最简单的视频网站（JavaEE+FFmpeg）

2015年02月23日 00:39:24　阅读数：30451

本文记录一个最简单的视频网站系统。此前做过一些基于JavaEE中的SSH (Strut2 + Spring + Hibernate)的网站系统，但是一直没有做过一个视频网站系统，所以就打算做一个“精简”的视频网站系统，以方便以后测试以及学习使用。本视频网站支持直播（通过RTMP实现）和点播（通过HTTP实现）。为了保持精简，这个视频网站系统仅制作了网络视频的管理功能（增删改查），以及相关的参数配置功能。由于自己在JavaEE方面没有深入学习过，所以这个系统有部分功能还没搞完，以后有时间再慢慢完善。

PS：源代码以及在线演示在文章的末尾

简介

架构

系统前台显示采用了Div + CSS+ Javascript技术。其中前台显示用了一些Javascript插件，例如说生成旋转灯笼式效果的CloudCarousel，生成导航菜单的ddsmoothmenu，用于表单验证的jQuery Validation Engine等等。其中视频播放器用了FlowPlayer（包含了RTMP插件）来支持HTTP的点播和RTMP的直播。按理说视频文件上传的时候可以使用Ajax实现，但是考虑工作比较繁琐，所以暂时还是使用直接文件上传的方式。

系统的后台使用了JavaEE系统中最传统的三层架构：Action层，Service层以及DAO层。其中Action层使用了Struts2框架，用于处理前台页面传来的请求；DAO层使用了Hibernate框架，用于和数据库的交互。Spring框架则用于将Action层，Service层以及DAO层整合起来。系统的前后台交互主要使用了JSTL标签和EL表达式。此外考虑到国际化方面的要求，采用了Struts2的i18n方式，将所有页面上的文字都抽取坐来保存到单独的文件中，这样就可以实现多种语言的显示。

系统开始运行后，会开启两个线程VideoThumbnailThread和VideoTranscoderThread。VideoThumbnailThread用于截取视频的缩略图，VideoTranscoderThread则用于转码上传的视频。这两个线程都是通过调用ffmpeg.exe完成相应的功能。

此外，系统中还包含了查看媒体信息的功能。该功能通过调用MediaInfo.dll完成，不再详述。

整个系统的框架如下图所示，后文中再详细记录每部分的功能。

效果

系统主要包含了以下几个页面：

主页（index.jsp）：进入系统的第一个页面。

视频列表（videolist.jsp）：列表显示视频资源的页面。按照视频源的类型的不同可以分为点播（VOD）视频列表和直播（Live）视频列表。

视频编辑（videoedit.jsp）：编辑视频资源的页面。按照视频源的类型的不同可以分为点播（VOD）编辑和直播（Live）编辑。按照操作的不同可以分为添加（Add）和编辑（Edit）。

视频内容（videocontent.jsp）：查看视频的内容的页面。按照视频源的类型的不同可以分为点播（VOD）视频内容和直播（Live）视频内容。

设置（configure.jsp）：设置页面。用于配制系统的参数。

关于（about.jsp）：显示系统相关的一些信息的页面。

首页

网站首页的如下图所示。Logo位于左上角，菜单栏位于右上角。为了美观一些，页面上方做了一个灯箱效果的特效，随机显示一些视频的缩略图。页面下方则分别列出“点播列表”和“直播列表”中最近添加的几个视频。

点播列表

点播列表页面如下图所示。在该页面中，可以对点播视频进行简单的管理：查看内容，添加，修改以及删除。

直播列表

点播列表页面如下图所示。直播列表实质上使用了和点播列表相同的页面。在该页面中，可以对直播视频进行简单的管理：查看内容，添加，修改以及删除。

视频内容页面

视频内容页面如下所示。该页面中包含了一个FlowPlayer播放器，该播放器用于播放视频内容。

点播视频内容页面如下所示。

直播视频内容页面如下所示。尽管表面上看点播和直播页面完全相似，但是实际上点播和直播的机制是不一样的。点播依靠的是HTTP渐进式下载，而直播依靠的是FlowPlayer的RTMP组件。

