[**第五篇：springboot整合 beatlsql**](http://blog.csdn.net/forezp/article/details/70662983)

参考：<http://blog.csdn.net/forezp/article/details/70662983>

转载请标明出处：   
<http://blog.csdn.net/forezp/article/details/70662983>   
本文出自[方志朋的博客](http://blog.csdn.net/forezp)

BeetSql是一个全功能DAO工具， 同时具有Hibernate 优点 & Mybatis优点功能，适用于承认以SQL为中心，同时又需求工具能自动生成大量常用的SQL的应用。

beatlsql 优点

* 开发效率
  + 无需注解，自动使用大量内置SQL，轻易完成增删改查功能，节省50%的开发工作量
  + 数据模型支持Pojo，也支持Map/List这种快速模型，也支持混合模型
  + SQL 模板基于Beetl实现，更容易写和调试，以及扩展
* 维护性
  + SQL 以更简洁的方式，Markdown方式集中管理，同时方便程序开发和数据库SQL调试。
  + 可以自动将sql文件映射为dao接口类
  + 灵活直观的支持一对一，一对多，多对多关系映射而不引入复杂的OR Mapping概念和技术。
  + 具备Interceptor功能，可以调试，性能诊断SQL，以及扩展其他功能
* 其他
  + 内置支持主从数据库支持的开源工具
  + 支持跨数据库平台，开发者所需工作减少到最小，目前跨数据库支持mysql,postgres,oracle,sqlserver,h2,sqllite,DB2.

引入依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.ibeetl</groupId>

<artifactId>beetl</artifactId>

<version>2.3.2</version>

</dependency>

<dependency>

<groupId>com.ibeetl</groupId>

<artifactId>beetlsql</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.0.5</version>

</dependency>

这几个依赖都是必须的。

整合阶段

由于springboot没有对 beatlsql的快速启动装配，所以需要我自己导入相关的bean，包括数据源，包扫描，事物管理器等。

在application加入以下代码：

@Bean(initMethod = "init", name = "beetlConfig")

public BeetlGroupUtilConfiguration getBeetlGroupUtilConfiguration() {

BeetlGroupUtilConfiguration beetlGroupUtilConfiguration = new BeetlGroupUtilConfiguration();

ResourcePatternResolver patternResolver = ResourcePatternUtils.getResourcePatternResolver(new DefaultResourceLoader());

try {

// WebAppResourceLoader 配置root路径是关键

WebAppResourceLoader webAppResourceLoader = new WebAppResourceLoader(patternResolver.getResource("classpath:/templates").getFile().getPath());

beetlGroupUtilConfiguration.setResourceLoader(webAppResourceLoader);

} catch (IOException e) {

e.printStackTrace();

}

//读取配置文件信息

return beetlGroupUtilConfiguration;

}

@Bean(name = "beetlViewResolver")

public BeetlSpringViewResolver getBeetlSpringViewResolver(@Qualifier("beetlConfig") BeetlGroupUtilConfiguration beetlGroupUtilConfiguration) {

BeetlSpringViewResolver beetlSpringViewResolver = new BeetlSpringViewResolver();

beetlSpringViewResolver.setContentType("text/html;charset=UTF-8");

beetlSpringViewResolver.setOrder(0);

beetlSpringViewResolver.setConfig(beetlGroupUtilConfiguration);

return beetlSpringViewResolver;

}

//配置包扫描

@Bean(name = "beetlSqlScannerConfigurer")

public BeetlSqlScannerConfigurer getBeetlSqlScannerConfigurer() {

BeetlSqlScannerConfigurer conf = new BeetlSqlScannerConfigurer();

conf.setBasePackage("com.forezp.dao");

conf.setDaoSuffix("Dao");

conf.setSqlManagerFactoryBeanName("sqlManagerFactoryBean");

return conf;

}

@Bean(name = "sqlManagerFactoryBean")

@Primary

public SqlManagerFactoryBean getSqlManagerFactoryBean(@Qualifier("datasource") DataSource datasource) {

SqlManagerFactoryBean factory = new SqlManagerFactoryBean();

BeetlSqlDataSource source = new BeetlSqlDataSource();

source.setMasterSource(datasource);

factory.setCs(source);

factory.setDbStyle(new MySqlStyle());

factory.setInterceptors(new Interceptor[]{new DebugInterceptor()});

factory.setNc(new UnderlinedNameConversion());//开启驼峰

factory.setSqlLoader(new ClasspathLoader("/sql"));//sql文件路径

return factory;

