- th:selected

用作给 HTML 元素设置选中，条件成立则选中，否则不选中。

```
@GetMapping("/select")
public ModelAndView select(){
    List<User> list = Arrays.asList(
            new User(1,"张三"),
            new User(2,"李四"),
            new User(3,"王五")
    );
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("test");
    modelAndView.addObject("list",list);
    modelAndView.addObject("name","李四");
    return modelAndView;
}
```

```
<select>
    <option
            th:each="user:${list}"
            th:value="${user.id}"
            th:text="${user.name}"
            th:selected="${user.name == name}"
    ></option>
</select>
```

结合 th:each 来使用，首先遍历 list 集合动态创建 option 元素，根据每次遍历出的 user.name 与业务数据中的 name 是否相等来决定是否要选择。

- th:attr

给 HTML 标签的任意属性赋值

```
@GetMapping("/attr")
public ModelAndView attr(){
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("test");
    modelAndView.addObject("attr","Spring Boot");
    return modelAndView;
}
```

```
<input th:attr="value=${attr}"/><br/>
<input th:value="${attr}"/>
```

Thymeleaf 对象

Thymeleaf 支持直接访问 Servlet Web 原生资源，HttpServletRequest、HttpServletResponse、HttpSession、ServletContext。

```
#request：获取 HttpServletRequest 对象
#response：获取 HttpServletResponse 对象
#session：获取 HttpSession 对象
#servletContext：获取 ServletContext 对象
```

```
@GetMapping("/servlet")
public String servlet(HttpServletRequest request){
    request.setAttribute("value","request");
    request.getSession().setAttribute("value","session");
    request.getServletContext().setAttribute("value","servletContext");
    return "test";
}
```

```
<p th:text="${#request.getAttribute('value')}"></p>
<p th:text="${#session.getAttribute('value')}"></p>
<p th:text="${#servletContext.getAttribute('value')}"></p>
<p th:text="${#response}"></p>
```

Thymeleaf 支持直接访问 session，`${#request.getAttribute('name')}` 也可以简化 `${name}`

```
@GetMapping("/servlet2")
public ModelAndView servlet2(HttpSession session){
    session.setAttribute("name","李四");
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("test");
    modelAndView.addObject("name","张三");
    return modelAndView;
}
```

```
<p th:text="${name}"></p>
<p th:text="${#request.getAttribute('name')}"></p>
<p th:text="${session.name}"></p>
<p th:text="${#session.getAttribute('name')}"></p>
```

Thymeleaf 内置对象

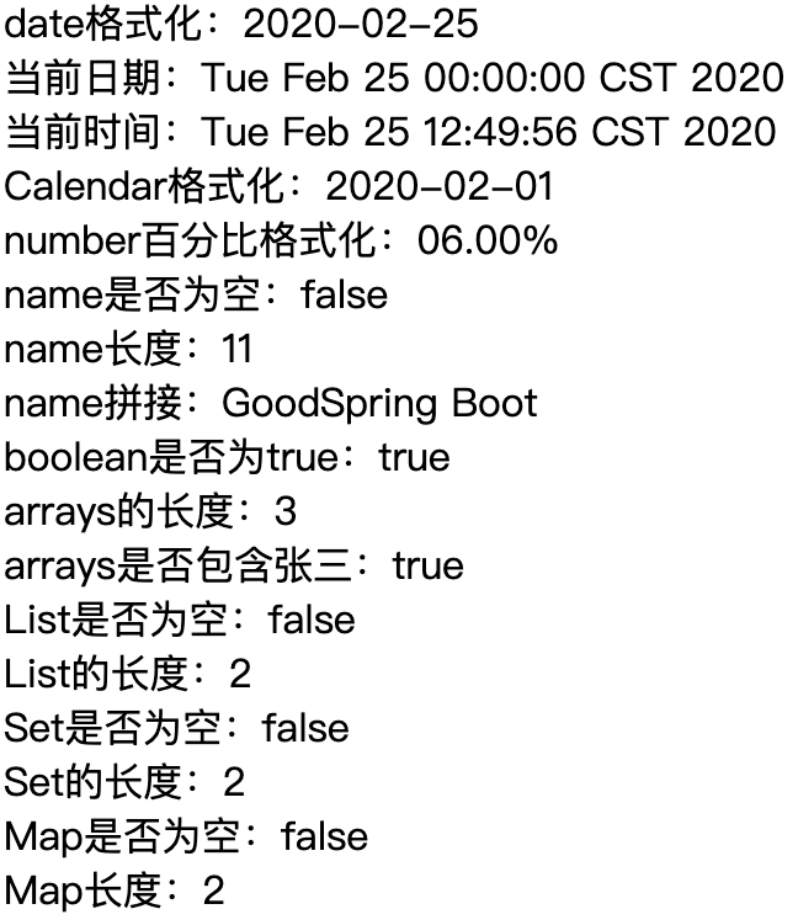
- dates：日期格式化
- calendars：日期操作
- numbers：数字格式化
- strings：字符串格式化
- bools：boolean

