



监听器 Listener

今日内容介绍

- ◆ 案例：使用监听器完成定时发送生日祝福邮件

今日内容学习目标

- ◆ 了解监听器执行原理
- ◆ 阐述 WEB 开发中使用到 8 种监听器
- ◆ 了解邮件协议
- ◆ 了解 Java 程序发送邮件

第1章 案例：使用监听器发送生日祝福邮件

1.1 需求

当我们使用 QQ 时，在生日当天会收到 QQ 系统发送的祝福邮件。今天我们将实现相同的功能。

1.2 相关知识点

1.2.1 监听器 Listener

1.2.1.1 概述

- 什么是监听器



- 所谓的监听器是指对整个 WEB 环境的监听,当被监视的对象发生改变时,立即调用相应的方法进行处理。
- 监听器最常见的应用场景:
 - Java SE GUI 编程
 - Android 手机开发编程
- 监听器术语:
 - 1.事件源: 被监听对象。(目标对象)
 - 2.监听器对象: 用于监听“事件源”的对象
 - 3.注册(绑定): 将“监听器对象”注册给“事件源”。当“事件源”发生某些行为时, 监听对象将被执行。
 - 4.事件: 事件源行为的称呼。
 - 5.事件对象: 在“监听器对象”中获得“事件源”.
- 监听器的编写流程
 1. 编写监听器实现类, 需要实现指定的接口
 2. 在 web.xml 文件配置监听器 (部分监听不需要配置)

```
<listener>
    <listener-class></listener-class>
</listener>
```

- JavaEE 规范规定了 8 个监听器接口, 用于监听不同的 WEB 对象。
 - 监听域对象创建与销毁
 - 监听域对象属性变更(添加, 替换, 删除)
 - 监听 session 作用域特殊 Javabean

Servlet context events	
<input type="checkbox"/> Lifecycle	I javax.servlet.ServletContextListener
<input type="checkbox"/> Changes to attributes	I javax.servlet.ServletContextAttributeListener

HTTP session events	
<input type="checkbox"/> Lifecycle	I javax.servlet.http.HttpSessionListener
<input type="checkbox"/> Changes to attributes	I javax.servlet.http.HttpSessionAttributeListener
<input type="checkbox"/> Session migration	I javax.servlet.http.HttpSessionActivationListener
<input type="checkbox"/> Object binding	I javax.servlet.http.HttpSessionBindingListener

Servlet request events	
<input type="checkbox"/> Lifecycle	I javax.servlet.ServletRequestListener
<input type="checkbox"/> Changes to attributes	I javax.servlet.ServletRequestAttributeListener

1.2.1.2 域对象本身

- ServletRequest 对象监听

I ServletRequestListener	
A	requestDestroyed(ServletRequestEvent) : void
A	requestInitialized(ServletRequestEvent) : void



ServletRequestListener	监听 Request 对象创建或销毁 javax.servlet.ServletRequestListener	
	request 创建方法 * 请求开始创建	requestInitialized(ServletRequestEvent sre)
	request 销毁方法 * 请求结束时销毁	requestDestroyed(ServletRequestEvent sre)
	事件对象	ServletRequestEvent <ul style="list-style-type: none">● ServletContext getServletContext()● ServletRequest getServletRequest()

- HttpSession 对象监听

```
▲ ① HttpSessionListener
  ● ^ sessionCreated(HttpSessionEvent) : void
  ● ^ sessionDestroyed(HttpSessionEvent) : void
```

HttpSessionListener	监听 Session 对象创建或销毁 javax.servlet.http.HttpSessionListener	
	session 创建方法 * 第一个调用 getSession()	sessionCreated(HttpSessionEvent se)
	session 销毁方法 1. 默认 30 分钟, web.xml 可配置 2. 执行 api 手动销毁, invalidate() 3. 服务器非正常关闭	sessionDestroyed(HttpSessionEvent se)
	事件对象	HttpSessionEvent <ul style="list-style-type: none">● getSession() 获得 session

- ServletContext 对象监听

```
▲ ① ServletContextListener
  ● ^ contextInitialized(ServletContextEvent) : void
  ● ^ contextDestroyed(ServletContextEvent) : void
```

