(1)——

TemplateEncryptProviderTemplateEncryptProvider

UI

BaseInterceptor.java

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

import com.fr.base.io.EncryptIOFileProxy;
import com.fr.base.io.IOFile;
import com.fr.file.FILE;
import com.fr.invoke.Reflect;
import com.fr.log.FineLoggerFactory;
import com.tptj.demo.hg.template.encrypt.ui.DecodeDialog;
import com.tptj.demo.hg.template.encrypt.ui.SpEncryptAttrMark;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/12
 **/
public class BaseInterceptor {
    /**
     *
     * @param file
     */
    public static void intercept4Edit(FILE file){
        try {
            file.asInputStream();
            if( !DemoEncryptor.hasSecret() ){
                return;
            }
            String secret = DemoEncryptor.getSecret();
            secret = new DecodeDialog(file,secret).getSecret();
            DemoEncryptor.setSecret(secret);
        } catch (Exception e) {
            FineLoggerFactory.getLogger().error(e,e.getMessage());
        }
    }

    /**
     *
     * @param origin
     */
    public static void intercept4Save( EncryptIOFileProxy origin ){
        IOFile report = Reflect.on(origin).get("file");
        SpEncryptAttrMark mark = report.getAttrMark(SpEncryptAttrMark.XML_TAG);
        if( null != mark ){
            DemoEncryptor.setSecret(mark.getSecret());
        }
    }
}
```

SpCptApp.java

```
package com.fr.design.mainframe.app;

import com.fr.file.FILE;
import com.fr.main.impl.WorkBook;
import com.tptj.demo.hg.template.encrypt.BaseInterceptor;
import com.tptj.tool.hg.dynamic.agent.source.AccessPoint;
import com.tptj.tool.hg.dynamic.agent.source.Context;
import com.tptj.tool.hg.dynamic.agent.source.Source;
import com.tptj.tool.hg.dynamic.agent.version.ModuleReport;
import com.tptj.tool.hg.dynamic.agent.version.VersionChecker;


/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/11
 * CptApp#asIOFile cptUI
 **/
@VersionChecker(value = 10020210101l, loader = ModuleReport.class)
@Source(CptApp.class)
public class SpCptApp extends Context {

    @AccessPoint
    public WorkBook asIOFile(FILE file, boolean needCheck) {
        BaseInterceptor.intercept4Edit(file);
        return null;
    }

}
```

SpFormApp.java

```
package com.fr.design.mainframe.app;

import com.fr.file.FILE;
import com.fr.form.main.Form;
import com.tptj.demo.hg.template.encrypt.BaseInterceptor;
import com.tptj.tool.hg.dynamic.agent.source.AccessPoint;
import com.tptj.tool.hg.dynamic.agent.source.Context;
import com.tptj.tool.hg.dynamic.agent.source.Source;
import com.tptj.tool.hg.dynamic.agent.version.ModuleReport;
import com.tptj.tool.hg.dynamic.agent.version.VersionChecker;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/12
 * FormApp#asIOFile formUI
 **/
@VersionChecker(value = 10020210101l, loader = ModuleReport.class)
@Source(FormApp.class)
public class SpFormApp extends Context {

    @AccessPoint
    public Form asIOFile(FILE file) {
        BaseInterceptor.intercept4Edit(file);
        return null;
    }

}
```

SpEncryptIOFileProxy.java

```
package com.fr.design.mainframe.app;

import com.fr.base.io.EncryptIOFileProxy;
import com.tptj.demo.hg.template.encrypt.BaseInterceptor;
import com.tptj.tool.hg.dynamic.agent.source.AccessPoint;
import com.tptj.tool.hg.dynamic.agent.source.Context;
import com.tptj.tool.hg.dynamic.agent.source.Source;
import com.tptj.tool.hg.dynamic.agent.version.ModuleReport;
import com.tptj.tool.hg.dynamic.agent.version.VersionChecker;

import java.io.OutputStream;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/11
 * EncryptIOFileProxy#export
 **/
@VersionChecker(value = 10020210101l, loader = ModuleReport.class)
@Source(EncryptIOFileProxy.class)
public class SpEncryptIOFileProxy extends Context {

