加载持久层的Mapper.xml配置文件：

1、定义文件的位置，此设置在spring容器中，让容器去加载

@Bean

public SqlSessionFactory sqlSessionFactoryBean() throws Exception {

SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();

sqlSessionFactoryBean.setDataSource(dataSource());

PathMatchingResourcePatternResolver resolver = new PathMatchingResourcePatternResolver();

sqlSessionFactoryBean.setMapperLocations(resolver.getResources("classpath:/mybatis/\*.xml"));

return sqlSessionFactoryBean.getObject();

}

2、spring-core架包PathMatchingResourcePatternResolver实现加载

@Override

public Resource[] getResources(String locationPattern) throws IOException {

Assert.notNull(locationPattern, "Location pattern must not be null");

if (locationPattern.startsWith(CLASSPATH\_ALL\_URL\_PREFIX)) {

// a class path resource (multiple resources for same name possible)

if (getPathMatcher().isPattern(locationPattern.substring(CLASSPATH\_ALL\_URL\_PREFIX.length()))) {

// a class path resource pattern

return findPathMatchingResources(locationPattern);

}

else {

// all class path resources with the given name

return findAllClassPathResources(locationPattern.substring(CLASSPATH\_ALL\_URL\_PREFIX.length()));

}

}

else {

// Only look for a pattern after a prefix here

// (to not get fooled by a pattern symbol in a strange prefix).

int prefixEnd = locationPattern.indexOf(":") + 1;

if (getPathMatcher().isPattern(locationPattern.substring(prefixEnd))) {

// a file pattern

return findPathMatchingResources(locationPattern);

}

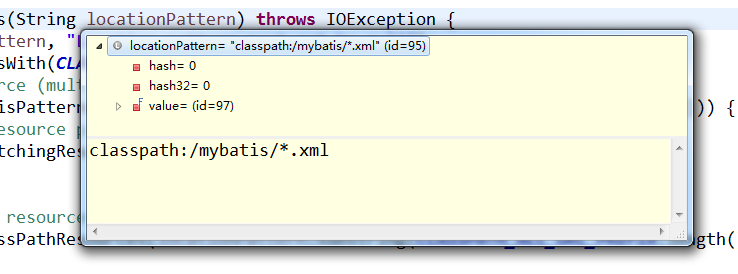
else {

// a single resource with the given name

return new Resource[] {getResourceLoader().getResource(locationPattern)};

}

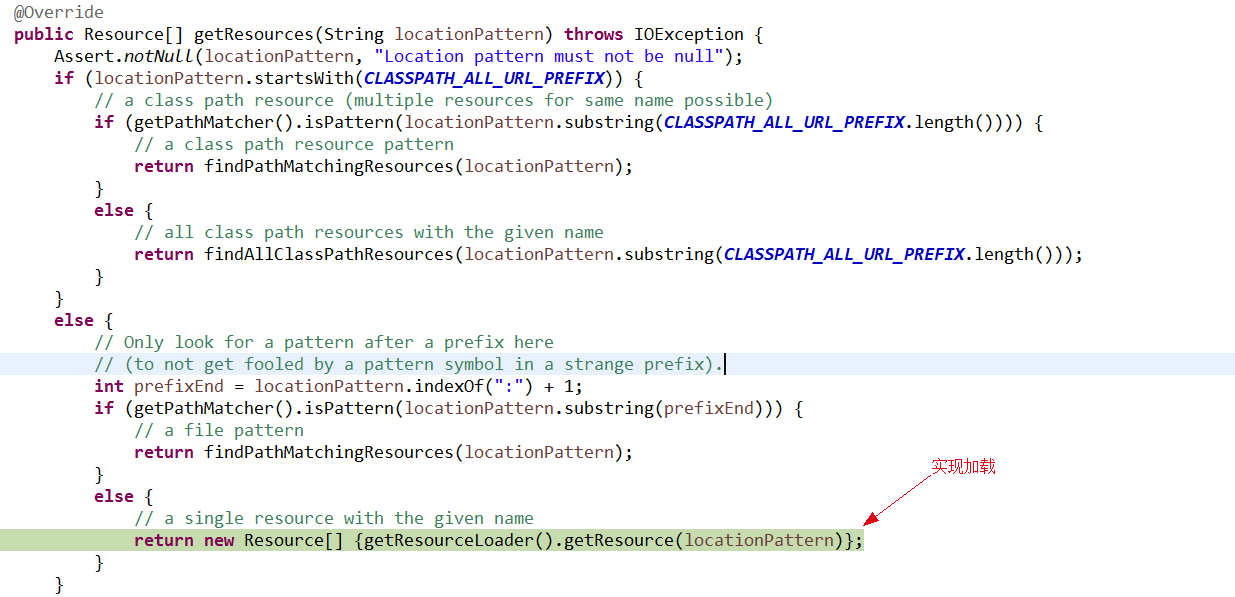
}

} 

3、加载资源



过程

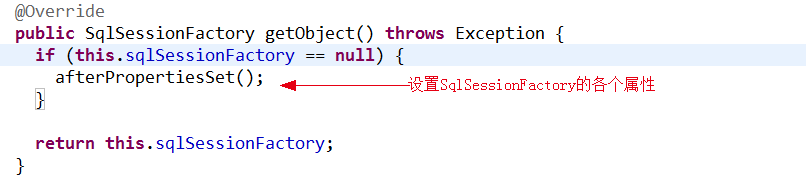


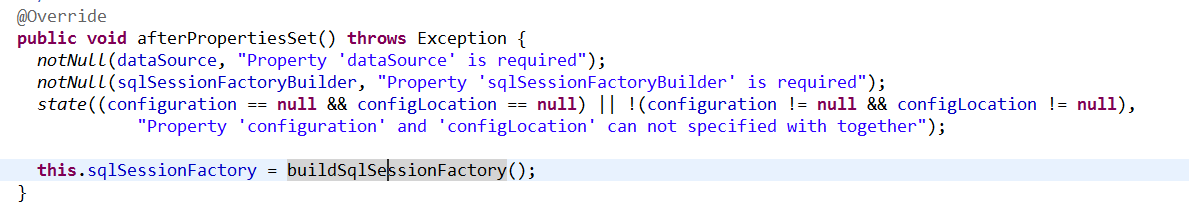
加载完以后存放在：Set<Resource> result = new LinkedHashSet<Resource>(16);