ServletContextListener	监听 ServletContext 对象创建或销毁 javax.servlet.ServletContextListener	
	ServletContext 创建方法 * 服务器启动时	contextInitialized(ServletContextEvent sce)
	ServletContext 销毁方法 * 服务器正常关闭时	contextDestroyed(ServletContextEvent sce)
	事件对象	ServletContextEvent <ul style="list-style-type: none">● ServletContext getServletContext()



1.2.1.3 域对象属性

作用域属性操作： `setAttribute(k,v)` / `getAttribute(k)` / `removeAttribute(k)`

● request 作用域属性

① ServletRequestAttributeListener
● attributeAdded(ServletRequestAttributeEvent) : void
● attributeRemoved(ServletRequestAttributeEvent) : void
● attributeReplaced(ServletRequestAttributeEvent) : void

ServletRequestAttributeListener	监听 request 对象属性 attribute 添加、替换、删除 javax.servlet.ServletRequestAttributeListener	
	添加 * 第一次设置数据	attributeAdded(ServletRequestAttributeEvent srae)
	替换 * 第二次设置数据 -- 通过 event 获得 old 数据, 如果希望 获得 new 数据, 需要 request.getAttribute 获得	attributeReplaced(ServletRequestAttributeEvent srae)
	删除	attributeRemoved(ServletRequestAttributeEvent srae)
	事件对象	ServletRequestAttributeEvent <ul style="list-style-type: none">● getName()● getValue()● getServletContext()● getServletRequest()

● session 作用域属性

① HttpSessionAttributeListener
● attributeAdded(HttpSessionBindingEvent) : void
● attributeRemoved(HttpSessionBindingEvent) : void
● attributeReplaced(HttpSessionBindingEvent) : void

HttpSessionAttributeListener	监听 session 对象属性 attribute 添加、替换、删除 javax.servlet.http.HttpSessionAttributeListener	
	添加	attributeAdded(HttpSessionBindingEvent event)
	替换	attributeReplaced(HttpSessionBindingEvent event)
	删除	attributeRemoved(HttpSessionBindingEvent event)
	事件对象	HttpSessionBindingEvent <ul style="list-style-type: none">● getName()● getValue()● getSession()



- `servletContext` 作用域属性

- `ServletContextAttributeListener`

- `attributeAdded(ServletContextAttributeEvent) : void`
 - `attributeRemoved(ServletContextAttributeEvent) : void`
 - `attributeReplaced(ServletContextAttributeEvent) : void`

<code>ServletContextAttributeListener</code>	监听 <code>servletcontext</code> 对象属性 <code>attribute</code> 添加、替换、删除 <code>javax.servlet.ServletContextAttributeListener</code>	
	添加	<code>attributeAdded(ServletContextAttributeEvent event)</code>
	替换	<code>attributeReplaced(ServletContextAttributeEvent event)</code>
	删除	<code>attributeRemoved(ServletContextAttributeEvent event)</code>
	事件对象	<code>ServletContextAttributeEvent</code> <ul style="list-style-type: none">● <code>getName()</code>● <code>getValue()</code>● <code>getServletContext()</code>

1.2.1.4 特殊 javabean 在 session 作用域

--特殊的两个监听器不需要再 web.xml 配置，其他的 6 个都需配置<listener>。

- 绑定和解绑：实现指定接口 `javabean`，从 `session` 作用域存放或异常监听

- `HttpSessionBindingListener`

- `valueBound(HttpSessionBindingEvent) : void`
 - `valueUnbound(HttpSessionBindingEvent) : void`

<code>HttpSessionBindingListener</code>	监听特殊 JavaBean 在 session 作用域绑定或解绑 <code>javax.servlet.http.HttpSessionBindingListener</code> , <code>javabean</code> 必须实现该接口	
	绑定 * 给作用域添加数据	<code>valueBound(HttpSessionBindingEvent event)</code>
	解绑 * 从作用域移除数据	<code>valueUnbound(HttpSessionBindingEvent event)</code>