    @AccessPoint
    public boolean export(OutputStream out) throws Exception {
        BaseInterceptor.intercept4Save(getOrigin());
        return false;
    }

}
```

LifeCycle.java

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

import com.fr.design.mainframe.app.SpCptApp;
import com.fr.design.mainframe.app.SpEncryptIOFileProxy;
import com.fr.design.mainframe.app.SpFormApp;
import com.fr.intelli.record.Focus;
import com.fr.plugin.context.PluginContext;
import com.fr.plugin.observer.inner.AbstractPluginLifecycleMonitor;
import com.fr.record.analyzer.EnableMetrics;
import com.tptj.tool.hg.dynamic.agent.base.DynamicAgent;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/11
 **/
@EnableMetrics
public class LifeCycle extends AbstractPluginLifecycleMonitor {
    public static final String PLUGIN_ID = "com.tptj.demo.hg.template.encrypt.v10";
    public static final String PLUGIN_NAME = "template encrypt";

    @Override
    @Focus(id = PLUGIN_ID, text = PLUGIN_NAME)
    public void afterRun(PluginContext context) {
        DynamicAgent.getInstance().register(
                new SpEncryptIOFileProxy(),
                new SpCptApp(),
                new SpFormApp());
    }

    @Override
    public void beforeStop(PluginContext context) {
        DynamicAgent.getInstance().reset();
    }
}
```

IOFileAttrMarkMenuHandler

SpEncryptAttrMark.java

```
package com.tptj.demo.hg.template.encrypt.ui;

import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.stable.StringUtils;
import com.fr.stable.fun.impl.AbstractIOFileAttrMark;
import com.fr.stable.xml.XMLPrintWriter;
import com.fr.stable.xml.XMLableReader;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/12
 **/
public class SpEncryptAttrMark extends AbstractIOFileAttrMark {
    public final static String XML_TAG = "SpEncryptAttrMark";

    private String secret;

    public String getSecret() {
        return secret;
    }

    public void setSecret(String secret) {
        this.secret = secret;
    }

    @Override
    public String xmlTag() {
        return XML_TAG;
    }

    @Override
    public void readXML(XMLableReader reader) {
        String tag = reader.getTagName();
        if( XML_TAG.equals(tag) ){
            secret = reader.getAttrAsString("secret", StringUtils.EMPTY);
        }
    }

    @Override
    public void writeXML(XMLPrintWriter writer) {
        writer.startTAG(XML_TAG).attr("secret",secret).end();
    }

    @Override
    public SpEncryptAttrMark clone() {
        SpEncryptAttrMark obj = (SpEncryptAttrMark)super.clone();
        obj.secret = secret;
        return obj;
    }

    @Override
    public JSONObject createJSONConfig() throws JSONException {
        JSONObject json = super.createJSONConfig();
        json.put("secret",secret);
        return json;
    }
}
```

MenuEncryptor.java

```
package com.tptj.demo.hg.template.encrypt.ui;

import com.fr.design.fun.impl.AbstractMenuHandler;
import com.fr.design.mainframe.DesignerContext;
import com.fr.design.mainframe.JTemplate;
import com.fr.design.mainframe.toolbar.ToolBarMenuDockPlus;
import com.fr.design.menu.ShortCut;

/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/12
 **/
public class MenuEncryptor extends AbstractMenuHandler {
    private static final int INSERT_POSITION = 2;
    @Override
    public int insertPosition(int i) {
        return INSERT_POSITION;
    }

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

    @Override
    public boolean insertSeparatorAfter() {
        return false;
    }

    @Override
    public String category() {
        return TEMPLATE;
    }

    @Override
    public ShortCut shortcut(){
        JTemplate template = DesignerContext.getDesignerFrame().getSelectedJTemplate();
        return shortcut(template);
    }
    @Override
    public ShortCut shortcut(ToolBarMenuDockPlus plus) {
        //ToolBarMenuDockPlus.
        if (!(plus instanceof JTemplate)){
            return null;
        }
        return new MenuEncryptAction( (JTemplate)plus );
    }
}
```