视频添加页面

点播视频添加页面如下所示。该页面用于上传一个点播视频。“名称”输入框用于指定视频的名称，“文件”用于指定上传的视频文件，“简介”可以指定针对该视频的一段简介。其中，“名称”和“文件”两个输入框为必填项，否则无法提交该页面的表单。

直播视频添加页面如下所示。从图中可以看出直播和点播的添加页面很类似。它们的不同在于：点播是上传一个视频文件，而直播是指定一个URL（目前该URL只支持RTMP）。和点播类似，“名称”和“URL”两个输入框为必填项，否则无法提交该页面的表单。

视频编辑页面

点播视频编辑页面如下所示。该页面用于编辑已经完成上传的点播视频的信息。

直播视频编辑页面如下所示。该页面用于编辑直播视频的信息。

配置

系统的配置页面如下图所示。该页面用于配置系统的各种设置（以转码设置为主）。每次配置完成后重启系统后设置生效。包含以下参数。

- 视频编码器：转码使用的视频编码器，默认值为“libx264”。
- 视频码率 (bps) ：视频编码的码率（单位为bps），默认值为500000（500kbps）。
- 视频帧率 (fps) ：视频编码的帧率（单位为fps），默认值为25。
- 音频编码器：转码时候使用的音频编码器，默认值为“libmp3lame”。
- 音频采样率 (Hz) ：音频编码的采样率（单位为Hz），默认值为22050（22.05kHz）。
- 音频码率 (bps) ：音频编码的码率（单位为bps），默认值为64000（64kbps）。
- 视频宽 (pixel) ：转码后视频的宽（单位为pixel），默认值为640。
- 视频高 (pixel) ：转码后视频的高（单位为pixel），默认值为360。
- 使用水印：选择转码后的视频是否加水印。如果该选项选择“是”，则会在转码后的视频中添加水印。
- 水印文件路径：水印文件的路径，此处为文件相对于网站根目录的路径，默认为“watermark/svw.png”。

- 水印位置-x坐标 (pixel) ：水印在视频中的位置——x坐标（单位为pixel），默认值为5。
- 水印坐标-y坐标 (pixel) ：水印在视频中的位置——y坐标（单位为pixel），默认值为5。
- 保持宽高比并且填充黑边：选择转码后是否保持宽高比。如果该选项选择“是”，则转码后的视频会保持宽高比，并且当输入和输出视频宽高比不一样的时候，会在输出视频中添加黑边。如果该选项选择“否”，则转码后的视频会拉伸成输出分辨率。
- 输出视频封装格式：输出视频的封装格式，默认为“flv”。
- 视频截图位置 (sec) ：获取视频缩略图的位置（单位为sec），默认为5sec。该默认值的含义是获取视频第5秒处的视频帧作为视频的缩略图。
- 上传视频存放文件夹：用于存放上传的视频文件，默认值为“videoori”。
- 转码视频存放文件夹：用于存放转码的视频文件，默认值为“video”。
- 视频截图存放文件夹：用于存放视频的截图，默认值为“videothumbnail”。