注意：

1. `javabean` 必须实现接口，不需要再 `web.xml` 注册。

2. `HttpSessionAttributeListener` 和 `HttpSessionBindingListener` 对比

`HttpSessionAttributeListener` 必须在 `web.xml` 注册，监听任意对象。

`HttpSessionBindingListener` 不需要再 `web.xml` 注册，只对当前 `javabean` 进行监听。



- 钝化和活化

- HttpSessionActivationListener

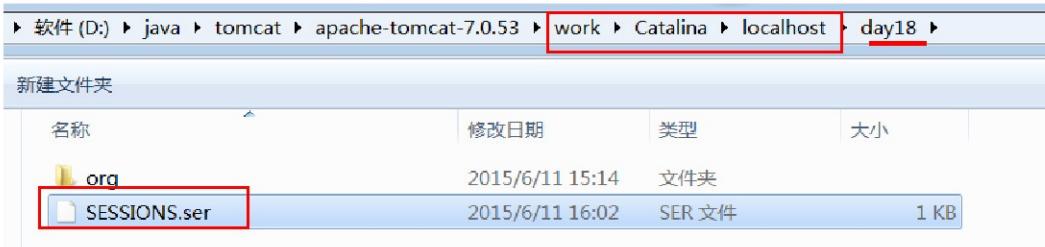
- sessionWillPassivate(HttpSessionEvent) : void
 - sessionDidActivate(HttpSessionEvent) : void

HttpSessionActivationListener	监听特殊 JavaBean 在 session 作用域钝化或活化 javax.servlet.http.HttpSessionActivationListener, javabean 必须实现该接口 钝化：在服务器正常关闭时，将 session 作用域的数据写入到文件 活化：在服务器启动时，将指定文件中的内容加载到 session 作用域。	
	活化	sessionDidActivate(HttpSessionEvent se)
	钝化	sessionWillPassivate(HttpSessionEvent se)
	事件对象	HttpSessionEvent <ul style="list-style-type: none">getSession()

注意：

- 必须实现序列号接口，java.io.Serializable，否则抛异常：java.io.NotSerializableException
- 持久化 session 数据到指定的位置：

%tomcat%\work\Catalina\localhost\项目名称\SESSIONS.ser



1.2.1.5 使用

- 步骤 1：编写 ServletContextListener 接口实现类，用于监听 ServletContext 对象的创建与销毁

```
public class PathServletContextListener implements ServletContextListener {
```

```
//初始化方法
public void contextInitialized(ServletContextEvent sce) {
    String contextLocation =
        sce.getServletContext().getInitParameter("contextLocation");
    System.out.println("contextLocation " + contextLocation);
}

//销毁方法
public void contextDestroyed(ServletContextEvent sce) {
}
```



}

● 步骤 2：在 web.xml 注册监听器

```
<!-- 配置系统初始化参数 -->
<context-param>
    <param-name>contextLocation</param-name>
    <param-value>/my2.xml</param-value>
</context-param>
```

● 步骤 3：在 web.xml 配置全局初始化参数

```
<!-- 配置监听器 -->
<listener>
    <listener-class>
        cn.itcast.web.servlet.PathServletContextListener
    </listener-class>
</listener>
```

1.2.2 电子邮件

1.2.2.1 概述

● 邮件服务器：

- 要在 Internet 上提供电子邮件功能，必须有专门的电子邮件服务器。例如现在 Internet 很多提供邮件服务的厂商：sina、sohu、163 等等他们都有自己的邮件服务器。
- 这些服务器类似于现实生活中的邮局，它主要负责接收用户投递过来的邮件，并把邮件投递到邮件接收者的电子邮箱中。
- 邮件服务器，按照提供的服务类型，可以分为发送邮件的服务器我接收邮件的服务器。

● 电子邮箱：

- 电子邮箱（E-mail 地址）的获得需要在邮件服务器上进行申请，确切地说，电子邮箱其实就是用户在邮件服务器上申请的一个帐户。用户在邮件服务器上申请了一个帐号后，邮件服务器就会为这个帐号分配一定的空间，用户从而可以使用这个帐号以及空间，发送电子邮件和保存别人发送过来的电子邮件。

1.2.2.2 邮件协议

■ SMTP 协议—发邮件协议

- 全称为 Simple Mail Transfer Protocol(简单邮件传输协议)，它定义了邮件客户端软件与 SMTP 服务器之间、以及两台 SMTP 服务器之间的通讯规则。
- 端口号：25。

