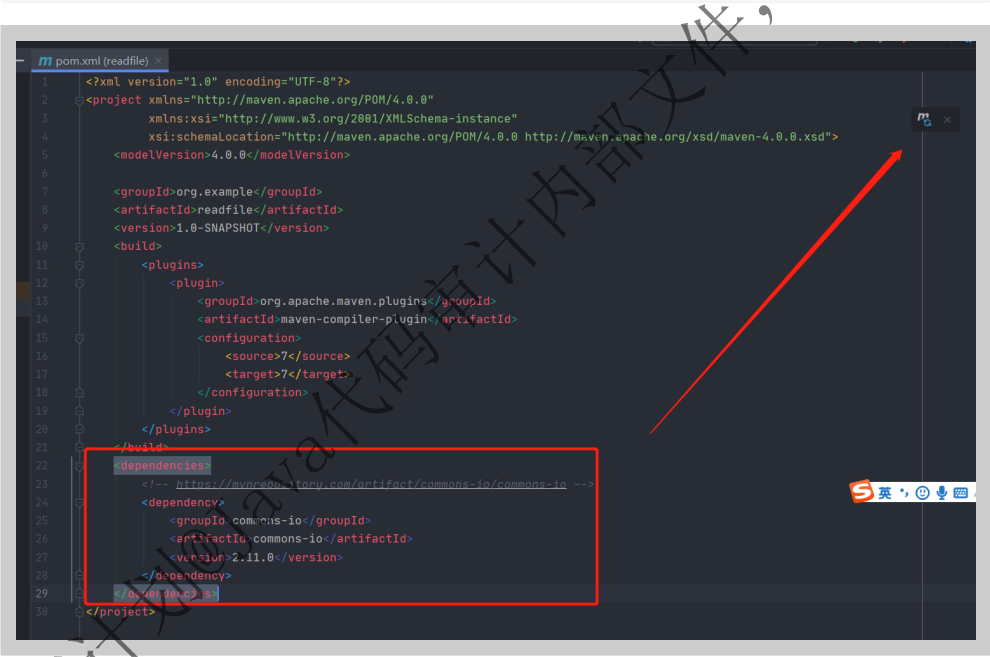
1、简述

使用Commons-io库一行代码文件来实现读取文件。

2、实现代码

①、首先要在 `pom.xml` 中引入 `Commons-io` 依赖后重载maven。如下图所示：

```
        <!-- https://mvnrepository.com/artifact/commons-io/commons-io
->
        <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>2.11.0</version>
        </dependency>
```