- arrays：数组内置对象
- lists：List 集合内置对象
- sets：Set 集合内置对象
- maps：Map 集合内置对象

```
@GetMapping("/utility")
public ModelAndView utility(){
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("test");
    modelAndView.addObject("date",new Date());
    Calendar calendar = Calendar.getInstance();
    calendar.set(2020,1,1);
    modelAndView.addObject("calendar",calendar);
    modelAndView.addObject("number",0.06);
    modelAndView.addObject("string","Spring Boot");
    modelAndView.addObject("boolean",true);
    modelAndView.addObject("array",Arrays.asList("张三","李四","王五"));
    List<User> list = new ArrayList<>();
    list.add(new User(1,"张三"));
    list.add(new User(2,"李四"));
    modelAndView.addObject("list",list);
    Set<User> set = new HashSet<>();
    set.add(new User(1,"张三"));
    set.add(new User(2,"李四"));
    modelAndView.addObject("set",set);
    Map<Integer,User> map = new HashMap<>();
    map.put(1,new User(1,"张三"));
    map.put(2,new User(2,"李四"));
    modelAndView.addObject("map",map);
    return modelAndView;
}
```

```
date格式化：<span th:text="${#dates.format(date,'yyyy-MM-dd')}"></span><br/>
当前日期：<span th:text="${#dates.createToday()}"></span><br/>
当前时间：<span th:text="${#dates.createNow()}"></span><br/>
Calendar格式化：<span th:text="${#calendars.format(calendar,'yyyy-MM-dd')}">
</span><br/>
number百分比格式化：<span th:text="${#numbers.formatPercent(number,2,2)}"></span>
<br/>
name是否为空：<span th:text="${#strings.isEmpty(string)}"></span><br/>
name长度：<span th:text="${#strings.length(string)}"></span><br/>
name拼接：<span th:text="${#strings.concat('Good',string)}"></span><br/>
boolean是否为true：<span th:text="${#bools.isTrue(boolean)}"></span><br/>
arrays的长度：<span th:text="${#arrays.length(array)}"></span><br/>
arrays是否包含张三：<span th:text="${#arrays.contains(array,'张三')}"></span>
<br/>
List是否为空：<span th:text="${#lists.isEmpty(list)}"></span><br/>
List的长度：<span th:text="${#lists.size(list)}"></span><br/>
```

```
Set是否为空：<span th:text="${#sets.isEmpty(set)}"></span><br/>
Set的长度：<span th:text="${#sets.size(set)}"></span><br/>
Map是否为空：<span th:text="${#maps.isEmpty(map)}"></span><br/>
Map长度：<span th:text="${#maps.size(map)}"></span>
```



Spring Boot 整合持久层

Spring Boot 整合 JdbcTemplate

JdbcTemplate 是 Spring 自带的 JDBC 模版组件，底层实现了对 JDBC 的封装，用法与 MyBatis 类似，需要开发者自定义 SQL 语句，JdbcTemplate 帮助我们完成数据库的连接，SQL 执行，结果集的封装。

不足之处是灵活性不如 MyBatis，因为 MyBatis 的 SQL 语句定义在 XML 中，更有利于维护和扩展，JdbcTemplate 以硬编码的方式将 SQL 直接写在 Java 代码中，不利于扩展维护。

1、pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

```
    <groupId>com.southwind</groupId>
    <artifactId>springbootdao</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>

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

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>

    </dependencies>

</project>
```

2、创建实体类

```
package com.southwind.entity;

import lombok.Data;

@Data
public class User {
    private Integer id;
    private String username;
    private String password;
    private Integer age;
}
```

3、创建 UserRepository

```
package com.southwind.repository;

import com.southwind.entity.User;

import java.util.List;

public interface UserRepository {
    public List<User> findAll();
    public User findById(Integer id);
    public int save(User user);
    public int update(User user);
    public int deleteById(Integer id);
}
```

