ErrorHandler

```
package com.fr.stable.fun;

import com.fr.stable.fun.mark.Immutable;

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

/**
 *
 */
public interface ErrorHandler extends Immutable {

    String MARK_STRING = "ErrorHandler";

    int CURRENT_LEVEL = 1;

    /**
     *
     *
     * @param req       HTTP
     * @param res       HTTP
     * @param throwable
     */
    void error(HttpServletRequest req, HttpServletResponse res, Throwable throwable);

    /**
     *
     *
     * @param req     HTTP
     * @param res     HTTP
     * @param message
     */
    void error(HttpServletRequest req, HttpServletResponse res, String message);
}
```

DefaultErrorHandler

```
package com.fr.web.core;

import com.fr.base.ConfigManager;
import com.fr.base.Env;
import com.fr.base.FRContext;
import com.fr.base.TemplateUtils;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.stable.CodeUtils;
import com.fr.stable.ProductConstants;
import com.fr.stable.StableUtils;
import com.fr.stable.StringUtils;
import com.fr.stable.fun.ErrorHandler;
import com.fr.stable.fun.impl.AbstractErrorHandler;
import com.fr.web.utils.WebUtils;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import java.util.HashMap;
import java.util.Map;

/**
 *
 */
public class DefaultErrorHandler extends AbstractErrorHandler {
    /**
     *
     */
    public DefaultErrorHandler() {
    }

    /**
     *
     *
     * @param req       HTTP
     * @param res       HTTP
     * @param throwable
     */
    public void error(HttpServletRequest req, HttpServletResponse res, Throwable throwable) {
        error(req, res, throwable.getMessage(), throwable);
    }

    /**
     *
     *
     * @param req     HTTP
     * @param res     HTTP
     * @param message
     */
    public void error(HttpServletRequest req, HttpServletResponse res, String message) {
        error(req, res, message, null);
    }

    /**
     *
     *
     * @param req       HTTP
     * @param res       HTTP
     * @param message
     * @param throwable
     */
    public void error(HttpServletRequest req, HttpServletResponse res,
                      String message, Throwable throwable) {
        if (WebUtils.getDevice(req).isMobile()) {
            postErrorMessageWithJSONObject(req, res, message, throwable);
        } else {
            postErrorMessageOnPC(req, res, message, throwable);
        }
    }

    private void postErrorMessageOnPC(HttpServletRequest req, HttpServletResponse res, String message,
Throwable throwable) {
        Map<String, Object> map = new HashMap<String, Object>();
        if (message == null) {//messagenull.
            message = StringUtils.EMPTY;
        }
        map.put("message", CodeUtils.htmlEncode(CodeUtils.cjkEncode(message)));
        map.put("duration", ProductConstants.HISTORY);
        map.put("companyname", ProductConstants.COMPANY_NAME);
        try {
            if (req.getParameter("frload") != null) {
                //loading.js,html,alert
                PrintWriter writer = WebUtils.createPrintWriter(res);
                writer.println(new StringBuffer().append("alert(\"").append(message.replaceAll("\"", "\\\\\"")).
append("\")"));
                writer.flush();
```

```
                writer.close();
                return;
            }

            if (throwable != null) {
                map.put("exception", createThrowableMessage(throwable, true));
            }
            //james:errorFRContext
            map.put("charset", CodeUtils.htmlEncode(ConfigManager.getProviderInstance().getServerCharset()));

            String errorTemplate = "/com/fr/web/core/errorIframe.html";

            //
            if (StringUtils.isNotEmpty(ConfigManager.getProviderInstance().getErrorTemplate())) {
                try {
                    userDefineHandler(req, res, map);
                } catch (Exception e) {
                    WebUtils.writeOutTemplate(errorTemplate, res, map);
                    FRContext.getLogger().error(e.getMessage());
                }
            } else {
                WebUtils.writeOutTemplate(errorTemplate, res, map);
            }
        } catch (IOException ioe) {
            FRContext.getLogger().errorWithServerLevel(ioe.getMessage(), ioe);
        }
    }

    private void postErrorMessageWithJSONObject(HttpServletRequest req, HttpServletResponse res, String
message, Throwable throwable) {
        JSONObject errorJSON = new JSONObject();
        PrintWriter writer = null;
        try {
            errorJSON.put("message", message);
            errorJSON.put("exception", createThrowableMessage(throwable, false));
            writer = WebUtils.createPrintWriter(res);
            writer.println(errorJSON);
            writer.flush();
            writer.close();
        } catch (Exception e) {
            try {
                errorJSON.put("exception", createThrowableMessage(e.getCause(), false));
            } catch (JSONException e1) {
                FRContext.getLogger().error(e.getMessage(), e);
            }
        } finally {
            if (writer != null) {
                writer.flush();
                writer.close();
            }
        }
    }

    private String createThrowableMessage(Throwable throwable, boolean useCJK) {
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);
        throwable.printStackTrace(printWriter);
        try {
            return useCJK ? CodeUtils.htmlEncode(CodeUtils.cjkEncode(stringWriter.toString())) : stringWriter.
toString();
        } finally {
            printWriter.flush();
            printWriter.close();
        }
    }

    private void userDefineHandler(HttpServletRequest req, HttpServletResponse res, Map map) throws Exception {
        Env currentEnv = FRContext.getCurrentEnv();
        if (currentEnv == null) {
            throw new IllegalStateException("Env is null..");
        }
```

```
        String errorTemplate = ConfigManager.getProviderInstance().getErrorTemplate();

        if (StringUtils.isNotEmpty(errorTemplate) && !errorTemplate.endsWith(".html")) {
            if (errorTemplate.length() > 0 && errorTemplate.charAt(0) == '/') {
                errorTemplate = req.getContextPath() + errorTemplate;
            }

            map.put("redirectURL", errorTemplate);
            WebUtils.writeOutTemplate("/com/fr/web/core/errorPostRedirect.html", res, map);
        } else {
            String envPath = currentEnv.getPath();
            File htmlFile = new File(StableUtils.pathJoin(new String[]{new File(envPath).getParent(),
errorTemplate}));

            PrintWriter printWriter = WebUtils.createPrintWriter(res);
            TemplateUtils.dealWithTemplate(new FileInputStream(htmlFile), StableUtils.RESOURCE_ENCODER,
printWriter, map);

            printWriter.flush();
            printWriter.close();
        }
    }
}
```

```
<extra-core>
   <ErrorHandler class="com.fr.plugin.xxx.youclassname"/>
</extra-core>
```