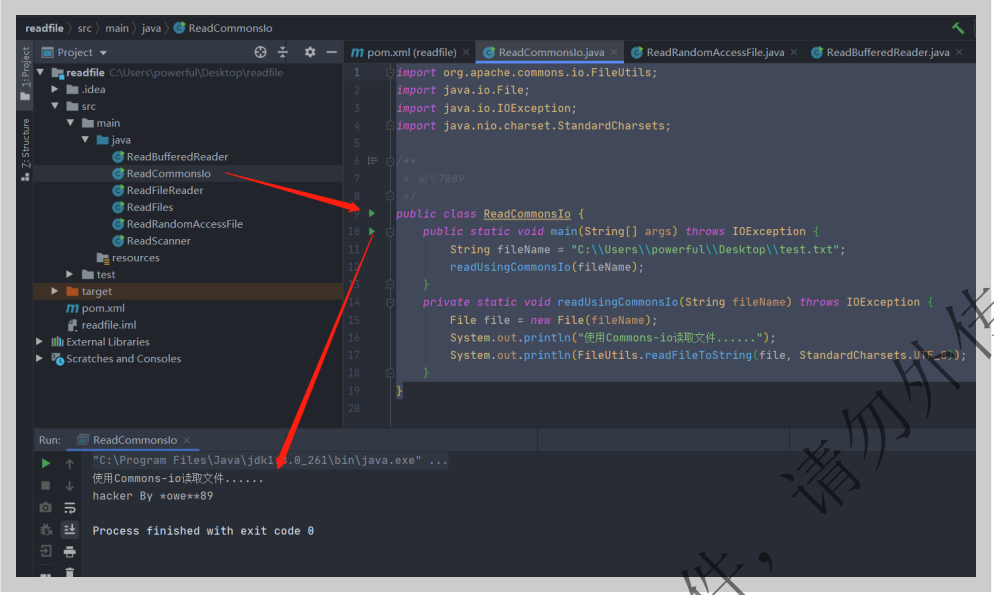


②、在刚才创建的工程下的 `src/main/java` 目录下创建一个名为 `ReadCommonsIo` 的Java Class。并键入以下代码。

```
import org.apache.commons.io.FileUtils;
import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;

public class ReadCommonsIo {
    public static void main(String[] args) throws IOException {
        String fileName = "C:\\Users\\powerful\\Desktop\\test.txt";
        readUsingCommonsIo(fileName);
    }
    private static void readUsingCommonsIo(String fileName) throws
IOException {
        File file = new File(fileName);
        System.out.println("使用Commons-io读取文件.....");
```

```
        System.out.println(FileUtils.readFileToString(file,
StandardCharsets.UTF_8));
    }
}
```

③、启动运行项目，观察结果，如下图所示：



方式七：使用Files.readString读取文本

①、简述

Java 11添加了readString()方法来读取小文件String。官方介绍：`https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/nio/file/Files.html#readString(java.nio.file.Path,java.nio.charset.Charset)`

②、实现代码

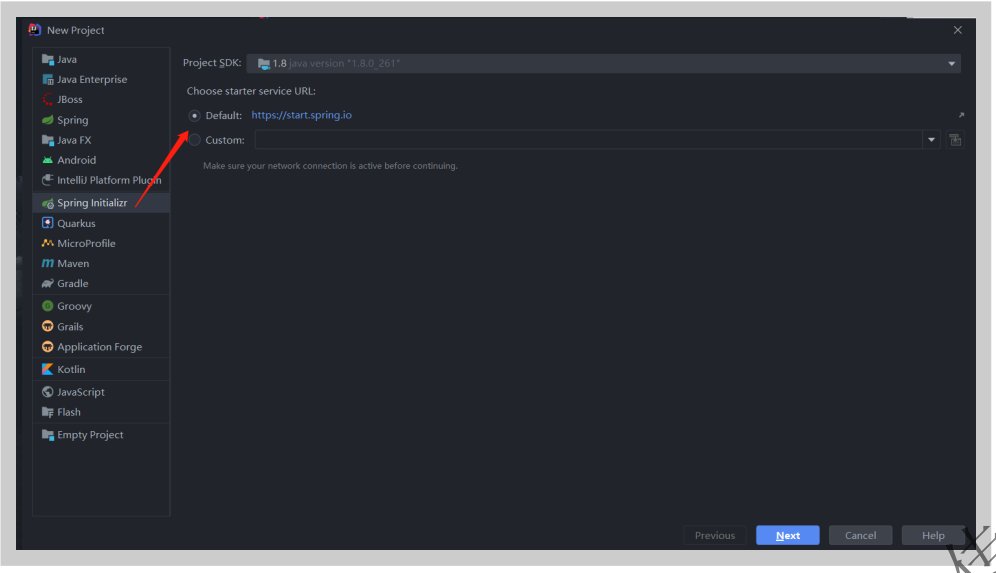
```
String content = Files.readString(path, StandardCharsets.US_ASCII);
```

