|  |
| --- |
| Spring |
| Начало SpringDB\_31    <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.3.0.RELEASE</version>  </dependency>  <dependency>  <groupId>cglib</groupId>  <artifactId>cglib</artifactId>  <version>3.2.5</version>  </dependency>  <dependency>  <groupId>org.xerial</groupId>  <artifactId>sqlite-jdbc</artifactId>  <version>3.16.1</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>4.3.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>4.3.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.3.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>4.3.0.RELEASE</version>  </dependency>  <dependency>  <groupId>org.apache.poi</groupId>  <artifactId>poi</artifactId>  <version>3.14</version>  </dependency>  <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjrt</artifactId>  <version>1.8.10</version>  </dependency>  </dependencies>  ---------------------------------------------------------  <context:component-scan base-package=*"ru.javabegin.training.spring.\*"* />  <aop:aspectj-autoproxy proxy-target-class=*"true"* />  <bean id=*"dataSource"*  class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>  <property name=*"driverClassName"* value=*"org.sqlite.JDBC"*></property>  <property name=*"url"* value=*"jdbc:sqlite:db/SpringDB2.db"*></property>  <property name=*"username"* value=*""*></property>  <property name=*"password"* value=*""*></property>  </bean>  ---------------------------------------  @Component("sqliteDAO")  public class SQLiteDAO implements MP3Dao {    private JdbcTemplate jdbcTemplate;  DataSource dataSource=null;      @Autowired  public void setDataSource(DataSource dataSource){  this.jdbcTemplate=new JdbcTemplate(dataSource);    }  @Override  public void insert(MP3 mp3) {  String sql="insert into mp3 (name, author) VALUES (?,?)";  mp3.setId(jdbcTemplate.update(sql, new Object[]{mp3.getAuthor(), mp3.getName()}));  System.*out*.println(mp3.getId());  } |
| @Override  public void delete(MP3 mp3) {  delete(mp3.getId());  }  @Override  public MP3 getMP3ByID(int id) {  String sql = "select \* from mp3 where id=?";  return jdbcTemplate.queryForObject(sql, new MP3RowMapper(), id);  }  @Override  public int getMP3Count() {  String sql = "select count(\*) from mp3 ";  return jdbcTemplate.queryForObject(sql, Integer.class);  }  @Override  public List<MP3> getMP3ListByName(String name) {  String sql = "select \* from mp3 where upper(name) like :name";  return jdbcTemplate.query(sql, new MP3RowMapper(), name);  }  @Override  public List<MP3> getMP3ListByAuthor(String author) {  String sql = "select \* from mp3 where upper(author) like :name";  return jdbcTemplate.query(sql, new MP3RowMapper(), author);  }  @Override  public void delete(int id) {  String sql = "delete from mp3 where id=?";  jdbcTemplate.update(sql, id);  }  @Override  public int insert(MP3 mp3) {    String sql = "insert into mp3 (name, author) VALUES (?,?)";  jdbcTemplate.update(sql, new Object[] { mp3.getName(), mp3.getAuthor() });    Map<String, Object> parameters = new HashMap<String, Object>(2);  parameters.put("name", mp3.getName());  parameters.put("author", mp3.getAuthor());    return (int) insertActor.executeAndReturnKey(parameters);  }    @Override  public int insertV2(MP3 mp3) {    GeneratedKeyHolder keyHolder= new GeneratedKeyHolder();  String sql = "insert into mp3 (name, author) VALUES ( :name,:author)";  MapSqlParameterSource namedParameters = new MapSqlParameterSource();  namedParameters.addValue("name", mp3.getName());  namedParameters.addValue("author", mp3.getAuthor());    namedParameterJdbcTemplate.update(sql, namedParameters, keyHolder);  return keyHolder.getKey().intValue();  }    public Map<String, Integer> groupBy(){    String sql = "select author, COUNT(\*) as count from mp3 group by author";  return jdbcTemplate.query(sql, new ResultSetExtractor<Map<String, Integer>>(){  @Override  public Map<String, Integer> extractData(ResultSet rs) throws SQLException, DataAccessException {  Map<String, Integer> map= new TreeMap<>();    while(rs.next()){  String author=rs.getString("author");  int count =rs.getInt("count");  map.put(author, count);    }  return map;  }    });  }  @Override  public void insertList(List<MP3> list)  {  String sql="insert into mp3 (name, author) values(:name, :author)";  SqlParameterSource[] batch= SqlParameterSourceUtils.