■ POP3 协议—收邮件协议

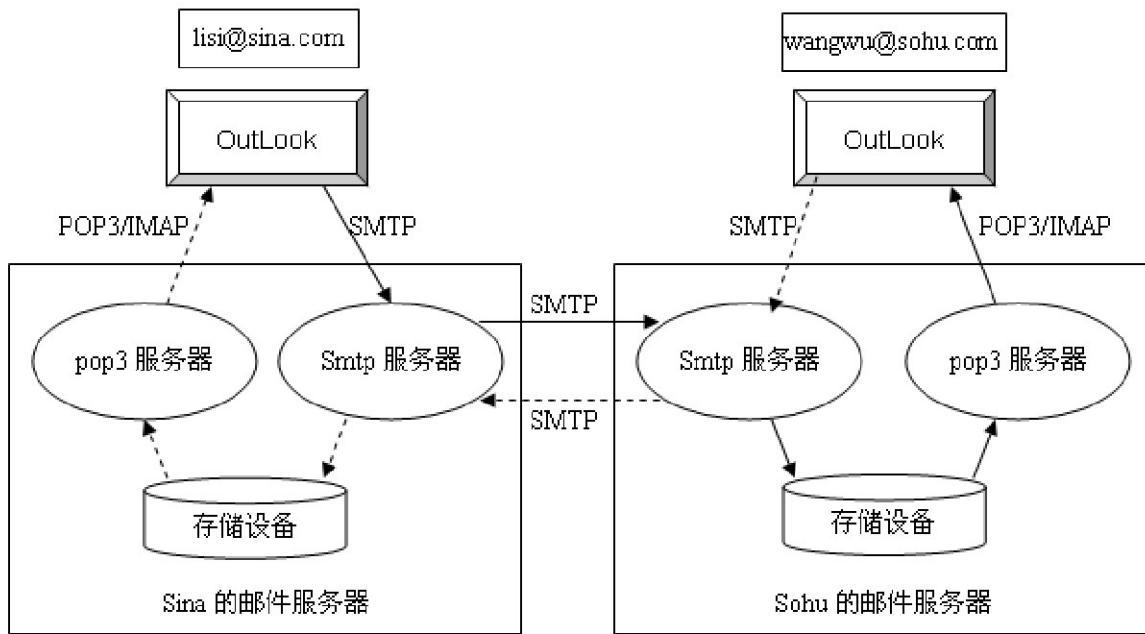
- 全称为 Post Office Protocol (邮局协议)，它定义了邮件客户端软件与 POP3 服务器的通讯规



则。

- 端口号：110.

1.2.2.3 发送邮件流程



1.2.2.4 使用 JavaMail 发送邮件

- 导入 jar 包：



- 编写实现：

```
/* 邮件发送过程
 * * smtp 协议：邮件发送协议，端口号：25
 * * pop3 协议：邮件接收协议，端口号：110
 * 使用 java 程序发送邮件，采用 smtp 协议。java 提供 javamail 用于发送邮件的，代码固定
 * 126 --> itcast@126.com //账号不存在，需要自己注册
 * 163 --> itheima@163.com //账号不存在，需要自己注册
 */

public static void main(String[] args) throws Exception {
```



```
//0.1 服务器的设置
Properties props = new Properties();
props.setProperty("mail.host", "smtp.126.com");
props.setProperty("mail.smtp.auth", "true");
//0.2 账号和密码
Authenticator authenticator = new Authenticator() {
    @Override
    protected PasswordAuthentication getPasswordAuthentication() {
        //126 账号和密码（模拟账号，需要自己注册）
        return new PasswordAuthentication("itcast", "123456");
    }
};

//1 与 126 服务器建立连接: Session
Session session = Session.getDefaultInstance(props, authenticator);

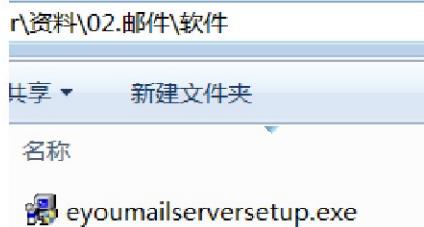
//2 编写邮件: Message
Message message = new MimeMessage(session);
//2.1 发件人（模拟账号）
message.setFrom(new InternetAddress("itcast@126.com"));
//2.2 收件人，to:收件人，cc：抄送，bcc: 暗送（密送）。（模拟账号）
message.setRecipient(RecipientType.TO,
    new InternetAddress("itheima@163.com"));
//2.3 主题
message.setSubject("这是我们得第一份邮件");
//2.4 内容
message.setContent("哈哈，您到我的商城注册了。", "text/html;charset=UTF-8");

//3 将消息进行发送: Transport
Transport.send(message);

}
```