三、文件读取/下载JavaWeb工程

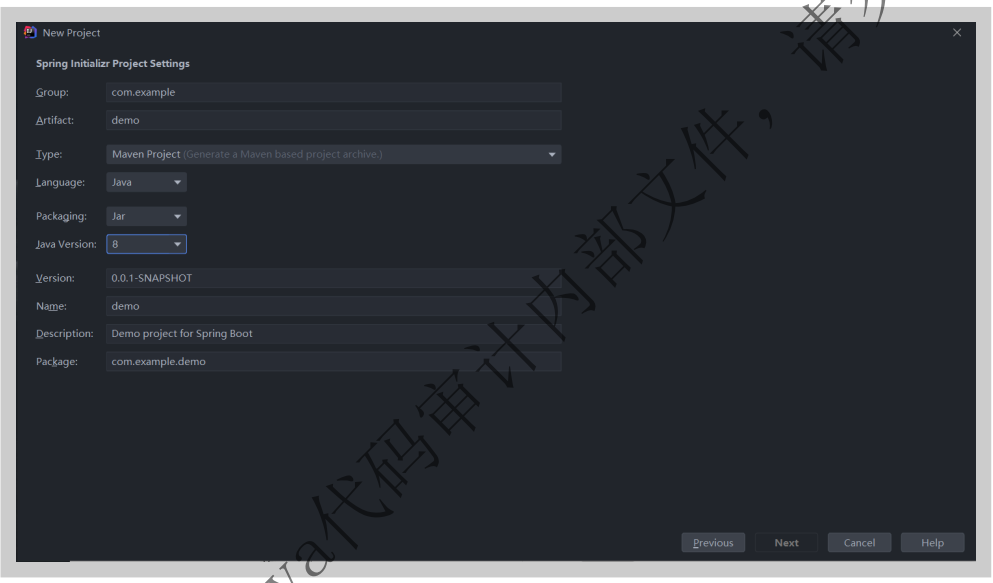
我们创建一个基于SpringBoot的读取文件的JavaWeb工程，这回我们从WEB角度调试上面几种方法。

①、双击IDEA启动，点击 `Create New Project` 。

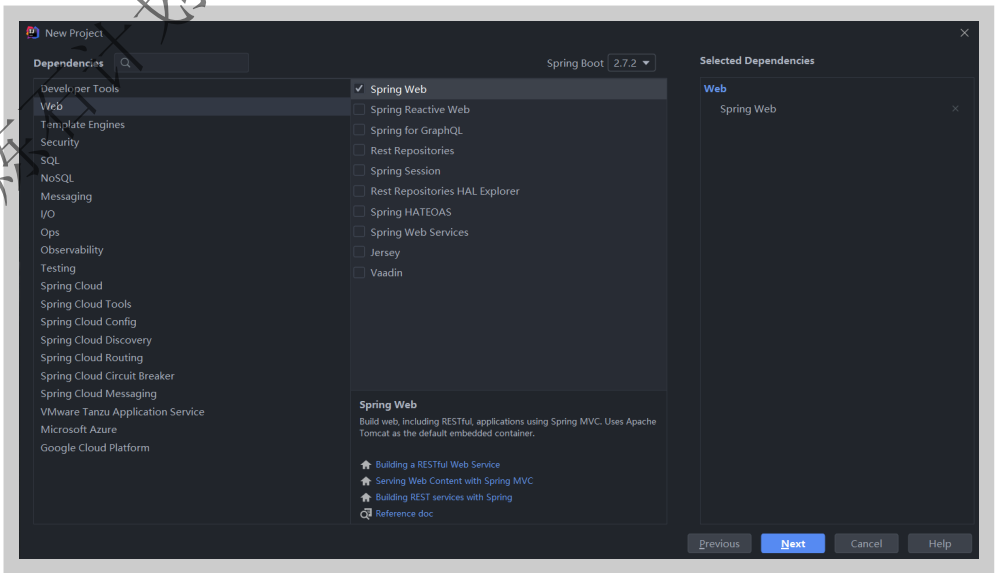
②、左侧选择 `Spring Initializr` ，内容默认即可，点击Next，如下图所示：



