

1-

1.

XMLFileManagerreadXmlwriteXmlxmlxmlxmlFileManager

xmlXmlLablereadXmlwriteXmlreadXmlwriteXml

ConfigManager private String resServerPathxmlXmllableprivate EmbeddedTableData embeddedTableDataembeddedTableData.readXmlwriteXmlprivate
Map styleMap = new ListMap(); keyStringValueStyleStyleStyleStyle

2.

ConfigHolderConfigHolderAPIholderholder

(propertynameSpace,propertypropertyIdentifier)

```
package com.fr.config.holder.factory;

import com.fr.config.holder.Conf;
import com.fr.config.holder.impl.ColConf;
import com.fr.config.holder.impl.MapConf;
import com.fr.config.holder.impl.ObjConf;
import com.fr.config.holder.impl.ObjectColConf;
import com.fr.config.holder.impl.ObjectMapConf;
import com.fr.config.holder.impl.SimConf;

import java.util.Collection;
import java.util.Map;

/**
 *
 * http://www.finedevelop.com/pages/viewpage.action?pageId=18648778
 */
public class Holders {
    /**
     * HolderTConfigHolder
     *
     * @param property
     * @param t
     * @param nameSpace
     * @param <T>
     * @return
     * @see Holders#simple(Object)
     */
    @Deprecated
    public static <T> Conf<T> simple(String property, T t, String nameSpace) {
        return new SimConf<T>(property, t).setNameSpace(nameSpace);
    }

    public static <T> Conf<T> simple(T t) {
        return new SimConf<T>(t);
    }

    /**
     * THolderTConfigHolder
     *
     * @param property
     * @param t
     * @param type
     * @param nameSpace
     * @param <T>
     * @return
     * @see Holders#obj(Object, Class)
     */
    @Deprecated
    public static <T> Conf<T> obj(String property, T t, Class<T> type, String nameSpace) {
        return new ObjConf<T>(property, t, type).setNameSpace(nameSpace);
    }
}
```

```

public static <T> Conf<T> obj(T t, Class<T> type) {
    return new ObjConf<T>(t, type);
}

/**
 * KCollectionHolderConfigHolder
 *
 * @param property
 * @param collection
 * @param valueType
 * @param nameSpace
 * @param <K>
 * @return
 * @see Holders#objCollection(Collection, Class)
 */
@Deprecated
@SuppressWarnings("unchecked")
public static <K> Conf<Collection<K>> collection(String property, Collection<K> collection, Class<K>
valueType, String nameSpace) {
    return new ColConf(property, collection, valueType).setNameSpace(nameSpace);
}

@SuppressWarnings("unchecked")
public static <K> ColConf<Collection<K>> collection(Collection<K> collection, Class<K> valueType) {
    return new ColConf(collection, valueType);
}

/**
 * TCollectionHolderTConfigHolder
 *
 * @param property
 * @param collection
 * @param type
 * @param nameSpace
 * @param <T>
 * @return
 * @see Holders#collection(Collection, Class)
 */
@Deprecated
public static <T> ObjectColConf<Collection<T>> objCollection(String property, Collection<T> collection,
Class<T> type, String nameSpace) {
    return new ObjectColConf<Collection<T>>(property, collection, type).setNameSpace(nameSpace);
}

public static <T> ObjectColConf<Collection<T>> objCollection(Collection<T> collection, Class<T> type) {
    return new ObjectColConf<Collection<T>>(collection, type);
}

public static <T> ObjectColConf<Collection<T>> objCollection(Collection<T> collection, Class<T> type,
boolean order) {
    return new ObjectColConf<Collection<T>>(collection, type, order);
}

/**
 * keyvalueMapHolderK,VConfigHolder,keyTypevalueType
 *
 * @param property
 * @param map
 * @param keyType
 * @param valueType
 * @param nameSpace
 * @param <K>
 * @param <V>
 * @return
 * @see Holders#map(Map, Class, Class)
 */
@Deprecated
@SuppressWarnings("unchecked")
public static <K, V> MapConf<Map<K, V>> map(String property, Map<K, V> map, Class<K> keyType, Class<V>