MediaInfo信息

MediaInfo信息位于视频内容页面中，在默认的情况下是不显示的。通过修改网页中代码可以显示视频的MediaInfo信息。

转码前文件信息如下图所示，从图中可以看出转码前的文件格式是MPEG-PS，视频编码为MPEG-1，音频编码为MP2。

转码后文件信息如下图所示，从图中可以看出转码前的文件是FLV格式的，视频编码为H.264，音频编码MP3。

多国语言

本系统支持多国语言。选择右上角菜单的“语言”->“English”可以将系统切换到英文。英文界面如下图所示。

英文版的视频点播列表如下图所示。

英文版的视频内容页面如下图所示。

英文版的配置界面如下图所示。

网站部分

下面简单记录一下网站部分的关键源代码。分成三个部分：数据库、前台和后台。

数据库

系统包含了4张表：video，videostate，category，configure。下面简单记录每个表的含义。

Video表

Video表用于存储视频记录。每一个视频对应Video表中的一条记录。该表中的字段如下表所示。

名称	类型	键	简介
id	int	主键	标识
name	varchar		名称
intro	varchar		简介
edittime	datetime		编辑时间
categoryid	int	外键	所属类别
islive	int		是否是直播
url	varchar		处理后视频URL
oriurl	varchar		上传视频URL（点播）
thumbnailurl	varchar		缩略图URL
videostateid	int	外键	状态
remark	varchar		备注

**其中点播视频的URL为视频文件的相对路径。*

下面例举Video表中的几项和URL有关数据（受限于篇幅，省略其它字段）。

Id	name	islive	url	Oriurl	Thumbnailurl
1	Avatar	0	video/1.flv	videoori/Avatar Blueray 3D.flv	videothumbnail/1.jpg
2	建国大业	0	video/2.flv	videoori/建国大业.mpg	videothumbnail/2.jpg
3	Sintel	0	video/3.flv	videoori/sintel.wmv	videothumbnail/3.jpg
4	Warcraft III	0	video/4.flv	videoori/Warcraft3_End.avi	videothumbnail/4.jpg
5	CUC IESchool	0	video/5.flv	videoori/cuc_ieschool.mkv	videothumbnail/5.jpg
6	中国合伙人	0	video/6.flv	videoori/中国合伙人.flv	videothumbnail/6.jpg
7	那些年，我们一起追的女孩	0	video/7.flv	videoori/那些年，我们一起追的女孩.mp4	videothumbnail/7.jpg
8	春晚是什么？	0	video/8.flv	videoori/春晚是什么？.mov	videothumbnail/8.jpg
9	Forrest Gump IMAX	0	video/9.flv	videoori/Forrest Gump IMAX.mp4	videothumbnail/9.jpg
10	屌丝男士	0	video/10.flv	videoori/屌丝男士.mov	videothumbnail/10.jpg
11	Titanic	0	video/11.flv	videoori/Titanic.mkv	videothumbnail/11.jpg
12	北广传媒移动电视	1	rtmp://www.bj-mobiletv.com:8000/live/live1		videothumbnail/12.jpg
13	香港电视台	1	rtmp://live.hkstv.hk.lxdns.com/live/hks		videothumbnail/13.jpg
14	东莞电视台	1	rtmp://ftv.sun0769.com/dgrtv1/mp4:b1		videothumbnail/14.jpg
15	看看新闻网	1	rtmp://live.kksmg.com:80/live/mp4:Stream_1		videothumbnail/15.jpg
16	绍兴新闻综合	1	rtmp://www.scbtv.cn/live/new		videothumbnail/16.jpg
17	央广购物	1	rtmp://wx.cnrmall.com/live/flv		videothumbnail/17.jpg
19	亚太卫视	1	rtmp://58.61.150.198:1935/live/Livestream		videothumbnail/19.jpg
20	CCTV中学生	1	rtmp://ams.studytv.cn/livepkgr/264		videothumbnail/20.jpg
22	广州综合	1	rtmp://116.199.115.228/live/gztv_tv		videothumbnail/22.jpg
23	睛彩广州	1	rtmp://116.199.115.228/live/cmmb		videothumbnail/23.jpg
24	深圳娱乐	1	rtmp://tv.sznews.com:1935/live/live_233_mc43		videothumbnail/24.jpg

25	南阳新闻综合	1	rtmp://61.136.113.35/tslsChannelLive/zyys888/live		videothumbnail/25.jpg
26	大庆综合	1	rtmp://live1.baihuwang.com:1935/live/zh		videothumbnail/26.jpg
27	温州新闻综合	1	rtmp://livetv.dhtv.cn:1935/live/news		videothumbnail/27.jpg

