# Mybatis-config.xml

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
  
<configuration>  
  
 <!-- autoMappingBehavior should be set in each test case -->  
  
 <environments default="development">  
 <environment id="development">  
 <transactionManager type="JDBC">  
 <property name="" value=""/>  
 </transactionManager>  
 <dataSource type="UNPOOLED">  
 <property name="driver" value="org.hsqldb.jdbcDriver"/>  
 <property name="url" value="jdbc:hsqldb:mem:automapping"/>  
 <property name="username" value="sa"/>  
 </dataSource>  
 </environment>  
 </environments>  
  
 <mappers>  
 <mapper resource="org/apache/ibatis/autoconstructor/AutoConstructorMapper.xml"/>  
 </mappers>  
  
</configuration>

# AutoConstructorMapper.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
  
<mapper namespace="org.apache.ibatis.autoconstructor.AutoConstructorMapper">  
</mapper>

# org.apache.ibatis.autoconstructor.AutoConstructorMapper

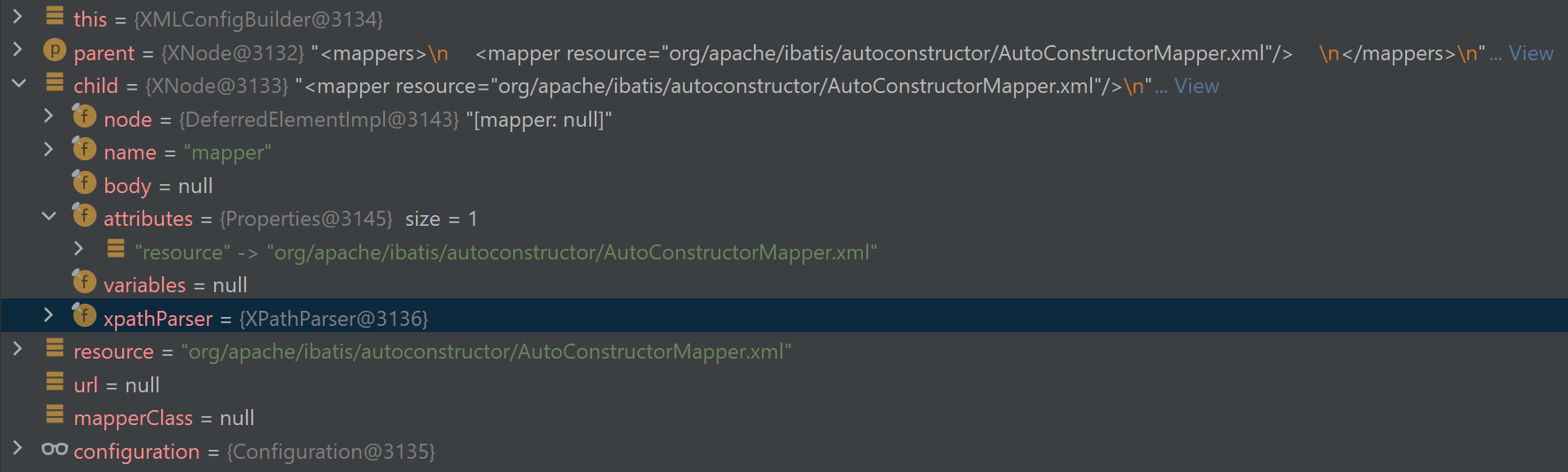
public interface AutoConstructorMapper {  
 @Select("SELECT \* FROM subject WHERE id = #{id}")  
 PrimitiveSubject getSubject(final int id);  
  
 @Select("SELECT \* FROM subject")  
 List<PrimitiveSubject> getSubjects();  
  
 @Select("SELECT \* FROM subject")  
 List<AnnotatedSubject> getAnnotatedSubjects();  
  
 @Select("SELECT \* FROM subject")  
 List<BadSubject> getBadSubjects();  
  
 @Select("SELECT \* FROM extensive\_subject")  
 List<ExtensiveSubject> getExtensiveSubjects();  
}

# #1: org.apache.ibatis.builder.xml.XMLConfigBuilder#parseConfiguration

private void parseConfiguration(XNode root) {  
 try {  
 // issue #117 read properties first  
 propertiesElement(root.evalNode("properties"));  
 Properties settings = settingsAsProperties(root.evalNode("settings"));  
 loadCustomVfs(settings);  
 loadCustomLogImpl(settings);  
 typeAliasesElement(root.evalNode("typeAliases"));  
 pluginElement(root.evalNode("plugins"));  
 objectFactoryElement(root.evalNode("objectFactory"));  
 objectWrapperFactoryElement(root.evalNode("objectWrapperFactory"));  
 reflectorFactoryElement(root.evalNode("reflectorFactory"));  
 settingsElement(settings);  
 // read it after objectFactory and objectWrapperFactory issue #631  
 environmentsElement(root.evalNode("environments"));  
 databaseIdProviderElement(root.evalNode("databaseIdProvider"));  
 typeHandlerElement(root.evalNode("typeHandlers"));  
 mapperElement(root.evalNode("mappers")); <-##1  
 } catch (Exception e) {  
 throw new BuilderException("Error parsing SQL Mapper Configuration. Cause: " + e, e);  
 }  
}

