(4)——/

/

1.

A.

B. excelwordpdf

C.

2.

1.excel

2.

3.()

4.

1.

2.OutputStream

3.

OutputStream

1.wrapper

2.

EncryptOutputStream.java

```
package com.tptj.demo.hg.export.encrypt;

import com.fr.intelli.record.Focus;
import com.fr.record.analyzer.EnableMetrics;
import com.fr.third.org.bouncycastle.util.Arrays;
import javax.servlet.ServletOutputStream;
import javax.servlet.WriteListener;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
```

```
 * @author
 * @version 10.0
 * Created by  on 2021/7/15
 **/
@EnableMetrics
public class EncryptOutputStream extends ServletOutputStream {

    private OutputStream original = null;
    private boolean closed = false;
    private ByteArrayOutputStream crypt = new ByteArrayOutputStream();

    public EncryptOutputStream( OutputStream original ){
        this.original = original;
    }

    /**
     *
     * @param bytes
     * @return
     */
    @Focus(id="com.tptj.demo.hg.export.encrypt.v10",text="Export Encrypt")
    private byte[] encrypt(byte[] bytes){
        //
        return Arrays.reverse(bytes);
    }

    @Override
    public void write(int b) throws IOException {
        if (closed) {
            throw new IOException("Cannot write to a closed output stream");
        }
        crypt.write(b);
    }

    @Override
    public void close() throws IOException {
        if( !closed ){
            byte[] bytes = crypt.toByteArray();
            bytes = encrypt(bytes);
            original.write(bytes);
            original.flush();
            original.close();
            closed = true;
        }
    }

    @Override
    public void flush() throws IOException {
        crypt.flush();
    }

    @Override
    public void write(byte[] data) throws IOException {
        write(data, 0, data.length);
    }

    @Override
    public void write(byte[] data, int off, int len) throws IOException {
        if (closed) {
            throw new IOException("Cannot write to a closed output stream");
        }
        crypt.write(data, off, len);
    }

    @Override
    public boolean isReady() {
        return true;
    }

    @Override
    public void setWriteListener(WriteListener listener) {
```

```
    }

}
```

com.fr.report.fun.ExportOperateProvidercpt

(1)ExporterOutputStreamExporterWrapperExporterOutputStreamWrapper

EncryptExporter.java

```
package com.tptj.demo.hg.export.encrypt.demo1;

import com.fr.io.exporter.AppExporter;
import com.fr.main.workbook.ResultWorkBook;
import com.fr.page.PageSetCreator;
import com.fr.page.PageSetProvider;
import com.fr.web.core.ReportRepositoryDeal;
import com.tptj.demo.hg.export.encrypt.EncryptOutputStream;

import java.io.OutputStream;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/7/15
 **/
public class EncryptExporter implements AppExporter {

    private AppExporter original;

    public EncryptExporter(AppExporter original){
        this.original = original;
    }
    @Override
    public void export( OutputStream out, ResultWorkBook book ) throws Exception {
        original.export( new EncryptOutputStream(out), book );
    }

    @Override
    public void export( OutputStream out, PageSetProvider pageSet ) throws Exception {
        original.export( new EncryptOutputStream(out), pageSet );
    }

    @Override
    public void export(OutputStream out, ResultWorkBook book, PageSetCreator creator, ReportRepositoryDeal
repo, int[] sheets) throws Exception {
        original.export( new EncryptOutputStream(out), book,creator,repo,sheets);
    }

    @Override
    public void export(OutputStream out, ResultWorkBook book, PageSetProvider pageSet, ReportRepositoryDeal
repo, int[] sheets) throws Exception {
        original.export( new EncryptOutputStream(out), book,pageSet,repo,sheets);
    }

    @Override
    public void setVersion(Object o) {
        original.setVersion(o);
    }
}
```

EncryptExcelOperate.java

```
package com.tptj.demo.hg.export.encrypt.demo1;

import com.fr.io.collection.ExportCollection;
import com.fr.io.exporter.AppExporter;
import com.fr.stable.web.SessionProvider;
import com.fr.web.core.reserve.ExcelOperate;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/7/15
 **/
public class EncryptExcelOperate extends ExcelOperate {

    @Override
    public ExportCollection newExportCollection(HttpServletRequest req, HttpServletResponse res,
                                                SessionProvider sessionIDInfor, String filename ){
        ExportCollection collection = super.newExportCollection(req, res, sessionIDInfor, filename);
        AppExporter exporter = collection.getExporter();
        collection.setExporter( new EncryptExporter(exporter) );
        return collection;
    }
}
```

com.fr.report.fun.ExportExtensionProcessorcpt

(1)HTTPExportExtensionProcessorExportOperateProviderExportExtensionProcessorcptExportOperateProvider

Demo.java

```
package com.tptj.demo.hg.export.encrypt.demo2;

import com.fr.io.collection.ExportCollection;
import com.fr.io.exporter.AppExporter;
import com.fr.web.core.ReportSessionIDInfor;
import com.fr.web.core.reserve.DefaultExportExtension;
import com.tptj.demo.hg.export.encrypt.demo1.EncryptExporter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/7/15
 **/
public class Demo extends DefaultExportExtension {

    @Override
    public ExportCollection createCollection(HttpServletRequest req, HttpServletResponse res,
                                             ReportSessionIDInfor info, String format, String filename, boolean
isEmbed) throws Exception {
        ExportCollection collection = super.createCollection(req, res, info, format,filename,isEmbed);
        AppExporter exporter = collection.getExporter();
        collection.setExporter( new EncryptExporter(exporter) );
        return collection;
    }
}
```

com.fr.decision.fun.GlobalRequestFilterProviderHTTP

wrapper

DecryptRequest.java

```
package com.tptj.demo.hg.export.encrypt.demo3;

import com.fr.general.IOUtils;
import com.fr.third.org.bouncycastle.util.Arrays;

import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/7/15
 **/
public class DecryptRequest extends HttpServletRequestWrapper {
    private byte[] data;
    public DecryptRequest(HttpServletRequest request) {
        super(request);
        init();
    }

    private void init(){
        try{
            data = IOUtils.inputStream2Bytes(getRequest().getInputStream());
            //()
            data = Arrays.reverse(data);
        }catch(Exception e){
            data = new byte[0];
        }
    }

    @Override
    public ServletInputStream getInputStream() throws IOException {
        try{
            return new DelegatingServletInputStream(new ByteArrayInputStream( data ));
        }catch (Exception e){
            return null;
        }
    }

    @Override
    public BufferedReader getReader()throws IOException{
        return new BufferedReader(new InputStreamReader(new ByteArrayInputStream( data )));
    }
}
```

EncryptResponse.java

```
package com.tptj.demo.hg.export.encrypt.demo3;

import com.tptj.demo.hg.export.encrypt.EncryptOutputStream;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletResponseWrapper;
import java.io.IOException;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/7/15
 **/
public class EncryptResponse extends HttpServletResponseWrapper {

    public EncryptResponse(HttpServletResponse response) {
        super(response);
    }

    public ServletOutputStream getOutputStream()throws IOException {
        ServletOutputStream original =  super.getOutputStream();
        return new EncryptOutputStream(original);
    }

}
```

SDKdemodemo

3DES\AES

demo-export-encrypt