**从 XML 中构建 SqlSessionFactory**

每个基于 MyBatis 的应用都是以一个 SqlSessionFactory 的实例为中心的。SqlSessionFactory 的实例可以通过 SqlSessionFactoryBuilder 获得。而 SqlSessionFactoryBuilder 则可以从 XML 配置文件或一个预先定制的 Configuration 的实例构建出 SqlSessionFactory 的实例。

从 XML 文件中构建 SqlSessionFactory 的实例非常简单，建议使用类路径下的资源文件进行配置。但是也可以使用任意的输入流(InputStream)实例，包括字符串形式的文件路径或者 file:// 的 URL 形式的文件路径来配置。MyBatis 包含一个名叫 Resources 的工具类，它包含一些实用方法，可使从 classpath 或其他位置加载资源文件更加容易。

String resource = "org/mybatis/example/mybatis-config.xml";

InputStream inputStream = Resources.getResourceAsStream(resource);

sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

XML 配置文件（configuration XML）中包含了对 MyBatis 系统的核心设置，包含获取数据库连接实例的数据源（DataSource）和决定事务范围和控制方式的事务管理器 （TransactionManager）。XML 配置文件的详细内容后面再探讨，这里先给出一个简单的示例：

<?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>

<environments default="development">

<environment id="development">

<transactionManager type="JDBC"/>

<dataSource type="POOLED">

<property name="driver" value="${driver}"/>

<property name="url" value="${url}"/>

<property name="username" value="${username}"/>

<property name="password" value="${password}"/>

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource="org/mybatis/example/BlogMapper.xml"/>

</mappers>

</configuration>

当然，还有很多可以在XML 文件中进行配置，上面的示例指出的则是最关键的部分。要注意 XML 头部的声明，用来验证 XML 文档正确性。environment 元素体中包含了事务管理和连接池的配置。mappers 元素则是包含一组 mapper 映射器（这些 mapper 的 XML 文件包含了 SQL 代码和映射定义信息）。

**不使用 XML 构建 SqlSessionFactory**

如果你更愿意直接从 Java 程序而不是 XML 文件中创建 configuration，或者创建你自己的 configuration 构建器，MyBatis 也提供了完整的配置类，提供所有和 XML 文件相同功能的配置项。

DataSource dataSource = BlogDataSourceFactory.getBlogDataSource();

TransactionFactory transactionFactory = new JdbcTransactionFactory();

Environment environment = new Environment("development", transactionFactory, dataSource);

Configuration configuration = new Configuration(environment);

configuration.addMapper(BlogMapper.class);

SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(configuration);

注意该例中，configuration 添加了一个映射器类（mapper class）。映射器类是 Java 类，它们包含 SQL 映射语句的注解从而避免了 XML 文件的依赖。不过，由于 Java 注解的一些限制加之某些 MyBatis 映射的复杂性，XML 映射对于大多数高级映射（比如：嵌套 Join 映射）来说仍然是必须的。有鉴于此，如果存在一个对等的 XML 配置文件的话，MyBatis 会自动查找并加载它（这种情况下， BlogMapper.xml 将会基于类路径和 BlogMapper.class 的类名被加载进来）。具体细节稍后讨论。

**从 SqlSessionFactory 中获取 SqlSession**

既然有了 SqlSessionFactory ，顾名思义，我们就可以从中获得 SqlSession 的实例了。SqlSession 完全包含了面向数据库执行 SQL 命令所需的所有方法。你可以通过 SqlSession 实例来直接执行已映射的 SQL 语句。例如：

SqlSession session = sqlSessionFactory.openSession();

try {

Blog blog = (Blog) session.selectOne("org.mybatis.example.BlogMapper.selectBlog", 101);

} finally {

session.close();

}

诚然这种方式能够正常工作，并且对于使用旧版本 MyBatis 的用户来说也比较熟悉，不过现在有了一种更直白的方式。使用对于给定语句能够合理描述参数和返回值的接口（比如说BlogMapper.class），你 现在不但可以执行更清晰和类型安全的代码，而且还不用担心易错的字符串字面值以及强制类型转换。

例如：

SqlSession session = sqlSessionFactory.openSession();

try {

BlogMapper mapper = session.getMapper(BlogMapper.class);

Blog blog = mapper.selectBlog(101);

} finally {

session.close();

}