Videostate表

Videostate表用于存储视频的状态。该表中字段如下表所示。

名称	类型	键	简介
id	int	主键	标识
name	varchar		名称
order	int		顺序
cssstyle	varchar		CSS样式
remark	varchar		备注

目前按照执行的顺序定义了4种状态：等待上传，等待截图，等待转码，完成。其中“CSS样式”用于辅助显示视频的状态。该表的内容如下

id	name	order	cssstyle	remark
1	等待上传	1	background:#CCFFFF	
2	等待截图	2	background:#00FF99	
3	等待转码	3	background:#00FF00	
4	完成	4	background:#FFFFFF	

Category表

Category表用于存储视频的分类。该表可以用于确定视频的分类，目前还没有做太多开发。

名称	类型	键	简介
id	int	主键	标识
name	varchar		名称
parentid	int		父类别
remark	varchar		备注

Configure表

Configure表用于存储系统配置信息。该表中字段如下表所示。

名称	类型	键	简介
id	int	主键	标识
name	varchar		名称
val	varchar		值
remark	varchar		备注

该表的内容如下。

Id	name	val	remark
1	transcoder_vcodec	libx264	Video encoder
2	transcoder_bv	500000	Video bitrate
3	transcoder_framerate	25	Video frame rate
4	transcoder_acodec	libmp3lame	Audio encoder
5	transcoder_ar	22050	Audio sample rate
6	transcoder_ba	64000	Audio bitrate
7	transcoder_scale_w	640	Video width
8	transcoder_scale_h	360	Video height
9	transcoder_watermarkuse	true	Use Watermark
10	transcoder_watermark_url	watermark/svw.png	Watermark file's URL
11	transcoder_watermark_x	5	Watermark's location (x)
12	transcoder_watermark_y	5	Watermark's location (y)
13	transcoder_keepaspectratio	true	Keep original aspect ratio
14	transcoder_outfmt	flv	Output file format
15	thumbnail_ss	5	When to get thumbnail (from start)
16	folder_videoori	videoori	Folder to save Upload Video
17	folder_video	video	Folder to save Transcode Video
18	folder_thumbnail	videothumbnail	Folder to save Thumbnail of Video

前台

前台部分没有太多可以记录的，基本上都是各种DIV和CSS的调整。其中使用了较多的JSTL标签以及EL表达式。此外使用了Struts2标签用于支持国际化。基本的jsp页面只有6个：

index.jsp：首页。
videolist.jsp：视频列表——直播视频列表，点播视频列表。
videoedit.jsp：视频编辑——直播视频添加，直播视频编辑，点播视频添加，点播视频编辑。
videocontent.jsp：视频内容——直播视频内容，点播视频内容。
configure.jsp：配置页面。
about.jsp：关于页面。

除了基本页面之外，还有一些辅助页面：

cfooter.jsp：页脚。
cheader.jsp：标题栏。
csidebar_recent.jsp：侧边栏（最近添加）。
error.jsp：错误页面。

FlowPlayer播放器

前台页面中用于视频播放的播放器是FlowPlayer。FlowPlayer播放器安装有3个步骤：
（1）包含FlowPlayer的Js文件

```html
<script type="text/javascript" src="videoplayer/flowplayer-3.2.8.min.js"></script>
```

（2）添加一个<a>标记

```html
<a href="http://www.mywebsite.com/myVideo.flv"
   style="display:block;width:425px;height:300px;"
   id="player">
</a>
```

（3）添加一段Javascript代码，其中要包含FlowPlayer的Swf文件

```html
<script language="JavaScript">
  flowplayer("player", "path/to/the/flowplayer-3.2.18.swf");
</script>
```

FlowPlayer通过HTTP播放点播视频文件的源代码如下所示。${video.url}中保存了视频的相对URL。

```html
<a id="player" href="${pageContext.request.scheme}://${pageContext.request.serverName}:${pageContext.request.serverPort}${pageContext
quest.contextPath}/${video.url}">
</a>
<script>
   flowplayer("player", "videoplayer/flowplayer-3.2.8.swf");
</script>
```