[file [E:\myspace\mybatis\ssm\bin\mybatis\TestMapper.xml], file [E:\myspace\mybatis\ssm\bin\mybatis\UserMapper.xml]]

并将结果存放到SqlSessionFactoryBean的Resource[] mapperLocation数组中

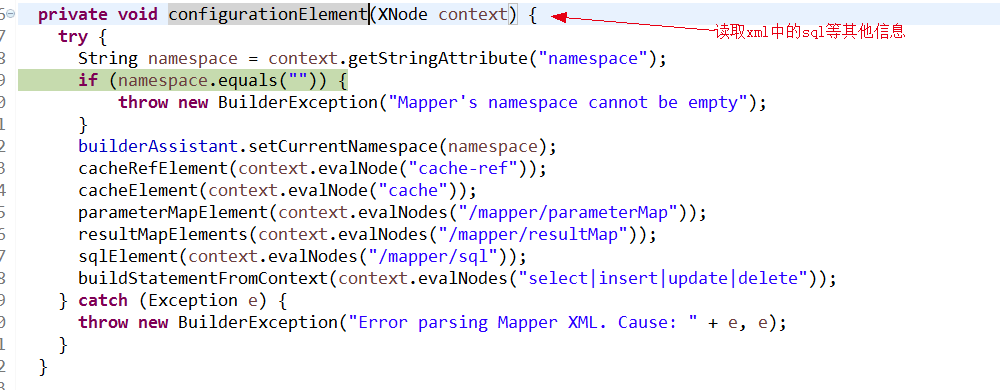
4、创建SqlSessionFactory:sqlSessionFactoryBean.getObject()，即第1步配置





其中buildSqlSessionFactory()方法就是实现将Mapper.xml配置文件加载到内存





其中入参XNode context就是整个xml的内容。



保存xml中sql的所有信息

public void parseStatementNode() {

String id = context.getStringAttribute("id");

String databaseId = context.getStringAttribute("databaseId");

if (!databaseIdMatchesCurrent(id, databaseId, this.requiredDatabaseId)) return;

Integer fetchSize = context.getIntAttribute("fetchSize");

Integer timeout = context.getIntAttribute("timeout");

String parameterMap = context.getStringAttribute("parameterMap");

String parameterType = context.getStringAttribute("parameterType");

Class<?> parameterTypeClass = resolveClass(parameterType);

String resultMap = context.getStringAttribute("resultMap");

String resultType = context.getStringAttribute("resultType");

String lang = context.getStringAttribute("lang");

LanguageDriver langDriver = getLanguageDriver(lang);

Class<?> resultTypeClass = resolveClass(resultType);

String resultSetType = context.getStringAttribute("resultSetType");

StatementType statementType = StatementType.valueOf(context.getStringAttribute("statementType", StatementType.PREPARED.toString()));

ResultSetType resultSetTypeEnum = resolveResultSetType(resultSetType);

String nodeName = context.getNode().getNodeName();

SqlCommandType sqlCommandType = SqlCommandType.valueOf(nodeName.toUpperCase(Locale.ENGLISH));

boolean isSelect = sqlCommandType == SqlCommandType.SELECT;

boolean flushCache = context.getBooleanAttribute("flushCache", !isSelect);

boolean useCache = context.getBooleanAttribute("useCache", isSelect);

boolean resultOrdered = context.getBooleanAttribute("resultOrdered", false);

// Include Fragments before parsing

XMLIncludeTransformer includeParser = new XMLIncludeTransformer(configuration, builderAssistant);

includeParser.applyIncludes(context.getNode());

// Parse selectKey after includes and remove them.

processSelectKeyNodes(id, parameterTypeClass, langDriver);

// Parse the SQL (pre: <selectKey> and <include> were parsed and removed)

SqlSource sqlSource = langDriver.createSqlSource(configuration, context, parameterTypeClass);

String resultSets = context.getStringAttribute("resultSets");

String keyProperty = context.getStringAttribute("keyProperty");

String keyColumn = context.getStringAttribute("keyColumn");

KeyGenerator keyGenerator;

String keyStatementId = id + SelectKeyGenerator.SELECT\_KEY\_SUFFIX;

keyStatementId = builderAssistant.applyCurrentNamespace(keyStatementId, true);

if (configuration.hasKeyGenerator(keyStatementId)) {

keyGenerator = configuration.getKeyGenerator(keyStatementId);

} else {

keyGenerator = context.getBooleanAttribute("useGeneratedKeys",

configuration.isUseGeneratedKeys() && SqlCommandType.INSERT.equals(sqlCommandType))

? new Jdbc3KeyGenerator() : new NoKeyGenerator();

}

builderAssistant.addMappedStatement(id, sqlSource, statementType, sqlCommandType,

fetchSize, timeout, parameterMap, parameterTypeClass, resultMap, resultTypeClass,

resultSetTypeEnum, flushCache, useCache, resultOrdered,

keyGenerator, keyProperty, keyColumn, databaseId, langDriver, resultSets);

}

最后所有对象全部存放到MappedStatement对象中（在MapperBuilderAssistant中完成）

