自定义持久层框架的代码实现一

项目结构

```
.
├── IPersistence
│   ├── IPersistence.iml
│   ├── pom.xml
│   └── src
└── IPersistence_test
    ├── IPersistence_test.iml
    ├── pom.xml
    ├── src
    └── target
```

具体代码

sqlMapperConfig.xml配置文件

UserMapper.xml配置文件

```
      select * from user

      select * from user where id = #{id} and username = #{username}
```

读取资源处理，Resources类

```
/**
 * 资源处理类
 *
 * @name: Resource
 * @author: terwer
 * @date: 2022-05-08 15:57
 */
object Resources {
    /**
     * 根据配置文件的路径，将配置文件加载成字节输入流，存储到内存中
     *
     * @param path
     * @return
     */
    @JvmStatic
    fun getResourceAsStream(path: String?): InputStream {
        return Resources::class.java.classLoader.getResourceAsStream(path)
    }
}

/**
 * 资源处理类
 *
 * @name: Resource
 * @author: terwer
 * @date: 2022-03-14 12:57
 **/
public class Resources {
    /**
     * 根据配置文件的路径，将配置文件加载成字节输入流，存储到内存中
     *
     * @param path
     * @return
     */
    public static InputStream getResourceAsStream(String path) {
        InputStream inputStream = Resources.class.getClassLoader().getResourceAsStream(path)

        return inputStream;
    }
}
```

SqlSessionFactoryBuider工厂构建对象

```
/**
 * 工厂构建对象
 *
 * @name: SqlSessionFactoryBuilder
 * @author: terwer
 * @date: 2022-05-08 15:18
 */
class SqlSessionFactoryBuilder {
    @Throws(DocumentException::class, PropertyVetoException::class)
    fun build(ips: InputStream?): SqlSessionFactory {
        // 1、解析配置文件，将解析出来的内容封装到Configuration中
        val xmlConfigBuilder = XmlConfigBuilder()
        val configuration = xmlConfigBuilder.parse(ips)

        // 2、创建SqlSessionFactory对象
        return DefaultSqlSessionFactory(configuration)
    }
}

/**
 * 工厂构建对象
 *
 * @name: SqlSessionFactoryBuilder
 * @author: terwer
 * @date: 2022-03-14 15:18
 **/
public class SqlSessionFactoryBuilder {
    public SqlSessionFactory build(InputStream in) throws DocumentException, PropertyVetoE
ception {
        // 1、解析配置文件，将解析出来的内容封装到Configuration中
        XmlConfigBuilder xmlConfigBuilder = new XmlConfigBuilder();
        Configuration configuration = xmlConfigBuilder.parse(in);

        // 2、创建SqlSessionFactory对象
        DefaultSqlSessionFactory sqlSessionFactory = new DefaultSqlSessionFactory(configurati
n);
        return sqlSessionFactory;
    }

}
```

配置文件解析

```
/**
 * @name: XmlConfigBuilder
 * @author: terwer
 * @date: 2022-03-14 15:40
 */
class XmlConfigBuilder {
    private val configuration: Configuration

    init {
        configuration = Configuration()
    }

    /**
     * 使用dom4j将配置文件进行解析，封装Configuration
     *
     * @param in
     * @return
     */
    @Throws(DocumentException::class, PropertyVetoException::class)
    fun parse(ips: InputStream?): Configuration {
        val document = SAXReader().read(ips)
        //
        val rootElement = document.rootElement
        val list: List = rootElement.selectNodes("//property")
        val properties = Properties()
        for (element in list) {
            val name = element!!.attributeValue("name")
            val value = element.attributeValue("value")
            properties.setProperty(name, value)
        }
        val comboPooledDataSource = ComboPooledDataSource()
        comboPooledDataSource.driverClass = properties.getProperty("driverClass")
        comboPooledDataSource.jdbcUrl = properties.getProperty("jdbcUrl")
        comboPooledDataSource.user = properties.getProperty("username")
        comboPooledDataSource.password = properties.getProperty("password")
        configuration.dataSource = comboPooledDataSource

        // mapper.xml解析，拿到路径，加载成字节输入流，进行解析
        val mapperList: List = rootElement.selectNodes("//mapper")
        //
        for (element in mapperList) {
            val mapperPath = element!!.attributeValue("resource")
            val resourceAsStream = Resources.getResourceAsStream(mapperPath)
            val xmlMapperBuilder = XmlMapperBuilder(configuration)
            xmlMapperBuilder.parse(resourceAsStream)
        }
        return configuration
    }
}

/**
 * @name: XmlConfigBuilder
 * @author: terwer
 * @date: 2022-03-14 15:40
 **/
public class XmlConfigBuilder {

    private Configuration configuration;

    public XmlConfigBuilder() {
        configuration = new Configuration();
    }

    /**
     * 使用dom4j将配置文件进行解析，封装Configuration
     *
     * @param in
     * @return
     */
    public Configuration parse(InputStream in) throws DocumentException, PropertyVetoExcep
ion {
        Document document = new SAXReader().read(in);
        //
        Element rootElement = document.getRootElement();

        List list = rootElement.selectNodes("//property");
        Properties properties = new Properties();
        for (Element element : list) {
            String name = element.attributeValue("name");
            String value = element.attributeValue("value");

            properties.setProperty(name, value);
        }

        ComboPooledDataSource comboPooledDataSource = new ComboPooledDataSource();
        comboPooledDataSource.setDriverClass(properties.getProperty("driverClass"));
        comboPooledDataSource.setJdbcUrl(properties.getProperty("jdbcUrl"));
        comboPooledDataSource.setUser(properties.getProperty("username"));
        comboPooledDataSource.setPassword(properties.getProperty("password"));

        configuration.setDataSource(comboPooledDataSource);

        // mapper.xml解析，拿到路径，加载成字节输入流，进行解析
        List mapperList= rootElement.selectNodes("//mapper");
        //
        for (Element element : mapperList) {
            String mapperPath = element.attributeValue("resource");
            InputStream resourceAsStream = Resources.getResourceAsStream(mapperPath);

            XmlMapperBuilder xmlMapperBuilder = new XmlMapperBuilder(configuration);
            xmlMapperBuilder.parse(resourceAsStream);
        }

        return configuration;
    }
}
```