# ##1: org.apache.ibatis.builder.xml.XMLConfigBuilder#mapperElement

private void mapperElement(XNode parent) throws Exception {  
 if (parent != null) {  
 for (XNode child : parent.getChildren()) {  
 if ("package".equals(child.getName())) {  
 String mapperPackage = child.getStringAttribute("name");  
 configuration.addMappers(mapperPackage);  
 } else {  
 String resource = child.getStringAttribute("resource");  
 String url = child.getStringAttribute("url");  
 String mapperClass = child.getStringAttribute("class");  
 if (resource != null && url == null && mapperClass == null) {  
 ErrorContext.*instance*().resource(resource); <-###1,###2  
 try(InputStream inputStream = Resources.*getResourceAsStream*(resource)) {  
 XMLMapperBuilder mapperParser = new XMLMapperBuilder(inputStream, configuration, resource, configuration.getSqlFragments() <-###3); <-###4  
 mapperParser.parse(); <-###5  
 }  
 } else if (resource == null && url != null && mapperClass == null) {  
 ErrorContext.*instance*().resource(url);  
 try(InputStream inputStream = Resources.*getUrlAsStream*(url)){  
 XMLMapperBuilder mapperParser = new XMLMapperBuilder(inputStream, configuration, url, configuration.getSqlFragments());  
 mapperParser.parse();  
 }  
 } else if (resource == null && url == null && mapperClass != null) {  
 Class<?> mapperInterface = Resources.*classForName*(mapperClass);  
 configuration.addMapper(mapperInterface);  
 } else {  
 throw new BuilderException("A mapper element may only specify a url, resource or class, but not more than one.");  
 }  
 }  
 }  
 }  
}



# ###1: org.apache.ibatis.executor.ErrorContext#instance

private static final ThreadLocal<ErrorContext> *LOCAL* = ThreadLocal.*withInitial*(ErrorContext::new);

private ErrorContext() {  
}

public static ErrorContext instance() {  
 return *LOCAL*.get();  
}

# ###2: org.apache.ibatis.executor.ErrorContext#resource

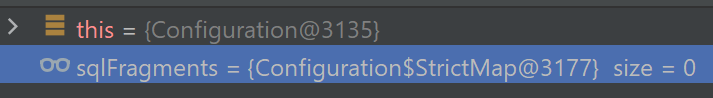
private String resource;

public ErrorContext resource(String resource) {  
 this.resource = resource;  
 return this;  
}

# ###3: org.apache.ibatis.session.Configuration#getSqlFragments

protected final Map<String, XNode> sqlFragments = new StrictMap<>("XML fragments parsed from previous mappers"); <-####1

public Map<String, XNode> getSqlFragments() {  
 return sqlFragments;  
}



# ####1: org.apache.ibatis.session.Configuration.StrictMap#StrictMap(java.lang.String)

protected static class StrictMap<V> extends HashMap<String, V>

private final String name;

public StrictMap(String name) {  
 super();  
 this.name = name;  
}

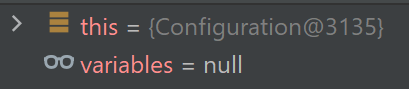
# ###4: org.apache.ibatis.builder.xml.XMLMapperBuilder#XMLMapperBuilder(java.io.InputStream, org.apache.ibatis.session.Configuration, java.lang.String, java.util.Map<java.lang.String,org.apache.ibatis.parsing.XNode>)

public XMLMapperBuilder(InputStream inputStream, Configuration configuration, String resource, Map<String, XNode> sqlFragments) {  
 this(new XPathParser(inputStream, true, configuration.getVariables() <-####1, new XMLMapperEntityResolver() <-####2) <-####3,  
 configuration, resource, sqlFragments); <-####4  
}

# ####1: org.apache.ibatis.session.Configuration#getVariables

protected Properties variables = new Properties();

public Properties getVariables() {  
 return variables;  
}



# ####2: org.apache.ibatis.builder.xml.XMLMapperEntityResolver

public class XMLMapperEntityResolver implements EntityResolver

private static final String *IBATIS\_CONFIG\_SYSTEM* = "ibatis-3-config.dtd";  
private static final String *IBATIS\_MAPPER\_SYSTEM* = "ibatis-3-mapper.dtd";  
private static final String *MYBATIS\_CONFIG\_SYSTEM* = "mybatis-3-config.dtd";  
private static final String *MYBATIS\_MAPPER\_SYSTEM* = "mybatis-3-mapper.dtd";  
  
private static final String *MYBATIS\_CONFIG\_DTD* = "org/apache/ibatis/builder/xml/mybatis-3-config.dtd";  
private static final String *MYBATIS\_MAPPER\_DTD* = "org/apache/ibatis/builder/xml/mybatis-3-mapper.dtd";

# ####3: org.apache.ibatis.parsing.XpathParser

public XPathParser(InputStream inputStream, boolean validation, Properties variables, EntityResolver entityResolver) {  
 commonConstructor(validation, variables, entityResolver); <-#####1  
 this.document = createDocument(new InputSource(inputStream)); <-#####2  
}