FlowPlayer通过RTMP播放直播视频文件的示例代码如下所示。${video.url}中保存了视频的URL。播放RTMP的时候用到了FlowPlayer的RTMP Plugin。需要注意的是，RTMP的URL需要拆分开来分别填到不同的位置。在这里通过split('/')函数按照"/"将字符串分离为字符串数组。然后将"protocol://server/app"填至plugins的netConnectionUrl字段，playpath填至clip的url字段。

```html
      <a id="player">
      </a>
<!-- Parse RTMP URL -->
<script>
      str='${video.url}';
      arr=str.split('/');
      //rtmp://server/app/playpath
      protocol=arr[0];
      server=arr[2];
      app=arr[3];
      playpath=arr[4];
lowplayer("player", "videoplayer/flowplayer-3.2.8.swf",{
        clip: {
          //scaling:'orig',
          url: playpath,
          provider: 'rtmp',
          live:'true'
        },
        plugins: {
           rtmp: {
             url: 'videoplayer/flowplayer.rtmp-3.2.8.swf',
             netConnectionUrl: protocol+'//'+server+'/'+ app
           }
       }
    });

script>
```

后台

后台部分采用了JavaEE的SSH (Struts2 + Spring + Hibernate)的三层架构，其中的代码不再详细记录。三层分别为Action层、Service层和DAO层。其中Action层采用了Struts2框架，DAO层采用了Hibernate框架，而Spring则整合了这三个层面。

BaseService/BaseDAO

SSH三层架构虽然在解耦合方面做得很好，但是却导致开发起来非常繁琐，严重降低了开发的速度。为了方便起见，本系统中在Service层编写了BaseService用于完成通用的Service操作；在DAO层也编写了BaseDAO用于完成通用的DAO操作。这样除非有特殊需求，一般情况下只要调用BaseService中的方法就可以完成各种基本的操作。BaseService包含的接口如下所示。

```java
/**
 * 最简单的视频网站
 * Simplest Video Website
 *
 * 雷霄骅 Lei Xiaohua
 *
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序是一个最简单的视频网站视频。它支持
```

```
 * 本程序是一个最简单的视频网站视频。包支持
 * 1.直播
 * 2.点播
 * This software is the simplest video website.
 * It support:
 * 1. live broadcast
 * 2. VOD
 */
package service;

import java.util.List;

/**
 * @author 雷霄骅
 * 对Object的Service
 * 包含了一些通用的方法
 */
public interface BaseService {
         /**
          * 保存一个对象
          * @param object 一个对象
          */
          public void save(Object object);
          /**
           * 更新一个对象
           * @param object 一个对象
           */
          public void update(Object object);
          /**
           * 删除一个对象
           * @param object 一个对象
           */
          public void delete(Object object);
          /**
           * 根据ID读取一个指定名称的对象
           * @param targetName 对象的名称
           * @param id 对象的ID
           * @return 一个对象
           */
          public Object ReadByID(String targetName,int id);

          @SuppressWarnings("rawtypes")
          /**
           * 获取指定类型的所有对象
           * @param targetName 对象类型名称
           * @return 对象的列表
           */
          public List ReadAll(String targetName);

          @SuppressWarnings("rawtypes")
          /**
           * 根据“属性-值”获取多个指定类型的对象
           * @param targetName 对象类型名称
           * @param propertyName 对象中属性的名称
           * @param propertyValue 对象中属性的值
           * @return 对象的列表
           */
          public List ReadByProperty(String targetName,String propertyName,Object propertyValue);

          /**
           * 根据“属性-值”获取一个指定类型的对象
           * @param targetName 对象类型名称
           * @param propertyName 对象中属性的名称
           * @param propertyValue 对象中属性的值
           * @return 一个对象
           */
          public Object ReadSingle(String targetName,String propertyName,Object propertyValue);

          /**
           * 获取多个指定类型的对象，可以限定获取对象数目的多少，并且根据特定的属性进行排序。
           * @param targetName 对象类型名称
           * @param propertyName 对象中属性的名称，用于排序
           * @param num 结果对象列表的最大数目
           * @param order 排序方式，可以选择“asc”或者“desc”
           * @return 对象的列表
           */
          public List ReadLimitedByOrder(String targetName,String propertyName,int num,String order);

          /**
           * 获取指定类型的对象的数量。
           * @param targetName 对象类型名称
           * @return 数量
           */
          public int ReadCount(String targetName);
          /**
           * 根据“属性-值”为条件，获取指定类型的对象的数量。
           * @param targetName 对象类型名称
           * @param propertyName 对象中属性的名称
           * @param propertyValue 对象中属性的值
           * @return 数量
           */
         public int ReadCountByProperty(final String targetName,String propertyName, Object value);
```

```
104.          /**
105.           * 两个功能：
106.           * 1.根据“属性-值”获取多个指定类型的对象
107.           * 2.限定获取对象数目的多少，并且根据特定的属性进行排序。
108.           * @param targetName 对象类型名称
109.           * @param readpropertyName 对象中属性的名称，用于获取对象
110.           * @param readvalue 对象中属性的值
111.           * @param orderpropertyName 对象中属性的名称，用于排序
112.           * @param num 结果对象列表的最大数目
113.           * @param order 排序方式，可以选择“asc”或者“desc”
114.           * @return
115.           */
116.          public List ReadByPropertyAndLimitedByOrder(final String targetName, final String readpropertyName,
117.                              final Object readvalue,final String orderpropertyName, final int num, final String order);
118.
119.  }
```