mapper映射文件解析

```
/**
 * mapper解析器
 *
 * @name: XmlMapperBuilder
 * @author: terwer
 * @date: 2022-05-08 16:16
 */
class XmlMapperBuilder(private val configuration: Configuration) {
    @Throws(DocumentException::class)
    fun parse(`in`: InputStream?) {
        val document = SAXReader().read(`in`)

        //
        val rootElement = document.rootElement
        val namespace = rootElement.attributeValue("namespace")
        val list: List = rootElement.selectNodes("//select")
        for (element in list) {
            val id = element!!.attributeValue("id")
            val resultType = element.attributeValue("resultType")
            val parameterType = element.attributeValue("parameterType")
            val sqlText = element.textTrim
            val mappedStatement = MappedStatement()
            mappedStatement.statementId = id
            mappedStatement.resultType = resultType
            mappedStatement.parameterType = parameterType
            mappedStatement.sql = sqlText
            val mappedStatementMap = configuration.mappedStatementMap
            val statementId = "$namespace.$id"
            mappedStatementMap[statementId] = mappedStatement
        }
    }
}

/**
 * mapper解析器
 *
 * @name: XmlMapperBuilder
 * @author: terwer
 * @date: 2022-03-14 16:16
 **/
public class XmlMapperBuilder {
    private Configuration configuration;

    public XmlMapperBuilder(Configuration configuration) {
        this.configuration = configuration;
    }

    public void parse(InputStream in) throws DocumentException {
        Document document = new SAXReader().read(in);

        //
        Element rootElement = document.getRootElement();
        String namespace = rootElement.attributeValue("namespace");

        List list = rootElement.selectNodes("//select");
        for (Element element : list) {
            String id = element.attributeValue("id");
            String resultType = element.attributeValue("resultType");
            String parameterType = element.attributeValue("parameterType");

            String sqlText = element.getTextTrim();

            MappedStatement mappedStatement = new MappedStatement();
            mappedStatement.setStatementId(id);
            mappedStatement.setResultType(resultType);
            mappedStatement.setParameterType(parameterType);
            mappedStatement.setSql(sqlText);

            Map mappedStatementMap = configuration.getMappedStatementMap();
            String statementId = namespace + "." + id;
            mappedStatementMap.put(statementId, mappedStatement);
        }
    }
}
```

SqlSession的具体实现

```
/**
 * @name: SqlSession
 * @author: terwer
 * @date: 2022-05-08 16:35
 */
interface SqlSession {
    /**
     * 查询所有
     */
    @Throws(Exception::class)
    fun  selectList(statementId: String?, vararg params: Any?): List?

    /**
     * 查询单个
     */
    @Throws(Exception::class)
    fun  selectOne(statementId: String?, vararg params: Any?): T
    fun  getMapper(mapperClass: Class<*>?): T
}

/**
 * @name: SqlSession
 * @author: terwer
 * @date: 2022-03-14 16:35
 **/
public interface SqlSession {
    /**
     * 查询所有
     */
    public  List selectList(String statementId, Object... params) throws Exception;

    /**
     * 查询单个
     */
    public  T selectOne(String statementId, Object... params) throws Exception;

    public  T getMapper(Class mapperClass);
}
```

文章更新历史

2022/05/08 feat:增加Kotlin实现。