java-mybatis-mapper-proxy

简单实例

```
public interface BlogMapper {
    Blog selectBlog(long id);
}



@Test
public void test() throws IOException {
    String resource = "mybatis-config.xml";
    InputStream inputStream = Resources.getResourceAsStream(resource);
    SqlSessionFactory sqlSessionFactory  = new SqlSessionFactoryBuilder().build(inputStream);

    SqlSession session = sqlSessionFactory.openSession();
    try {
        BlogMapper mapper = session.getMapper(BlogMapper.class);
        Blog blog = mapper.selectBlog(1);

        System.out.println(blog);
    } finally {
        session.close();
    }
}
```

我们都知道，BlogMapper生成的代理类，调用mapper.selectBlog方法将执行数据库查询，将返回查询结果。

mapper代理实现

跟踪 `session.getMapper` 方法，最终调用到 `DefaultSqlSession.getMapper` 方法

```
  public <T> T getMapper(Class<T> type) {
    return configuration.<T>getMapper(type, this);
  }
```

`Configuration.getMapper` 内容如下

```
  public <T> T getMapper(Class<T> type, SqlSession sqlSession) {
    return mapperRegistry.getMapper(type, sqlSession);
  }
```

可以看到这里将调用mapperRegistry.getMapper生成代理类。

mapperRegistry初始化

这里先看一下mapperRegistry的初始化过程 `SqlSessionFactoryBuilder().build(inputStream);` 构造初始环境，这里会调用到 `XMLConfigBuilder.mapperElement` 方法：

```
  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);
            InputStream inputStream = Resources.getResourceAsStream(resource);
            XMLMapperBuilder mapperParser = new XMLMapperBuilder(inputStream, configuration, resource, conf
iguration.getSqlFragments());
            mapperParser.parse();
          } else if (resource == null && url != null && mapperClass == null) {
            ErrorContext.instance().resource(url);
            InputStream inputStream = Resources.getUrlAsStream(url);
            XMLMapperBuilder mapperParser = new XMLMapperBuilder(inputStream, configuration, url, configura
tion.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.");
          }
        }
      }
    }
  }
```

这里会对Mapper.xml文件进行解析，解析结果最终会调用configuration.addMapper/configuration.addMapper方法，这些方法会调用mapperRegistry.addMappers/mapperRegistry.addMapper对mapperRegistry初始化。

`MapperRegistry.addMapper` 内容如下

```
public <T> void addMapper(Class<T> type) {
    ...
    knownMappers.put(type, new MapperProxyFactory<T>(type));
}
```

mapperRegistry生成代理

`mapperRegistry.getMapper` 内容如下

```
public <T> T getMapper(Class<T> type, SqlSession sqlSession) {
final MapperProxyFactory<T> mapperProxyFactory = (MapperProxyFactory<T>) knownMappers.get(type);
    ...
    return mapperProxyFactory.newInstance(sqlSession);
}
```

`MapperProxyFactory.newInstance` 内容如下

```
  protected T newInstance(MapperProxy<T> mapperProxy) {
    return (T) Proxy.newProxyInstance(mapperInterface.getClassLoader(), new Class[] { mapperInterface }, ma
pperProxy);
  }

  public T newInstance(SqlSession sqlSession) {
    final MapperProxy<T> mapperProxy = new MapperProxy<T>(sqlSession, mapperInterface, methodCache);
    return newInstance(mapperProxy);
  }
```

`MapperProxy` 即为动态代理类，实现了InvocationHandler接口

```
public class MapperProxy<T> implements InvocationHandler, Serializable {
    ...
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    if (Object.class.equals(method.getDeclaringClass())) {  //
      try {
        return method.invoke(this, args);
      } catch (Throwable t) {
        throw ExceptionUtil.unwrapThrowable(t);
      }
    }
    final MapperMethod mapperMethod = cachedMapperMethod(method);
    return mapperMethod.execute(sqlSession, args);
    }
}
```