```

```

valueType, String nameSpace) {
    return new MapConf(property, map, keyType, valueType).setNameSpace(nameSpace);
}

@SuppressWarnings("unchecked")
public static <K, V> MapConf<Map<K, V>> map(Map<K, V> map, Class<K> keyType, Class<V> valueType) {
    return new MapConf(map, keyType, valueType);
}

/**
 * valueVMapHolderVConfigHolder
 * K KKString,KValueWriterKString
 * KValueReaderStringKValueReader.registerReader, ValueWriter.registerWriter
 * keyTypemapkeyStringValue
 *
 * @param property
 * @param map
 * @param keyType
 * @param valueType
 * @param nameSpace
 * @param <K>
 * @param <V>
 * @return
 * @see Holders#objMap(Map, Class, Class)
 */
@Deprecated
public static <K, V> ObjectMapConf<Map<K, V>> objMap(String property, Map<K, V> map, Class<K> keyType,
Class<V> valueType, String nameSpace) {
    return new ObjectMapConf<Map<K, V>>(property, map, keyType, valueType).setNameSpace(nameSpace);
}

public static <K, V> ObjectMapConf<Map<K, V>> objMap(Map<K, V> map, Class<K> keyType, Class<V> valueType) {
    return new ObjectMapConf<Map<K, V>>(map, keyType, valueType);
}

public static <K, V> ObjectMapConf<Map<K, V>> objMap(Map<K, V> map, Class<K> keyType, Class<V> valueType,
boolean ordered) {
    return new ObjectMapConf<Map<K, V>>(map, keyType, valueType, ordered);
}
}

```

1.StringJDK Conf<T> xxx = Holders.simple(t);

2.MapMapkeyvalueStringJDK MapConf<K,V> xx = Holdes.map(map,keyType,valueType); keyTypeValueTypeMapConfmapgetputremoveclearAB
MapABMapConfB A.getB.putremove

3.StringJDKColConf<Collection<K>> xxxx = Holders.collection(collection,valueType);

4.Conf<T> tt = Holders.obj(t,type), Conf

5. ObjectColConf<T> ttt = Holders.objCollection();ObjectColConf addremoveclearCollectionObjectColConf

6.MapkeyStringJDKvalueMapConf<Map<K.V>> kkk = Holders.map(map,keytype,valueType); keytypevalueType

UniqueKey.

2-

xmlableholderXmlablereadXmlwriteXml

```

package com.fr.config.holder.factory;

import com.fr.config.holder.Conf;
import com.fr.config.holder.impl.xml.XmlColConf;
import com.fr.config.holder.impl.xml.XmlConf;
import com.fr.config.holder.impl.xml.XmlMapConf;
import com.fr.stable.xml.XMLable;

import java.util.Collection;
import java.util.Map;

/**
 * http://www.finedevelop.com/pages/viewpage.action?pageId=18648778
 * xml
 * xml
 */
@Deprecated
public class XmlHolders {
    //XmlablereadxmlwriteXml
    // config,property arr XXXXXX xxxxmlable
    @Deprecated
    @SuppressWarnings("unchecked")
    public static <T extends XMLable> Conf<T> obj(String property, T t, Class<T> clazz, String nameSpace) {
        return new XmlConf(property, t, clazz).setNameSpace(nameSpace);
    }

    @SuppressWarnings("unchecked")
    public static <T extends XMLable> Conf<T> obj(T t, Class<T> clazz) {
        return new XmlConf(t, clazz);
    }

    //XmlablereadxmlwriteXml
    // config,property arr id
    // config.arr.id XXXXXX xxxxmlable
    @Deprecated
    public static <T extends XMLable> XmlColConf<Collection<T>> collection(String property, Collection<T> t,
Class<T> clazz, String nameSpace) {
        return new XmlColConf<Collection<T>>(property, t, clazz).setNameSpace(nameSpace);
    }

    public static <T extends XMLable> XmlColConf<Collection<T>> collection(Collection<T> t, Class<T> clazz) {
        return new XmlColConf<Collection<T>>(t, clazz);
    }

    //XmllableMapreadxmlwriteXml
    /// config,property arr
    // config.arr.key XXXXXX xxxxmlable
    //K KKString,KValueWriterKString
    // KValueReaderStringK
    @Deprecated
    public static <K, V extends XMLable> XmlMapConf<Map<K, V>> map(String property, Map<K, V> t, Class keyType,
Class<V> clazz, String nameSpace) {
        return new XmlMapConf<Map<K, V>>(property, t, keyType, clazz).setNameSpace(nameSpace);
    }

    public static <K, V extends XMLable> XmlMapConf<Map<K, V>> map(Map<K, V> t, Class keyType, Class<V> clazz) {
        return new XmlMapConf<Map<K, V>>(t, keyType, clazz);
    }
}

```

holderMapkeyString KKString,KValueWriterKStringValueWriter.registerWriter KValueReaderStringKValueReader.registerReader