FFmpeg部分

截取缩略图线程和转码线程

系统在启动后会开启两个线程：VideoThumbnailThread和VideoTranscoderThread。其中VideoThumbnailThread线程会不停的检测需要截取缩略图的视频，调用相应的FFmpeg命令截取缩略图；VideoTranscoderThread线程会不停的监测需要转码的视频，调用相应的FFmpeg命令转码视频。何以确定视频是否需要截取缩略图以及是否需要转码？这是通过video表中的videostateid字段来标识的。视频开始上传后，该视频记录会被标记为“等待上传”（第1步）；上传完毕后，该视频记录会被标记为“等待截图”（第2步）；截图完毕后，该视频记录会被标记为“等待转码”（第3步）；转码完毕后，该视频记录会被标记为“完成”（第4步）。VideoThumbnailThread会不停地检查系统中“等待截图”的视频，截图完成后将视频标记为“等待转码”；VideoTranscoderThread会不停地检查系统中“等待转码”的视频，转码完成后将视频标记为“完成”。

FFmpeg和Java的整合

FFmpeg和Java整合的过程中有以下几个需要注意的地方：

（1）路径的处理

FFmpeg处理视频的时候需要用到绝对路径，所以涉及到绝对路径获取的问题。使用ServletContext的getRealPath("/")方法可以获得当前Web应用根目录的绝对路径。例如在自己的电脑上下述代码可以获得Web应用的据对路径：

```java
ServletContext servletContext;
System.out.println(servletContext.getRealPath("/"));
```

执行完后输出为：

```plain
D:\MyEclipseWorkspace\.metadata\.me_tcat\webapps\simplest_video_website\
```

相对路径中的URL中路径的分隔采用“/”（正斜杠，Unix系统使用），而上述代码中路径的分隔采用“\”（反斜杠，Windows系统使用）。如果把相对文件路径和绝对目录路径拼接起来传递给FFmpeg的话，FFmpeg是可以识别的，但是这样一来路径中一会“/”一会“\”会给人一种很别扭的感觉，因此可以使用String的replace()方法将“\”统一替换为“/”，这样就整齐多了。例如下面代码：

```java
ServletContext servletContext;
System.out.println(servletContext.getRealPath("/").replace('\\', '/'));
```

执行完后输出为：

```plain
D:/MyEclipseWorkspace/.metadata/.me_tcat/webapps/simplest_video_website/
```