TemplateEncryptProvider

DemoEncryptor.java

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

import com.fr.general.CommonIOUtils;
import com.fr.report.fun.impl.AbstractTemplateEncryptProvider;
import com.fr.security.SecurityToolbox;
import com.fr.stable.CodeUtils;
import com.fr.stable.StringUtils;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
```

```
/**
 * @author
 * @version 10.0
 * Created by  on 2021/9/12
 **/
public class DemoEncryptor extends AbstractTemplateEncryptProvider {

    public static final String TAG_START = "-----Encrypt Start-----";
    public static final String TAG_END = "-----Encrypt End-----";

    public static ThreadLocal<String> holder = new ThreadLocal<String>();
    public static ThreadLocal<Integer> pos = new ThreadLocal<Integer>();
    public static ThreadLocal<Boolean> init = new ThreadLocal<Boolean>();

    public static void setSecret(String secret) {
        holder.set(secret);
    }

    public static String getSecret() {
        return holder.get();
    }

    public static boolean hasSecret() {
        return StringUtils.isNotEmpty(getSecret());
    }

    public static int getPos(){
        return pos.get();
    }

    public static void setPos(int idx){
        pos.set(idx);
    }

    public static void initData(InputStream in){
        init.set(true);
        String data = StringUtils.EMPTY;
        try{
            data = CommonIOUtils.inputStream2String(in);
            if(!data.startsWith(TAG_START)){
                setSecret(StringUtils.EMPTY);
                setPos(-1);
                return;
            }
            int end = data.indexOf(TAG_END);
            String part = data.substring(TAG_START.length(),end);
            String sha = part.substring(0,64);
            String code = part.substring(64);
            String secret = CodeUtils.passwordDecode("___"+code);
            if( !CodeUtils.md5Encode(secret, StringUtils.EMPTY,"SHA-256").equals(sha) ){
                setSecret(StringUtils.EMPTY);
                setPos(-1);
                return;
            }
            setSecret(secret);
            setPos(end+TAG_END.length());
        }catch(Exception e){
            setSecret(StringUtils.EMPTY);
            setPos(-1);
        }
    }

    private String codePassword(){
        String secret = getSecret();
        String code = CodeUtils.passwordEncode(secret).substring(3);
        String sha = CodeUtils.md5Encode(secret, StringUtils.EMPTY,"SHA-256");
        return sha+code;
    }

    @Override
```

```
    public InputStream encode(InputStream in) {
        if( !hasSecret() ){
            return in;
        }
        byte[] bytes = CommonIOUtils.inputStream2Bytes(in);
        try{
            String secret = getSecret();
            String data = new String(bytes,"UTF-8");
            data = TAG_START +codePassword()+TAG_END+
                    SecurityToolbox.aesEncrypt( data, secret);
            return new ByteArrayInputStream(data.getBytes("UTF-8"));
        }catch (Exception e){
            return new ByteArrayInputStream(bytes);
        }
    }

    @Override
    public InputStream decode(InputStream in) {
        byte[] bytes = CommonIOUtils.inputStream2Bytes(in);
        try{
            if( !Boolean.TRUE.equals(init.get()) ){
               initData(new ByteArrayInputStream(bytes));
            }
            if( !hasSecret() ){
                return new ByteArrayInputStream(bytes);
            }
            String data = new String(bytes,"UTF-8");
            data = data.substring( getPos() );
            data = SecurityToolbox.aesDecrypt( data, getSecret() );
            return new ByteArrayInputStream(data.getBytes("UTF-8"));
        }catch (Exception e){
            return new ByteArrayInputStream(bytes);
        }
    }

}
```

1.TemplateEncryptProviderIDThreadLocal

2.

3.

demo-template-encrypt