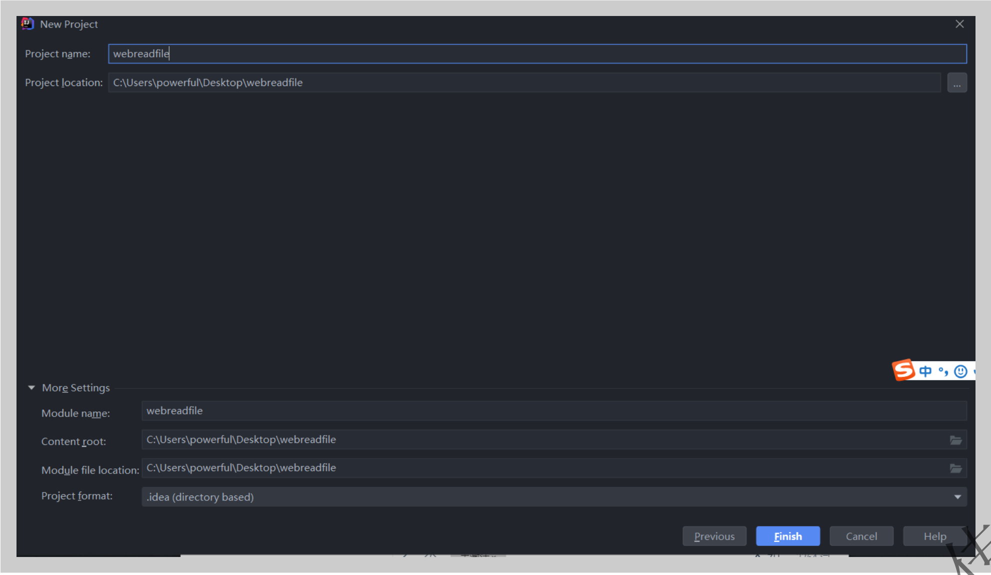
③、稍等片刻， Srping Initializr Project Settings页面配置内容将Java Version选择为8，其他默认即可。如下图所示：



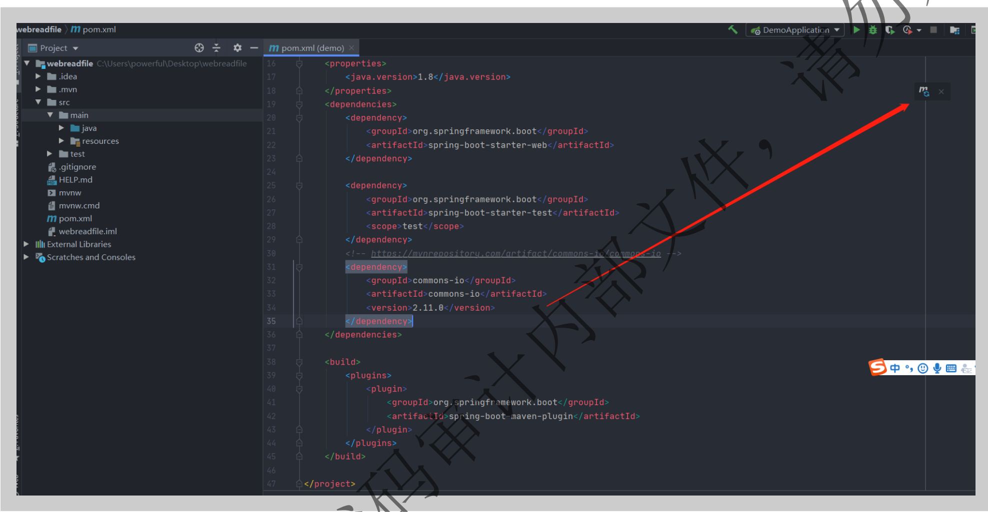
④、点击Next，进入依赖项选择页面，我们选择 `Web -> Spring Web` 这一个即可，如下图所示：



⑤、起个项目名称，最后点击Finish，如下图所示：



⑥、在 `pom.xml` 中引入Commons-Io依赖后重载Maven，便于后面使用。如下图所示：



⑦、在 `src/main/java/com/example/demo` 目录下创建一个名为 `ReadFilesController` 的Java Class，并键入以下代码。

```
package com.example.demo;

import org.apache.commons.io.FileUtils;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.Scanner;

/**
 * 编号7089
 */
@Controller
@ResponseBody
public class ReadFilesController {
```