（2）FFmpeg命令行的调用

FFmpeg命令行的的调用可以分成两个步骤：

（a）生成符合设置的命令

这一步骤实际上就是一个简单的字符串拼接的过程。根据配置的参数拼接成相应的转码命令。需要注意的是，在输入和输出的文件路径两边要加上双引号。否则当文件路径中包含空格的时候，会导致路径解析错误。

（b）调用命令行

调用命令行使用Runtime.getRuntime().exec(cmdstr)方法就可以了。其中cmdstr常见的格式有两种：

```
[plain]
1. cmd /c {FFmpeg Command}
2. cmd /c start {FFmpeg Command}
```

第一种格式是在本窗口中直接执行命令，第二种格式是新打开一个窗口执行命令。第一种方法我在JavaEE环境中测试有问题，而且不弹出窗口不便于调试，所以使用第二种执行方法。

（c）等待调用的命令行执行完毕

Runtime.getRuntime().exec()在调用命令后就直接返回当前线程了。这不符合实际的需求。实际中需要系统完成FFmpeg转码工作后，才能做下一步操作。可以用Process的waitFor()方法阻塞当先线程直至调用程序运行结束。

截取缩略图命令

FFmpeg截取缩略图命令如下：

```
[plain]
1. ffmpeg -y -i "ourtime.flv" -ss 5 -s 220x110 -f image2 -vframes 1 "thumbnail.jpg"
```

参数含义如下：

-y：输出文件重名的时候，自动覆盖。
-i：输入文件路径（可以是相对路径或者绝对路径）。
-ss：截取缩略图的时间点，这里是5s处。
-s：输出缩略图的分辨率，这里是220x110。
-f：输出文件格式，这里的image2代表文件格式为图片。
-vframes：输出视频帧的个数，这里是1。
最后一个参数为输出的缩略图文件路径。

转码命令

FFmpeg转码命令如下：

```
[plain]
1. ffmpeg -y -i "ourtime.flv" -vcodec libx264 -b:v 500000 -r 25 -acodec libmp3lame -b:a 64000 -ar 22050 -
vf scale=w=640:h=360:force_original_aspect_ratio=decrease,pad=w=640:h=360:x=(ow-iw)/2:y=(oh-ih)/2[aa];movie=svw.png[bb];[aa][bb]over
lay=x=5:y=5 "ourtime_convert.flv"
```

参数含义如下：

-y：输出文件重名的时候，自动覆盖。
-i：输入文件路径（可以是相对路径或者绝对路径）。
-vcodec：视频编码器，这里是libx264。
-b:v：视频码率，这里是500000bps。
-r：视频帧率，这里是25fps。
-acodec：音频编码器，这里是libmp3lame。
-b:a：音频码率，这里是64000bps。
-ar：音频采样率，这里是22050Hz。
-vf：滤镜，用于图像拉伸以及水印叠加。
最后一个参数为输出的视频文件路径。

Filter（滤镜）配置

几句话介绍一下Filter的配置：

- Filter配置参数使用在FFmpeg的-vf参数之后
- Filter之间可以使用“,”连接，连接之后前一个Filter的输出作为后一个Filter的输入。几个Filter构成一个FilterChain（滤镜链），每个FilterChain以“;”结尾。几个FilterChain可以构成一个FilterGraph（滤镜图）。可以在Filter的Pad上添加“标签”用于连接其它Filter，Filter标签的形式为“[xx]”（其中“xx”可以随意写一些字符，只要可以起到标记作用就可以了）。
- Filter的可以使用多个参数，参数之间使用“:”分割。

下面分别记录几个滤镜的使用方法。

【scale滤镜】

scale滤镜使用libswscale对图像进行拉伸。它的参数含义如下：

w：输出图像的宽。

h：输出图像的高。

force_original_aspect_ratio：保持视频宽高比的方法，可以使用如下值：（1）disable——不保持宽高比；（2）decrease——需要的时候降低宽或者高；（3）increase——需要的时候提高宽或者高。

