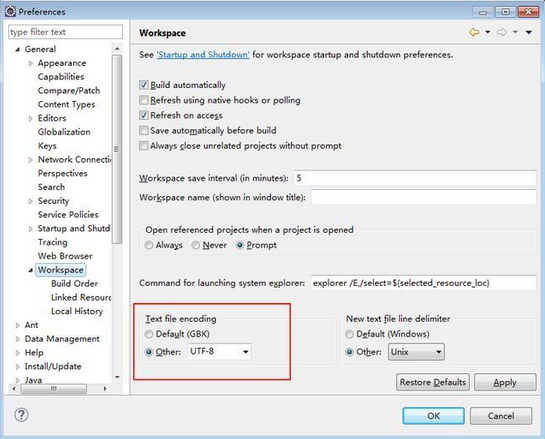
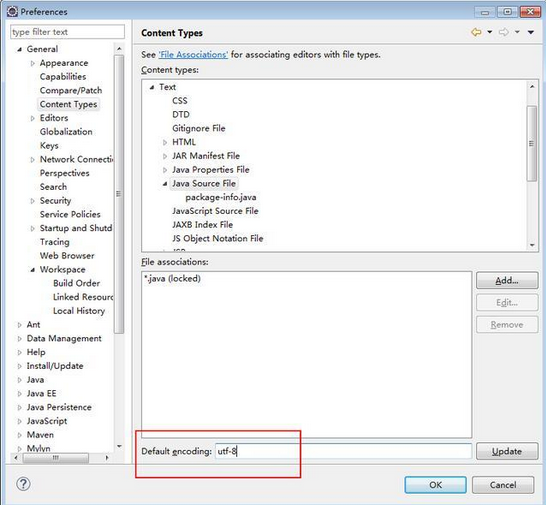
1. **eclipse workspace中的字符编码：**



windows->Preferences->general->Workspace

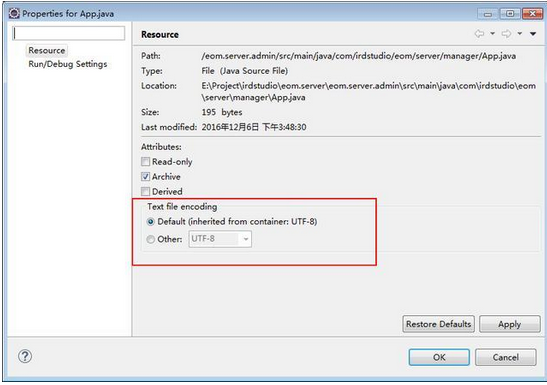
以后新建立工程其属性对话框中的Text file encoding即为UTF-8，这个也是安装完Eclipse之后应该立刻做的第一件事，但是这种修改只对当前的workspace有效，如果新建或者切换到一个新的workspace下，还是得做同样的步骤。



windows->Preferences->general->Content Types

点开Text，选择Java Source File，在下面的Default encoding输入框中输入UTF-8，点Update，则设置Java文件编码为UTF-8。其他java应用开发相关的文件如：properties、XML等已经由Eclipse缺省指定，分别为ISO8859-1，UTF-8，如开发中确需改变编码格式则可以在此指定。

**2、Java源文件编码格式指定**



先中文件,右键属性打开

指定文件编码格式为UTF-8，作用**：**

(1)、当我们用编辑器编写java源文件，程序文件在保存时会采用操作系统默认的编码格式（一般我们中文的操作系统采用的是GBK编码格式）形成一个.java文件。java源文件是采用操作系统默认支持的file.encoding编码格式保存的。通过System.out.println(System.getProperty("file.encoding")); 可以查看系统的file.encoding参数值。

(2)、JDK在编译的时候，如果没有用-encoding参数指定java源程序的编码格式，则javac首先获得操作系统默认采用的编码格式(file.encoding)，也即在编译java程序时，若不指定源程序文件的编码格式，JDK首先获得操作系统默认的编码格式，xp是GBK，然后jdk就把java源程序从该编码格式转化为java内部的unicode格式放入内存中。然后，javac把转换后的unicode格式的文件进行编译成.class类文件，此时.class文件是unicode编码的，它暂放在内存中，紧接着，JDK将此以unicode编码的编译后的class文件保存到操作系统中形成.class文件。最终获得的.class文件是内容以unicode编码格式保存的类文件，它内部包含源程序中的中文字符串，只不过此时它己经由安装系统编码格式转化为unicode格式了。