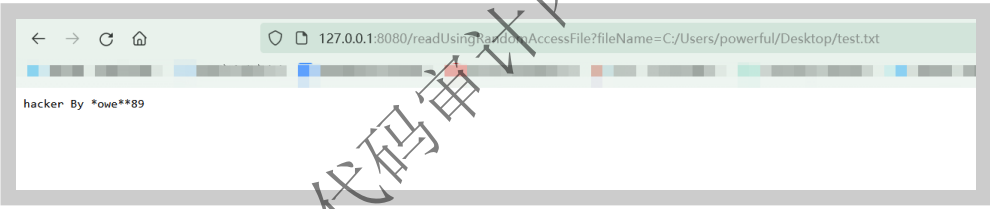
```
    @RequestMapping("/readUsingFiles")
    public String readUsingFiles(String fileName, HttpServletResponse
response) throws IOException {
        //使用Java 7中的Files类处理小文件，获取完整的文件数据
        Path path = Paths.get(fileName);
        //将文件读取到字节数组
        byte[] bytes = Files.readAllBytes(path);
        System.out.println("使用File类读取文件.........");
        @SuppressWarnings("unused")
        List<String> allLines = Files.readAllLines(path,
StandardCharsets.UTF_8);
        //将注释去掉，重新运行启动项目，在浏览器键入要读取的文件地址，观察下效果有什
么不一样。
        //response.reset();
        //response.setContentType("application/octet-stream");
        //response.addHeader("Content-Disposition", "attachment;
filename=" + URLEncoder.encode(fileName, "UTF-8"));

        System.out.println(new String(bytes));
        return new String(bytes);

    }

    @RequestMapping("/readUsingFileReader")
    public void readUsingFileReader(String fileName,
HttpServletResponse response) throws IOException {
        //使用FileReader读取，没有编码支持，效率不高
        File file = new File(fileName);
        FileReader fr = new FileReader(file);
        BufferedReader br = new BufferedReader(fr);
        String line;
        System.out.println("使用FileReader读取文本文件......");
        //将注释去掉，重新运行启动项目，在浏览器键入要读取的文件地址，观察下效果有什
么不一样。
        //response.reset();
        //response.setContentType("application/octet-stream");
        //response.addHeader("Content-Disposition", "attachment;
filename=" + URLEncoder.encode(fileName, "UTF-8"));
        PrintWriter out = response.getWriter();
        while ((line = br.readLine()) != null) {
            //逐行读取
            System.out.println(line);
            out.print(line);
        }
        br.close();
        fr.close();
    }

    @RequestMapping("/ReadBufferedReader")
    public void readBufferedReader(String fileName, HttpServletResponse
response) throws IOException{
        File file = new File(fileName);
        FileInputStream fis = new FileInputStream(file);
        InputStreamReader isr = new InputStreamReader(fis,
StandardCharsets.UTF_8);
        BufferedReader br = new BufferedReader(isr);
```

```
        String line;
        //将注释去掉，重新运行启动项目，在浏览器键入要读取的文件地址，观察下效果有什么不一样。
        //response.reset();
        //response.setContentType("application/octet-stream");
        //response.addHeader("Content-Disposition", "attachment;filename=" + URLEncoder.encode(fileName, "UTF-8"));
        PrintWriter out = response.getWriter();
        System.out.println("使用BufferedReader读取文本文件......");
        while((line = br.readLine()) != null){
            //逐行读取
            System.out.println(line);
            out.print(line);
        }
        br.close();
    }

    @RequestMapping("/readScanner")
    public void readScanner(String fileName, HttpServletResponse response) throws IOException{
        Path path = Paths.get(fileName);
        Scanner scanner = new Scanner(path);
        System.out.println("使用Scanner读取文本文件.....");
        //将注释去掉，重新运行启动项目，在浏览器键入要读取的文件地址，观察下效果有什么不一样。
        //response.reset();
        //response.setContentType("application/octet-stream");
        //response.addHeader("Content-Disposition", "attachment;filename=" + URLEncoder.encode(fileName, "UTF-8"));
        PrintWriter out = response.getWriter();
        //逐行读取
        while(scanner.hasNextLine()){
            //逐行处理
            String line = scanner.nextLine();
            System.out.println(line);
            out.print(line);
        }
        scanner.close();
    }


    @RequestMapping("/readUsingRandomAccessFile")
    public void readUsingRandomAccessFile(String fileName, HttpServletResponse response) throws IOException{
        RandomAccessFile file = new RandomAccessFile(fileName, "r");
        String str;
        //将注释去掉，重新运行启动项目，在浏览器键入要读取的文件地址，观察下效果有什么不一样。
        //response.reset();
        //response.setContentType("application/octet-stream");
        //response.addHeader("Content-Disposition", "attachment;filename=" + URLEncoder.encode(fileName, "UTF-8"));
        PrintWriter out = response.getWriter();
        while ((str = file.readLine()) != null) {
            System.out.println("使用RandomAccessFile来实现断点续传读取/下载文件......");
            System.out.println(str);
            out.print(str);
```