*createBatch*(list.toArray());  namedParameterJdbcTemplate.batchUpdate(sql, batch);  }  @Override  public void insertList(List<MP3> list)  {  String sql="insert into mp3 (name, author) values(:name, :author)";  SqlParameterSource[] batch = new SqlParameterSource[list.size()];  for (int i = 0; i < list.size(); i++) {  MP3 array\_element = list.get(i);  MapSqlParameterSource parameters = new MapSqlParameterSource();  parameters.addValue("name", array\_element.getName());  parameters.addValue("author", array\_element.getAuthor());    batch[i]=parameters;      }  // SqlParameterSource[] batch= SqlParameterSourceUtils.createBatch(list.toArray());  namedParameterJdbcTemplate.batchUpdate(sql, batch);  }  private static final class MP3RowMapper implements RowMapper<MP3> {  @Override  public MP3 mapRow(ResultSet rs, int rowNum) throws SQLException {  MP3 mp3 = new MP3();  mp3.setId(rs.getInt("id"));  mp3.setAuthor(rs.getString("author"));  mp3.setName(rs.getString("name"));  return mp3;  }  } |
| Aus dem Buch  <http://docs.spring.io/spring/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/jdbc.html>  int rowCount = this.jdbcTemplate.queryForObject("select count(\*) from t\_actor", Integer.class);  -  int countOfActorsNamedJoe = this.jdbcTemplate.queryForObject(  "select count(\*) from t\_actor where first\_name = ?", Integer.class, "Joe");  -  String lastName = this.jdbcTemplate.queryForObject(  "select last\_name from t\_actor where id = ?",  new Object[]{1212L}, String.class);  -  Actor actor = this.jdbcTemplate.queryForObject(  "select first\_name, last\_name from t\_actor where id = ?",  new Object[]{1212L},  new RowMapper<Actor>() {  public Actor mapRow(ResultSet rs, int rowNum) throws SQLException {  Actor actor = new Actor();  actor.setFirstName(rs.getString("first\_name"));  actor.setLastName(rs.getString("last\_name"));  return actor;  }  });  List<Actor> actors = this.jdbcTemplate.query(  "select first\_name, last\_name from t\_actor",  new RowMapper<Actor>() {  public Actor mapRow(ResultSet rs, int rowNum) throws SQLException {  Actor actor = new Actor();  actor.setFirstName(rs.getString("first\_name"));  actor.setLastName(rs.getString("last\_name"));  return actor;  }  });  -  public List<Actor> findAllActors() {  return this.jdbcTemplate.query( "select first\_name, last\_name from t\_actor", new ActorMapper());  }  private static final class ActorMapper implements RowMapper<Actor> {  public Actor mapRow(ResultSet rs, int rowNum) throws SQLException {  Actor actor = new Actor();  actor.setFirstName(rs.getString("first\_name"));  actor.setLastName(rs.getString("last\_name"));  return actor;  }  } Updating (INSERT/UPDATE/DELETE) with jdbcTemplate this.jdbcTemplate.update(  "insert into t\_actor (first\_name, last\_name) values (?, ?)",  "Leonor", "Watling");  this.jdbcTemplate.update(  "update t\_actor set last\_name = ? where id = ?",  "Banjo", 5276L);  this.jdbcTemplate.update(  "delete from actor where id = ?",  Long.valueOf(actorId)); |
| private NamedParameterJdbcTemplate namedParameterJdbcTemplate;  public void setDataSource(DataSource dataSource) {  this.namedParameterJdbcTemplate = new NamedParameterJdbcTemplate(dataSource);  }  public int countOfActorsByFirstName(String firstName) {  String sql = "select count(\*) from T\_ACTOR where first\_name = :first\_name";  SqlParameterSource namedParameters = new MapSqlParameterSource("first\_name", firstName);  return this.namedParameterJdbcTemplate.queryForObject(sql, namedParameters, Integer.class);  } |
| public void doExecute() {  this.jdbcTemplate.execute("create table mytable (id integer, name varchar(100))");  }  public int getCount() {  return this.jdbcTemplate.queryForObject("select count(\*) from mytable", Integer.class);  }  public String getName() {  return this.jdbcTemplate.queryForObject("select name from mytable", String.class);  }  public List<Map<String, Object>> getList() {  return this.jdbcTemplate.queryForList("select \* from mytable");  }    public void setName(int id, String name) {  this.jdbcTemplate.update("update mytable set name = ? where id = ?", name, id);  } |
|  |
|  |