也就是说，指定源文件编码格式将告诉编译器从何种编码格式转为UNICODE，以生成最终的.class，就是说我们常见的.class文件中的内容无论是中文字符还是英文字符，他们都已经转换为Unicode编码格式了。

**3、javac -encoding**

见上段描述，如果设置了文件编码格式，就不需要指定了，在windows与linux下编译都不会有问题。

**4、JSP编码格式指定**

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

我们知道JSP页面是需要转换为servlet的，在转换过程中肯定是要进行编码的，在上面代码中有两个地方存在编码：pageEncoding、contentType的charset。其中pageEncoding是jsp文件本身的编码，而contentType的charset是指服务器发送给客户端时的内容编码。

pageEncoding

JVM将JSP编译为.java文件，JVM首先会获取pageEncoding的值，如果该值存在则采用它设定的编码来编译，否则则采用file.encoding编码来编译。

charset

后台经过处理后将结果输送到客户端，如果设置了charset则浏览器就会使用指定的编码格式进行解码，否则采用默认的ISO-8859-1编码格式进行解码处理。

**5、Tomcat server.xml中URIEncoding**

提交请求到服务器，主要有GET跟POST两种方式，利用request.setCharacterEncoding("UTF-8");来设置Tomcat接收请求的编码格式，只对POST方式提交的数据有效，对GET方式提交的数据无效!

要设置GET的编码，可以修改server.xml文件中的配置：

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" URIEncoding="UTF-8" />

如果没有设置则会使用默认的ISO-8859-1来解码，假如我们在页面将编码设置为UTF-8，而URIEncoding设置的不是或者没有设置，那么服务器进行解码时就会产生乱码。这个时候我们一般可以通过new String(request.getParameter("name").getBytes("iso-8859-1"),"utf-8") 的形式来获取正确数据。

**6、spring过滤器中的编码设置**

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

源代码中有以下两行代码：

request.setCharacterEncoding(this.encoding); //用途：设置客户端post过来的数据的编码格式。

response.setCharacterEncoding(this.encoding); //用途：设置服务器将数据发送到客户端时的编码格式，这行代码要在设置forceEncoding为true时才起作用。

**7、java -Dfile.encoding**

设置启动JVM进程时系统属性 file.encoding 为 UTF-8，在运行java程序时JVM首先会把保存在操作系统中的class文件读入到内存中，这个时候内存中class文件编码格式为Unicode，然后JVM运行它。如果需要用户输入信息，则会采用file.encoding编码格式对用户输入的信息进行编码同时转换为Unicode编码格式保存到内存中。程序运行后，将产生的结果再转化为file.encoding格式返回给操作系统并输出到界面去，如果是servlet以GET或POST的形式来接收数据，则参见第5段的说明。

简单的说：源文件编码的格式与java进程的file.encoding编码格式一样，那么在控制台的输入与输出则不会乱码，对于通过get或post接收的数据则通过需要过滤器统一设置编码格式，见第5、6段的说明。

我们常见的websphere、weblogic在linux下运行时，通常需要为java进程设置-Dfile.encoding参数就是这个原因。

**8、JDBC编码**

java程序与数据库的连接都是通过JDBC驱动程序来连接的，而JDBC驱动程序默认的是ISO-8859-1编码格式的，也就是说我们通过java程序向数据库传递数据时，JDBC首先会将Unicode编码格式的数据转换为ISO-8859-1的编码格式，然后在存储在数据库中，即在数据库保存数据时，默认格式为ISO-8859-1，如果数据库服务设定了字符集则会按设定的格式转化，或者在配置jdbc连接串时就需要指定连接的字符编码格式。

**所以统一都设置为UTF-8是个好方案。。。**