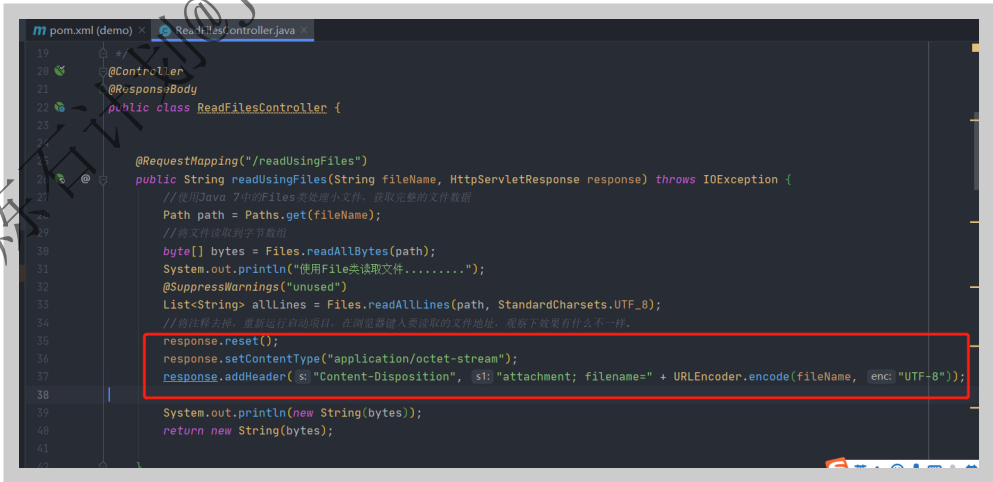
```
        }
        file.close();
    }

    @RequestMapping("/readUsingCommonsIo")
    public String readUsingCommonsIo(String
fileName,HttpServletResponse response) throws IOException{
        File file = new File(fileName);
        //将注释去掉，重新运行启动项目，在浏览器键入要读取的文件地址，观察下效果有什
么不一样。
        response.reset();
        response.setContentType("application/octet-stream");
        response.addHeader("Content-Disposition", "attachment;
filename=" + URLEncoder.encode(fileName, "UTF-8"));
        System.out.println("使用Commons-io读取文件......");
        System.out.println(FileUtils.readFileToString(file,
StandardCharsets.UTF_8));
        return FileUtils.readFileToString(file,
StandardCharsets.UTF_8);
    }

}
```

⑧、启动运行项目，打开浏览器，访问其中一个读取/下载文件的接口，如下图所示：

```
http://127.0.0.1:8080/readUsingRandomAccessFile?
fileName=C:/Users/powerful/Desktop/test.txt
```

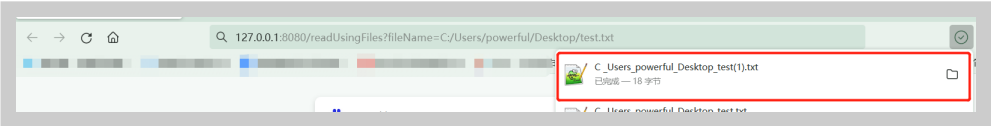


⑨、在代码文件中，每个接口我都留了相同的注释，我们选择一个接口将注释去除掉，如下图所示：



⑩、再次启动运行，访问去掉注释的这个接口，我们可以看到将文件下载到了本地，如下图所示：

```
http://127.0.0.1:8080/readUsingFiles?
fileName=C:/Users/powerful/Desktop/test.txt
```



至此，文件读取/下载的几种方式讲完了。

一定要动手实践调试。