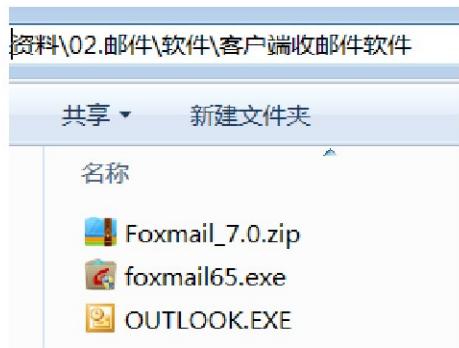
1.2.2.5 搭建本地环境

- 安装邮件服务器

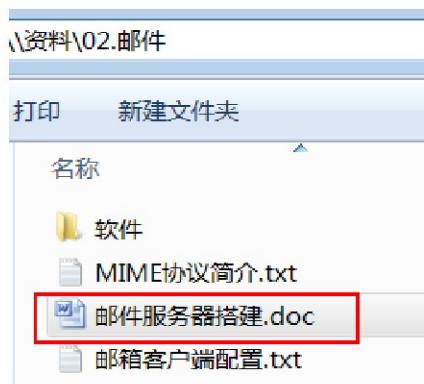




- 安装客户端软件



- 参考：



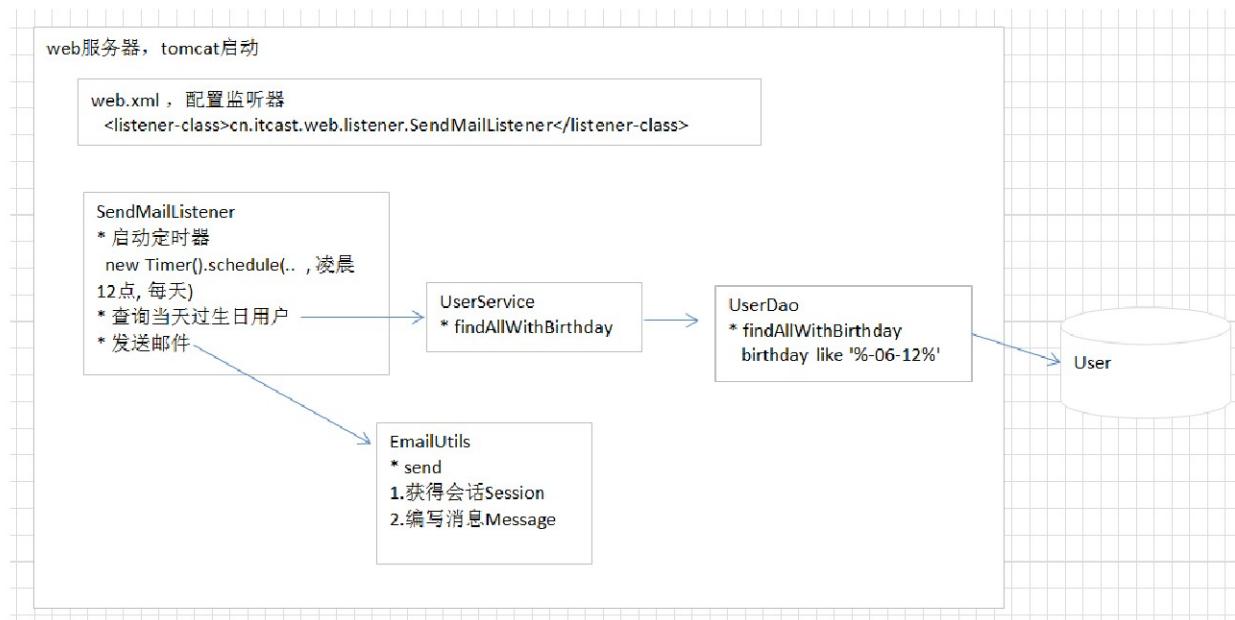
1.2.3 定时器

- JDK 提供工具类 Timer，用于触发定时器，执行 TimerTask 执行任务

```
System.out.println(new Date().toLocaleString());
//1 定时器核心类
Timer timer = new Timer();
//2 定时执行指定任务
// 参数1：需要执行的任务
// 参数2：执行任务的延迟时间，单位：毫秒
// 参数3：执行任务的轮回时间（周期），单位：毫秒
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        System.out.println(new Date().toLocaleString());
    }
}, 2000, 4000);
2016-6-12 18:19:06 延迟2秒执行
2016-6-12 18:19:08
2016-6-12 18:19:12 间隔4秒执行
2016-6-12 18:19:16
```