有关force_original_aspect_ratio可以举个例子：输入视频分辨率是1920x800，设置滤镜为：

```
[plain]
1. scale=w=640:h=360:force_original_aspect_ratio=decrease
```

输出的视频分辨率为640x266。

【pad滤镜】

pad滤镜用于给拉伸后的图像加“黑边”。经过scale滤镜处理之后，视频的宽一定小于等于640，而视频的高一定小于等于360，此时需要使用pad滤镜填充视频的两边（上下或者是左右），保证输出的视频的分辨率为640x360。

Pad滤镜有四个基本的参数：

w：填充后视频的宽度

h：填充后视频的高度

x：输入视频的左上角在填充后视频中的x坐标

y：输入视频的左上角在填充后视频中的y坐标

一个基本的如法如下所示：

```
[plain]
1. pad=w=640:h=480:x=0:y=40
```

上述滤镜将视频填充为640x480，同时将输入视频的左上角放在（0，40）的位置上。

在给视频加黑边的过程中，需要把输入的视频放在填充后视频的中央，而pad滤镜中输入视频的位置是以左上角来确定的，因此确定输入视频的左上角的坐标是一个比较麻烦的事。对此，pad滤镜提供了以下几个变量。

iw：输入视频的宽

ih：输入视频的高

ow：输出视频的宽

oh：输出视频的高

通过上述几个变量，可以得知如果想把输入视频放在填充后视频的中央，输入视频的x坐标应该为(ow-iw)/2，y坐标应该为(oh-ih)/2。

【movie滤镜】

Movie滤镜属于Source滤镜，它用于读取水印图片文件（一般情况下是一个PNG文件）。有关movie滤镜有一个地方需要特别注意：它在Windows下似乎不支持输入路径为绝对路径。因为Windows下文件的绝对路径通常为“D:\test\.....\watermark.png”，即其中开头的盘符后面跟着一个“:”。“:”在Filter中是一个特殊字符，会被解析成参数的分隔符，从而导致输入的绝对路径被解析为2个参数，最后造成错误。这个问题当时我调试了半天仍然没有得到解决。因此，Windows下使用movie滤镜的时候，需要保证水印文件就在当前的工作目录中，才能正常运行。

【overlay滤镜】

Overlay滤镜用于叠加两路输入。具体到本文的系统中就是叠加水印文件到视频文件上。它常用的有两个参数：

x：水印左上角的x坐标

y：水印左上角的y坐标

经过上述几个滤镜的处理，最终就可以得到一个分辨率为640x360，保持宽高比（加黑边），叠加过水印的视频。

项目主页

Simplest Video Website

项目主页

开源中国： http://git.oschina.net/leixiaohua1020/simplest_video_website

Github： https://github.com/leixiaohua1020/simplest_video_website

在线演示

http://www.velab.com.cn:8080/svw/

PS：由于本系统没有做用户验证功能，所以访客可以随意添加删除视频。最近发现视频经常被修改或者删除掉，这样就失去了演示的意义。还请访客们尽量少修改演示系统上的视频.....

安装步骤

1.安装JavaEE环境：
(1)下载安装JDK
(2)下载安装Tomcat
(3)下载安装MySQL
2.FFmpeg
(1)下载并且解压缩FFmpeg可执行程序
(2)把FFmpeg的bin目录(其中包含ffmpeg.exe)添加至系统的"path"环境变量(重要，这样才可以在系统任意目录中使用ffmpeg命令)
3.复制程序
(1)修改Webroot\WEB-INF\classes\hibernate.cfg.xml中数据库的用户名和密码
(2)拷贝WebRoot目录至Tomcat的webapps目录，重新取个名字(例如"svw")
(3)在MySQL中创建数据库"svw"，在其中执行svw.sql，创建数据库中的表，并且添加测试数据
4.启动Tomcat
5.使用浏览器访问http://localhost:8080/svw

文章标签： 视频 网站 FFmpeg JavaEE

个人分类： 我的开源项目

所属专栏： FFmpeg