

监听器 Listener

教学导航

教学目标	案例-使用监听器完成定时生日祝福
教学方法	

一、监听器 Listener

javaEE 包括 13 门规范 在课程中主要学习 servlet 技术 和 jsp 技术

其中 servlet 规范包括三个技术点：servlet listener filter

1. 什么是监听器？

监听器就是监听某个对象的状态变化的组件

监听器的相关概念：

事件源：被监听的对象 ----- 三个域对象 request session servletContext

监听器：监听事件源对象 事件源对象的状态的变化都会触发监听器 ----- 6+2

注册监听器：将监听器与事件源进行绑定

响应行为：监听器监听到事件源的状态变化时 所涉及的功能代码 ----- 程序员编写代码

2. 监听器有哪些？

第一维度：按照被监听的对象划分：ServletRequest 域 HttpSession 域
ServletContext 域

第二维度：监听的内容分：监听域对象的创建与销毁的 监听域对象的属性变化的

	ServletContext域	HttpSession域	ServletRequest域
域对象的创建与销毁	ServletContextListener	HttpSessionListener	ServletRequestListener
域对象内的属性的变化	ServletContextAttributeListener	HttpSessionAttributeListener	ServletRequestAttributeListener

3. 监听三大域对象的创建与销毁的监听器

(1) 监听 ServletContext 域的创建与销毁的监听器 ServletContextListener

1) Servlet 域的生命周期

何时创建：服务器启动创建

何时销毁：服务器关闭销毁

2) 监听器的编写步骤（重点）：

a、编写一个监听器类去实现监听器接口

b、覆盖监听器的方法

c、需要在 web.xml 中进行配置---注册

3) 监听的方法：

ServletContextListener监听器

```

@Override
public void contextInitialized(ServletContextEvent sce) {
    System.out.println("context init....");
}

@Override
public void contextDestroyed(ServletContextEvent sce) {
    System.out.println("context destory....");
}

```

创建ServletContext时执行

事件源对象
被监听的对象

ServletContext销毁时执行

4) 配置文件：

```
<!-- 监听servletContext创建于销毁的监听器 -->
<listener>
  <listener-class>cn.itcast.listener.create.MyServletContextListener</listener-class>
</listener>
```

5) ServletContextListener 监听器的主要作用

- a、初始化的工作 :初始化对象 初始化数据 ---- 加载数据库驱动 连接池的初始化
- b、加载一些初始化的配置文件 --- spring 的配置文件
- c、任务调度----定时器----Timer/TimerTask

任务调度：

```
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
Date parse = null;
try {
    parse = format.parse("2016-03-27 24:00:00");
} catch (ParseException e) {
    e.printStackTrace();
}

//web应用一起动 就开启任务调度
Timer timer = new Timer();
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        //定时执行的任务代码
        System.out.println("runner .....");
    }
}, parse, 3000); //24*60*60*1000
```



(2) 监听 HttpSession 域的创建于销毁的监听器 HttpSessionListener

1) HttpSession 对象的生命周期

何时创建：第一次调用 request.getSession 时创建

何时销毁：服务器关闭销毁 session 过期 手动销毁

2) HttpSessionListener 的方法

```
@Override
public void sessionCreated(HttpSessionEvent se) {
    HttpSession session = se.getSession();
    System.out.println("session创建: "+session.getId());
}
```

```
@Override
public void sessionDestroyed(HttpSessionEvent se) {
    HttpSession session = se.getSession();
    System.out.println("session销毁: "+session.getId());
}
```

(3) 监听 ServletRequest 域创建与销毁的监听器 ServletRequestListener

1) ServletRequest 的生命周期

创建：每一次请求都会创建 request

销毁：请求结束

2) ServletRequestListener 的方法



```
@Override
public void requestDestroyed(ServletRequestEvent sre) {
    sre.getServletRequest();
    System.out.println("request销毁");
}

@Override
public void requestInitialized(ServletRequestEvent sre) {
    System.out.println("request创建");
}
```

4. 监听三大域对象的属性变化的

(1) 域对象的通用的方法：

setAttribute(name,value)
--- 触发添加属性的监听器的方法
--- 触发修改属性的监听器的方法
getAttribute(name)
removeAttribute(name)
--- 触发删除属性的监听器的方法

(2) ServletContextAttributeListener 监听器

```
//监听添加属性的方法
@Override
public void attributeAdded(ServletContextAttributeEvent scab) {
    System.out.println("添加属性了:"+scab.getName()+":"+scab.getValue());
}
//监听删除属性的方法
@Override
public void attributeRemoved(ServletContextAttributeEvent scab) {
    System.out.println("删除属性了:"+scab.getName()+":"+scab.getValue());
}

//监听修改属性的方法
@Override
public void attributeReplaced(ServletContextAttributeEvent scab) {
    System.out.println("修改属性了:"+scab.getName()+":"+scab.getValue());
}
```