}

//配置数据库

@Bean(name = "datasource")

public DataSource getDataSource() {

return DataSourceBuilder.create().url("jdbc:mysql://127.0.0.1:3306/test").username("root").password("123456").build();

}

//开启事务

@Bean(name = "txManager")

public DataSourceTransactionManager getDataSourceTransactionManager(@Qualifier("datasource") DataSource datasource) {

DataSourceTransactionManager dsm = new DataSourceTransactionManager();

dsm.setDataSource(datasource);

return dsm;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65

在resouces包下，加META\_INF文件夹，文件夹中加入spring-devtools.properties:

restart.include.beetl=/beetl-2.3.2.jar

restart.include.beetlsql=/beetlsql-2.3.1.jar

* 1
* 2
* 3

在templates下加一个index.btl文件。

加入jar和配置beatlsql的这些bean，以及resources这些配置之后，springboot就能够访问到数据库类。

举个restful的栗子

初始化数据库的表

# DROP TABLE `account` IF EXISTS

CREATE TABLE `account` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(20) NOT NULL,

`money` double DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

INSERT INTO `account` VALUES ('1', 'aaa', '1000');

INSERT INTO `account` VALUES ('2', 'bbb', '1000');

INSERT INTO `account` VALUES ('3', 'ccc', '1000');

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

bean

public class Account {

private int id ;

private String name ;

private double money;

getter...

setter...

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

数据访问dao层

public interface AccountDao extends BaseMapper<Account> {

@SqlStatement(params = "name")

Account selectAccountByName(String name);

}

* 1
* 2
* 3
* 4
* 5
* 6

接口继承BaseMapper，就能获取单表查询的一些性质，当你需要自定义sql的时候，只需要在resouses/sql/account.md文件下书写文件：

selectAccountByName

===

\*根据name获account

select \* from account where name= #name#

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

其中“=== ”上面是唯一标识，对应于接口的方法名，“\* ”后面是注释，在下面就是自定义的sql语句，具体的见官方文档。

web层

这里省略了service层，实际开发补上。

@RestController

@RequestMapping("/account")

public class AccountController {

@Autowired

AccountDao accountDao;

@RequestMapping(value = "/list",method = RequestMethod.GET)

public List<Account> getAccounts(){

return accountDao.all();

}

@RequestMapping(value = "/{id}",method = RequestMethod.GET)

public Account getAccountById(@PathVariable("id") int id){

return accountDao.unique(id);

}

@RequestMapping(value = "",method = RequestMethod.GET)

public Account getAccountById(@RequestParam("name") String name){

return accountDao.selectAccountByName(name);

}

@RequestMapping(value = "/{id}",method = RequestMethod.PUT)

public String updateAccount(@PathVariable("id")int id , @RequestParam(value = "name",required = true)String name,

@RequestParam(value = "money" ,required = true)double money){

Account account=new Account();

account.setMoney(money);

account.setName(name);

account.setId(id);

int t=accountDao.updateById(account);

if(t==1){

return account.toString();

}else {

return "fail";

}

}

@RequestMapping(value = "",method = RequestMethod.POST)

public String postAccount( @RequestParam(value = "name")String name,

@RequestParam(value = "money" )double money) {

Account account = new Account();

account.setMoney(money);

account.setName(name);

KeyHolder t = accountDao.insertReturnKey(account);

if (t.getInt() > 0) {

return account.toString();

} else {

return "fail";

}

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52

通过postman 测试，代码已全部通过。

个人使用感受，使用bealsql做了一些项目的试验，但是没有真正用于真正的生产环境，用起来非常的爽。但是springboot没有提供自动装配的直接支持，需要自己注解bean。另外使用这个orm的人不太多，有木有坑不知道，在我使用的过程中没有遇到什么问题。另外它的中文文档比较友好。

源码下载：<https://github.com/forezp/SpringBootLearning>

参考资料

[BeetlSQL2.8中文文档](http://www.ibeetl.com/guide/#beetlsql)

优秀文章推荐：

* 更多springboot 教程：[springBoot非官方教程 | 文章汇总](http://blog.csdn.net/forezp/article/details/70341818)
* 更多springcoud 教程：[史上最简单的 SpringCloud 教程 | 文章汇总](http://blog.csdn.net/forezp/article/details/70148833)