1.3 案例分析



1.4 案例实现

- 步骤 1: 导入 jar 包

```
WEB-INF
  lib
    c3p0-0.9.1.2.jar
    commons-beanutils-1.8.3.jar
    commons-dbtutils-1.4.jar
    commons-logging-1.1.1.jar
    jstl.jar
    mail.jar
    mysql-connector-java-5.0.4-bin.jar
    standard.jar
```

- 步骤 2: 初始化数据库

```
create database day23_db;
use day23_db;
create table t_user(
    id int primary key auto_increment,
    username varchar(50),
    birthday date,
    email varchar(100)
```



```
);

insert t_user(username,birthday,email)
values('jack','2015-12-12','itcast_lt@163.com');

insert t_user(username,birthday,email)
values('rose','2015-12-12','itcast_lt@163.com');

insert t_user(username,birthday,email)
values('tom','2015-12-12','itcast_lt@163.com');
```

● 步骤 3：导入 JdbcUtils 和 c3p0 配置文件

└─ src
 └─ cn.itcast.utils
 └─ JdbcUtils.java
 └─ c3p0-config.xml

```
<!-- 连接数据库的4项基本参数 -->
<property name="driverClass">com.mysql.jdbc.Driver</property>
<property name="jdbcUrl">jdbc:mysql://127.0.0.1:3306/day23_db</property>
<property name="user">root</property>
<property name="password">1234</property>
```

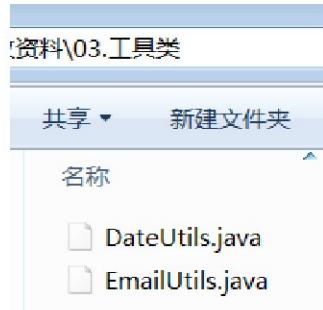
● 步骤 4：编写监听器

```
//实现类
public class SendMailListener implements ServletContextListener {
    public void contextInitialized(ServletContextEvent sce) {
    }

    public void contextDestroyed(ServletContextEvent sce) {
    }
}
```

```
//web.xml 配置内容
<listener>
    <listener-class>cn.itcast.web.listener.SendMailListener</listener-class>
</listener>
```

● 步骤 5：导入时间工具类（了解）





● 步骤 6：编写定时器

```
public class SendMailListener implements ServletContextListener {  
  
    public void contextInitialized(ServletContextEvent sce) {  
        //1 启动定时器，凌晨 12 点执行 run()方法  
        new Timer().schedule(new TimerTask() {  
            @Override  
            public void run() {  
                try {  
                    // 2. 查询所有 查询所有的用户  
                    UserService userService = new UserService();  
                    // 3. 查询当前过生日的所有人  
                    List<User> findAll = userService.findAllWithBirthday();  
                    for (User user : findAll) {  
                        EmailUtils.send(user);  
                    }  
                } catch (Exception e) {  
                    throw new RuntimeException(e);  
                }  
            }  
        }, DateUtils.getDelayTime(), DateUtils.getOneDay());  
        //      }, 100, DateUtils.getOneDay());      //测试程序，运行后 100 毫秒执行  
    }  
}
```

● 步骤 7：编写 JavaBean

```
public class User {  
    private Integer id;  
    private String username;  
    private String password;  
  
    private String email;  
    private String nickname;  
    private String gender;  
  
    private String birthday;
```

● 步骤 8：编写 service

```
public class UserService {  
    /**  
     * 查询当天过生日的用户  
     * @return  
     * @throws SQLException  
     */  
    public List<User> findAllWithBirthday() throws SQLException {
```



```
UserDao userDao = new UserDao();
return userDao.findAllWithBirthday();
}

}
```

● 步骤 9：编写 dao

```
public class UserDao {

    public List<User> findAllWithBirthday() throws SQLException {
        // 查询条件: -06-12
        String birthday      =      "-" + DateUtils.getCurrentMonth() + "-" +
DateUtils.getCurrentDay();

        System.out.println(birthday);

        QueryRunner queryRunner = new QueryRunner(JdbcUtils.getDataSource());
        String sql = "select * from t_user where birthday like ?";
        Object[] params = {"%" + birthday + "%"};
        return queryRunner.query(sql, new BeanListHandler<User>(User.class),
params);
    }

}
```

第2章 总结