(3) HttpSessionAttributeListener 监听器（同上）

(4) ServletRequestAttributeListener 监听器（同上）

5. 与 session 中的绑定的对象相关的监听器（对象感知监听器）

(1) 即将要被绑定到 session 中的对象有几种状态

绑定状态：就一个对象被放到 session 域中

解绑状态：就是这个对象从 session 域中移除了

钝化状态：是将 session 内存中的对象持久化（序列化）到磁盘

活化状态：就是将磁盘上的对象再次恢复到 session 内存中

面试题：当用户很对时，怎样对服务器进行优化？

(2) 绑定与解绑的监听器 HttpSessionBindingListener

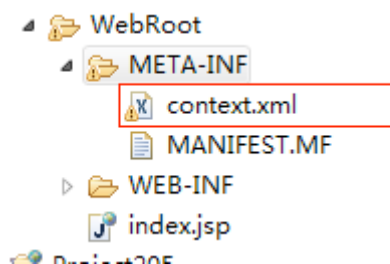


```
//感知user被绑定到session中的方法
@Override
public void valueBound(HttpSessionBindingEvent event) {
    System.out.println("user被绑定到session域中了");
    System.out.println(event.getName());
}

//感知user从session中解绑的方法
@Override
public void valueUnbound(HttpSessionBindingEvent event) {
    System.out.println("user从session域中解绑了");
    System.out.println(event.getName());
}
```

(3) 钝化与活化的监听器 HttpSessionActivationListener

可以通过配置文件 指定对象钝化时间 --- 对象多长时间不用被钝化
在 META-INF 下创建一个 context.xml



<Context>

<!-- maxIdleSwap:session 中的对象多长时间不使用就钝化 -->

<!-- directory:钝化后的对象的文件写到磁盘的哪个目录下 配置钝化的对象文件在
work/catalina/localhost/钝化文件 -->

<Manager className="org.apache.catalina.session.PersistentManager"

maxIdleSwap="1">

<Store className="org.apache.catalina.session.FileStore" directory="itcast205" />

</Manager>

</Context>

```
//钝化
@Override
public void sessionWillPassivate(HttpSessionEvent se) {
    System.out.println("costomer被钝化了....");
}
//活化
@Override
public void sessionDidActivate(HttpSessionEvent se) {
    System.out.println("costomer被活化了....");
}
```