4、创建实现类

```
package com.southwind.repository.impl;

import com.southwind.entity.User;
import com.southwind.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class UserRepositoryImpl implements UserRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public List<User> findAll() {
        return jdbcTemplate.query(
                "select * from t_user",
                new BeanPropertyRowMapper<>(User.class)
        );
    }

    @Override
    public User findById(Integer id) {
        return jdbcTemplate.queryForObject(
                "select * from t_user where id = ?",
                new Object[]{id},
```

```
                new BeanPropertyRowMapper<>(User.class)
        );
    }

    @Override
    public int save(User user) {
        return jdbcTemplate.update(
                "insert into t_user(username,password,age) values (?,?,?)",
                user.getUsername(),
                user.getPassword(),
                user.getAge()
        );
    }

    @Override
    public int update(User user) {
        return jdbcTemplate.update(
                "update t_user set username = ?,password = ?,age = ? where id
= ?",
                user.getUsername(),
                user.getPassword(),
                user.getAge(),
                user.getId()
        );
    }

    @Override
    public int deleteById(Integer id) {
        return jdbcTemplate.update(
                "delete from t_user where id = ?",
                id
        );
    }
}
```

5、Handler

```
package com.southwind.controller;

import com.southwind.entity.User;
import com.southwind.repository.impl.UserRepositoryImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/user")
public class UserHandler {
```

```
    @Autowired
    private UserRepositoryImpl userRepository;

    @GetMapping("/findAll")
    public List<User> findAll(){
        return userRepository.findAll();
    }

    @GetMapping("/findById/{id}")
    public User findById(@PathVariable("id") Integer id){
        return userRepository.findById(id);
    }

    @PostMapping("/save")
    public int save(@RequestBody User user){
        return userRepository.save(user);
    }

    @PutMapping("/update")
    public int update(@RequestBody User user){
        return userRepository.update(user);
    }

    @DeleteMapping("/deleteById/{id}")
    public int deleteById(@PathVariable("id") Integer id){
        return userRepository.deleteById(id);
    }
}
```

query

> query(String sql,RowMapper rowMapper)

```
@FunctionalInterface
public interface RowMapper<T> {
    @Nullable
    T mapRow(ResultSet var1, int var2) throws SQLException;
}
```

RowMapper 是一个接口，作用是解析结果集，将 JDBC 查询出的 ResultSet 对象转换成对应的 POJO。

> queryForObject(String sql,Object[] args,RowMapper rowMapper)

该方法用来查询一条数据，并将结果封装成一个 POJO。

update

增加、删除、修改的操作都可以调用个这个方法。

Spring Boot 整合 MyBatis

1、pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.southwind</groupId>
    <artifactId>springbootdao</artifactId>
    <version>1.0-SNAPSHOT</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.4.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>1.3.1</version>
        </dependency>

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

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>

    </dependencies>

</project>
```

2、实体类

```
package com.southwind.entity;

import lombok.Data;

@Data
public class User {
    private Integer id;
    private String username;
    private String password;
    private Integer age;
}
```

3、创建 UserRepository

```
package com.southwind.mybatis.repository;

import com.southwind.entity.User;

import java.util.List;

public interface UserRepository {
    public List<User> findAll();
    public User findById(Integer id);
    public int save(User user);
    public int update(User user);
    public int deleteById(Integer id);
}
```

4、/resources/mapping 创建 UserRepository.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.southwind.mybatis.repository.UserRepository">

    <select id="findAll" resultType="User">
        select * from t_user
    </select>

    <select id="findById" parameterType="java.lang.Integer" resultType="User">
        select * from t_user where id = #{id}
    </select>

    <insert id="save" parameterType="User">
        insert into t_user(username,password,age) values(#{username},#
{password},#{age})
```

```
    </insert>

    <update id="update" parameterType="User">
        update t_user set username=#{username},password=#{password},age=#{age}
where id = #{id}
    </update>

    <delete id="deleteById" parameterType="java.lang.Integer">
        delete from t_user where id = #{id}
    </delete>

</mapper>
```

5、创建 Handler

```
package com.southwind.controller.mybatis;

import com.southwind.entity.User;
import com.southwind.mybatis.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/user")
public class UserHandler {

    @Autowired
    private UserRepository userRepository;

    @GetMapping("/findAll")
    public List<User> findAll() {
        return userRepository.findAll();
    }

    @GetMapping("/findById/{id}")
    public User findById(@PathVariable("id") Integer id) {
        return userRepository.findById(id);

    }

    @PostMapping("/save")
    public int save(@RequestBody User user) {
        return userRepository.save(user);
    }

    @PutMapping("/update")
    public int update(@RequestBody User user){
```

```
        return userRepository.update(user);
    }

    @DeleteMapping("/deleteById/{id}")
    public int deleteById(@PathVariable("id") Integer id){
        return userRepository.deleteById(id);
    }

}
```

6、配置文件

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: root
    password: root
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.southwind.entity
```

7、创建启动类

```
package com.southwind;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@MapperScan("com.southwind.mybatis.repository")
public class MyBatisApplication {
    public static void main(String[] args) {
        SpringApplication.run(MyBatisApplication.class,args);
    }
}
```