

RequestParameterCollector

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

import javax.servlet.http.HttpServletRequest;
import java.util.Map;

/**
 * @author : richie
 * @since : 8.0
 */
public interface RequestParameterCollector extends Level {

    String XML_TAG = "RequestParameterCollector";

    int CURRENT_LEVEL = 1;

    /**
     * HTTP
     *
     * @param req HTTP
     * @return
     */
    Map<String, Object> getParametersFromSession(HttpServletRequest req);

    /**
     * HTTP
     *
     * @param req HTTP
     * @return
     */
    Map<String, Object> getParametersFromAttribute(HttpServletRequest req);

    /**
     * HTTP
     *
     * @param req HTTP
     * @return
     */
    Map<String, Object> getParametersFromReqInputStream(HttpServletRequest req);

    /**
     *
     *
     * @param req HTTP
     * @return
     */
    Map<String, Object> getParametersFromParameter(HttpServletRequest req);

    /**
     * JSON
     *
     * @param req HTTP
     * @return
     */
    Map<String, Object> getParametersFromJSON(HttpServletRequest req, Map<String, Object> parameterMap);
}
```

DefaultRequestParameterCollector

```
package com.fr.web.utils;

import com.fr.general.FRLogger;
import com.fr.general.GeneralUtils;
import com.fr.general.IOUtils;
import com.fr.general.http.HttpClient;
import com.fr.general.web.ParameterConsts;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.stable.CodeUtils;
import com.fr.stable.EncodeConstants;
import com.fr.stable.StringUtils;
import com.fr.stable.fun.impl.AbstractRequestParameterCollector;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.io.InputStream;
import java.net.URLDecoder;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

/**
 * @author : richie
 * @since : 8.0
 */
public class DefaultRequestParameterCollector extends AbstractRequestParameterCollector {
    private static DefaultRequestParameterCollector instance;

    public static synchronized DefaultRequestParameterCollector getInstance() {
        if (instance == null) {
            instance = new DefaultRequestParameterCollector();
        }
        return instance;
    }

    public Map<String, Object> getParametersFromSession(HttpServletRequest req) {
        HttpSession session = req.getSession(false);
        Map<String, Object> map = new HashMap<String, Object>();
        if (session == null) {
            return map;
        }
        Enumeration hsessionEnum = session.getAttributeNames();
        while (hsessionEnum.hasMoreElements()) {
            Object nextElem = hsessionEnum.nextElement();
            String decodedElem = CodeUtils.decodeText("'" + nextElem);

            map.put(decodedElem.toUpperCase(), session.getAttribute("'" + nextElem));
        }
        return map;
    }

    public Map<String, Object> getParametersFromAttribute(HttpServletRequest req) {
        Map<String, Object> map = new HashMap<String, Object>();
        Enumeration attrEnum = req.getAttributeNames();
        while (attrEnum.hasMoreElements()) {
            Object nextElem = attrEnum.nextElement();
            String decodedElem = CodeUtils.decodeText("'" + nextElem);

            try {
                if (map.containsKey(decodedElem.toUpperCase())) {
                    map.remove(decodedElem.toUpperCase());
                }
                map.put(decodedElem, WebUtils.getHTTPRequestParameter(req, "'" + nextElem));
            } catch (Exception e) {
            }
        }
    }
}
```

```

        // alex: do nothing
    }
}
return map;
}

public Map<String, Object> getParametersFromParameter(HttpServletRequest req) {
    Map<String, Object> parameterMap = new HashMap<String, Object>();
    Enumeration paramEnum = req.getParameterNames();
    while (paramEnum.hasMoreElements()) {
        Object nextElem = paramEnum.nextElement();
        String decodedElem = CodeUtils.decodeText("'" + nextElem);

        try {
            if (parameterMap.containsKey(decodedElem.toUpperCase())) {
                parameterMap.remove(decodedElem.toUpperCase());
            }
            parameterMap.put(decodedElem, WebUtils.getHttpRequestParameter(req, "'" + nextElem));
            if ("reportlet".equalsIgnoreCase(decodedElem)) {
                parameterMap.put("reportName", WebUtils.getHttpRequestParameter(req, "'" + nextElem));
            }
            if ("formlet".equalsIgnoreCase(decodedElem)) {
                parameterMap.put("formletName", WebUtils.getHttpRequestParameter(req, "'" + nextElem));
            }

            if ("chartlet".equalsIgnoreCase(decodedElem)) {
                parameterMap.put("chartName", WebUtils.getHttpRequestParameter(req, "'" + nextElem));
            }
            //encode, decode
            parameterMap.put(URLDecoder.decode(decodedElem, EncodeConstants.ENCODING_UTF_8), WebUtils.
getHttpRequestParameter(req, "'" + nextElem));
        } catch (Exception e) {
            // alex: do nothing
        }
    }
    return parameterMap;
}

public Map<String, Object> getParametersFromJSON(HttpServletRequest req, Map<String, Object> parameterMap) {
    Map<String, Object> map = new HashMap<String, Object>();
    Object typeSensitiveParameter = parameterMap.get(ParameterConsts.__PARAMETERS__);
    if (typeSensitiveParameter instanceof String) {
        try {
            map = GeneralUtils.jsonString2Map((String) typeSensitiveParameter);
            // alex:java.util.Map,JSON
            // shoc:session,,bug31092
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                String key = (String) entry.getKey();
                if (parameterMap.containsKey(key.toUpperCase())) {
                    parameterMap.remove(key.toUpperCase());
                }
            }
            parameterMap.remove(ParameterConsts.__PARAMETERS__);
        } catch (JSONException e) {
            FRLogger.getLogger().error(e.getMessage(), e);
        }
    }
    return map;
}

/**
 *
 * @since 2014-9-25
 * @category TODO:
 *
 * @see com.fr.stable.fun.RequestParameterCollector#getParametersFromReqInputStream(javax.servlet.http.
HttpServletRequest)
 *
 */

```

```
public Map<String, Object> getParametersFromReqInputStream(HttpServletRequest req) {
    Map<String, Object> map = new HashMap<String, Object>();
    InputStream in;
    try {
        String paraJson = (String) req.getAttribute(HttpClient.CLOSED);
        if(paraJson == null){
            in = req.getInputStream();

            if(in == null){
                return map;
            }
            paraJson = IOUtils.inputStream2String(in);
            //
            req.setAttribute(HttpClient.CLOSED, paraJson);
        }

        if(StringUtils.isEmpty(paraJson)){
            return map;
        }

        JSONObject jo = new JSONObject(paraJson);
        Iterator<String> it = jo.keys();
        while (it.hasNext()) {
            String key = it.next();
            map.put(key, jo.get(key));
        }
    } catch (Exception e) {
    } catch (Error e){
    }

    return map;
}
}
```

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