被钝化到 work/catalina/localhost/的文件

二、邮箱服务器

1. 邮箱服务器的基本概念

邮件的客户端：可以只安装在电脑上的也可以是网页形式的

邮件服务器：起到邮件的接受与推送的作用

邮件发送的协议：

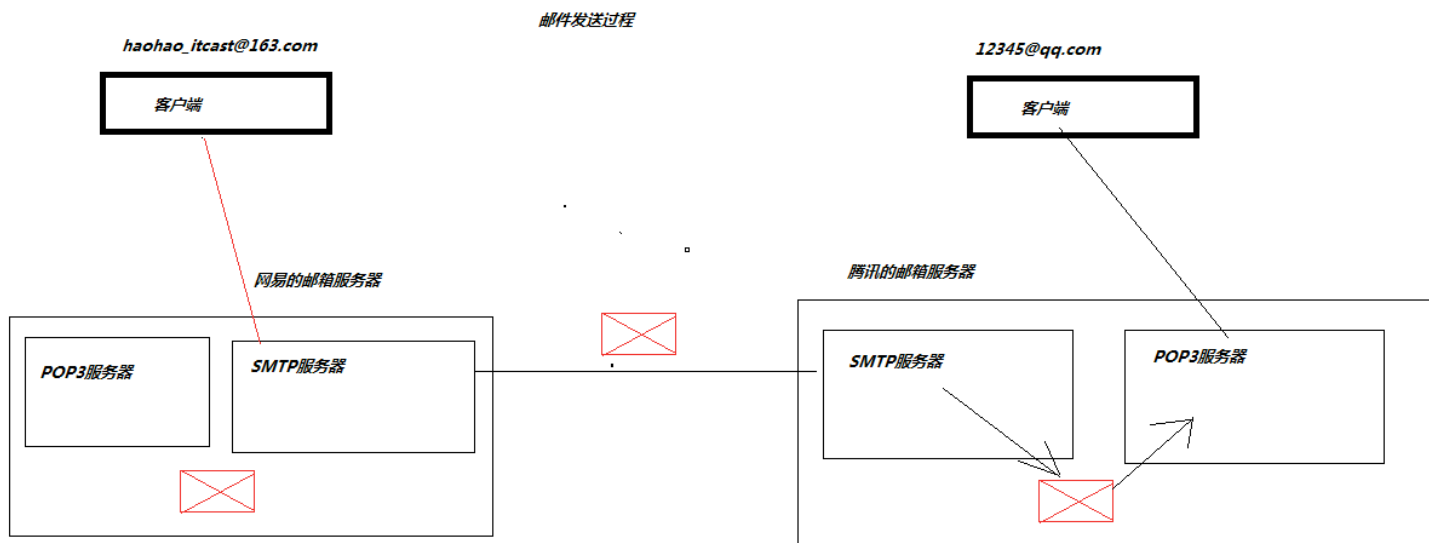
协议：就是数据传输的约束

接受邮件的协议：POP3 IMAP

服务器地址： POP3服务器: pop.126.com
SMTP服务器: smtp.126.com
IMAP服务器: imap.126.com

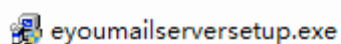
发送邮件的协议：SMTP

2. 邮箱的发送过程

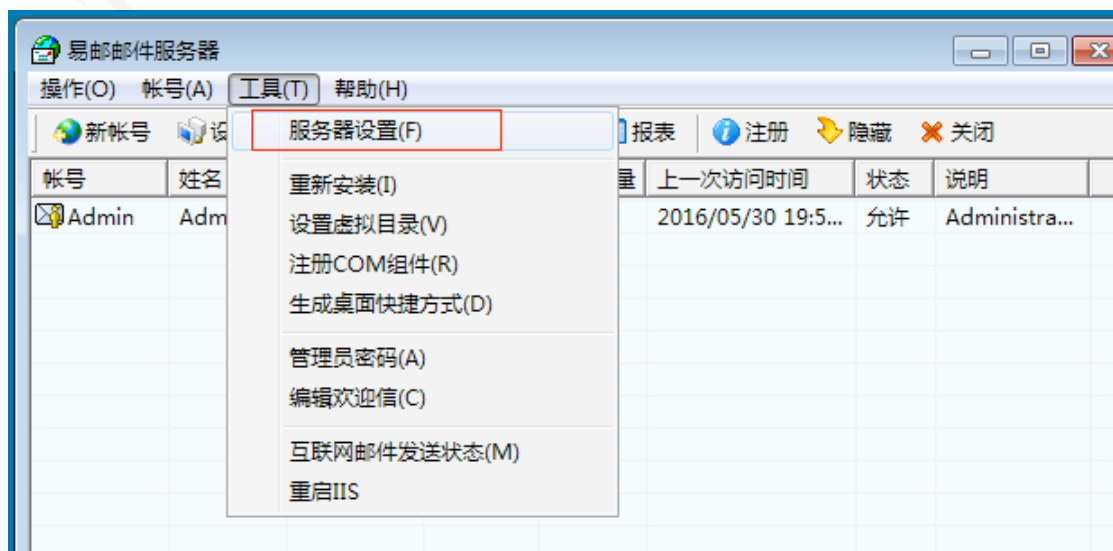


3. 邮箱服务器的安装

1) 双击邮箱服务器软件



2) 对邮箱服务器进行配置



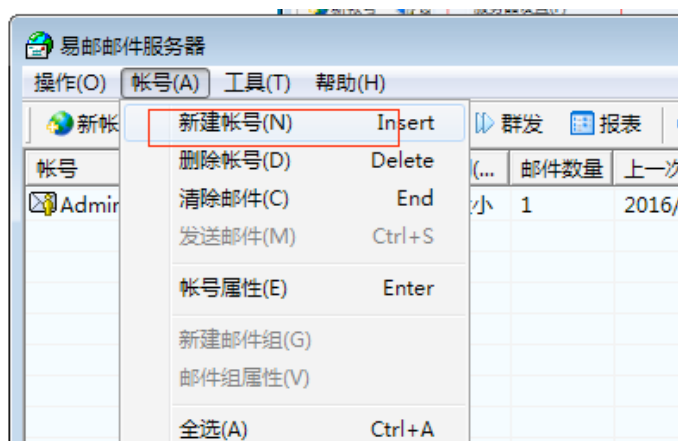
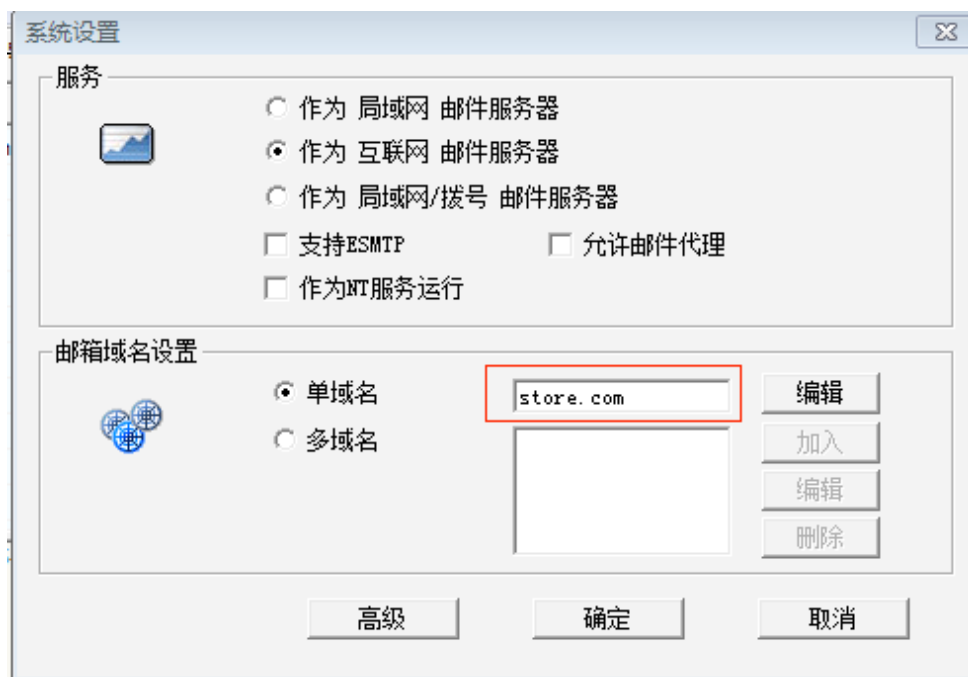


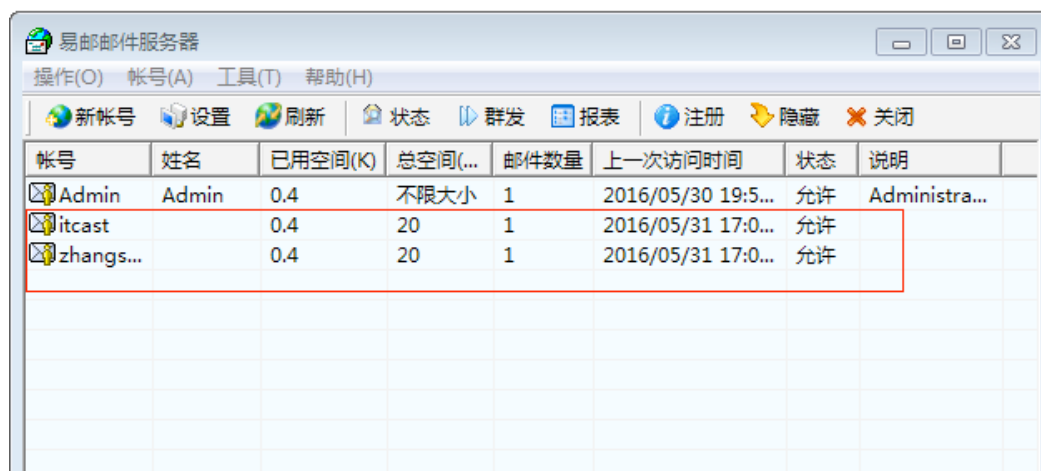
黑马程序员
www.itheima.com

传智播客旗下
高端IT教育品牌

改变中国IT教育，我们正在行动

传智播客Java学院





帐号	姓名	已用空间(K)	总空间(...)	邮件数量	上一次访问时间	状态	说明
Admin	Admin	0.4	不限大小	1	2016/05/30 19:5...	允许	Administra...
itcast		0.4	20	1	2016/05/31 17:0...	允许	
zhangs...		0.4	20	1	2016/05/31 17:0...	允许	

4. 邮箱客户端的安装



foxmail65.exe

向导



建立新的用户帐户

红色项是您需要填写的。其它选填，如“密码”可在收发邮件时再输入。

[必填] 电子邮件地址(A): zhangsan@store.com

密 码(W):

"帐户名称"是在Foxmail中显示的名称，以区分不同的邮件帐户。"邮件中采用的名称"可填您的姓名或昵称，将包含在发出的邮件中。

[必填] 帐户显示名称(U): zhangsan@store.com

邮件中采用的名称(S): zhangsan

"邮箱路径"按默认即可。您也可以自行指定邮件的保存路径。

邮箱路径(M): <默认>

选择(B)...

默认(D)

< 上一步(B)

下一步(X) >

取消(C)

帮助(H)



向导



指定邮件服务器

POP3(PostOffice Protocol 3)服务器是用来接收邮件的服务器，您的邮件保存在其上。如public.guangzhou.gd.cn。



接收服务器类型(T):

接收邮件服务器(I):

邮件帐户(A):

SMTP(Simple Mail Transfer Protocol)服务器用来中转发送您发出的邮件。SMTP服务器与POP3服务器可以不同。

发送邮件服务器(O):

+  itcast@store.com
+  zhangsan@store.com

5. 邮件发送代码

 mail.jar
 MailUtils.java