

Pentaho 本地化指南

原文版本	1.1.5
原文链接	
翻译	王娜、覃辉
校对	覃辉
Pentaho 中文讨论组 QQ 群: 12635055 论坛: http://www.bipub.org/bipub/index.asp	

Copyright © 2006 Pentaho Corporation. Redistribution permitted. All trademarks are the property of their respective owners.

For the latest information, please visit our web site at www.pentaho.org

目录

LOCALIZATION GUIDE	- 1 -
本地化指南.....	- 1 -
译者：王娜.....	ERROR! BOOKMARK NOT DEFINED.
1 介绍	- 3 -
2 开始	- 3 -
3 一般概念.....	- 3 -
4 翻译SAMPLES.....	- 4 -
4.1 查看你的 SAMPLES 翻译结果	- 6 -
5 翻译PENTAHO APPLICATION	- 6 -
5.1 查看你的APPLICATION 翻译结果	- 8 -
5.2 在 INTERNET EXPLORER 中改变语言首选项.....	- 8 -
5.3 在FIREFOX中改变语言首选项.....	- 8 -
6 UNICODE 支持.....	- 9 -
6.1 PROPERTIES 文件.....	- 9 -
6.2 XML 文件.....	- 9 -
6.3 XSL文件.....	- 9 -
6.4 JFREEREPORT 定义	- 9 -
7 提交你的翻译.....	- 10 -
8 此文档没有覆盖的内容.....	- 10 -
9 其他资源.....	- 10 -

1 介绍

这个指南将帮助你将英文版的 Pentaho BI 平台翻译成其他语言。你需要预配置的安装包和一个文本编辑器。用户接口和日志文件的所有消息均存放在文本文件中。这些文本文件的名字均以 **.properties** 为后缀，位于很多文件夹中。为支持一种新的语言，你需要为你的语言创建 **.properties** 文件。这些文件的命名是有标准的，我们已经为几种语言创建好了文件。

2 开始

1. 首先访问 <http://www.pentaho.org/discussion> 上的 Internationalization 论坛，看看是否有其他人已经开始进行你的语言上的翻译了。

2. 如果你想翻译成 French, German, Spanish, Italian, Portuguese 或 Chinese, 我们已经为你创建好了 **.properties** 文件。如果你想翻译成其他语言，发送 email 到 communityconnection@pentaho.org 或在 Pentaho 论坛上发帖。

3. 从 SourceForge 下载名为 **pentaho_demo.x.x.x.zip** 的预配置安装包。确保你知道如何运行 demo 网页，portal 页面和 samples。帮助信息请查看 **Getting Started Guide** 和论坛信息。

4. 在预配置安装包和 **pentaho-solutions** 目录中寻找后缀为 **.properties** 的文件，你会看到如下类似信息：

- a. *.properties (English)
- b. *_fr.properties (French)
- c. *_de.properties (German)
- d. *_it.properties (Italian)
- e. *_es.properties (Spanish)
- f. *_pt.properties (Portugese)
- g. *_cn.properties (Chinese)

5. 如果你的语言不在列表中，或你想创建某种语言的特定版本，例如 French Canadian，请联系 communityconnection@pentaho.org，我们将提供相应帮助。

标准语言和国家代码的列表请参考以下网页：

- 语言代码：<http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt>
- 国家代码：http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

3 一般概念

向用户显示的所有文本信息均存储在 **.properties** 文件中。每当平台需要一块文本，就加载适当的 **.properties** 文件，从中读取必需的文本。对于系统支持的每种语言，都有 **.properties** 文件。两个字符的语言代码和两个字符的国家代码（可选）附加在基本的文件名上，以标识文件

的本地版本。例如：

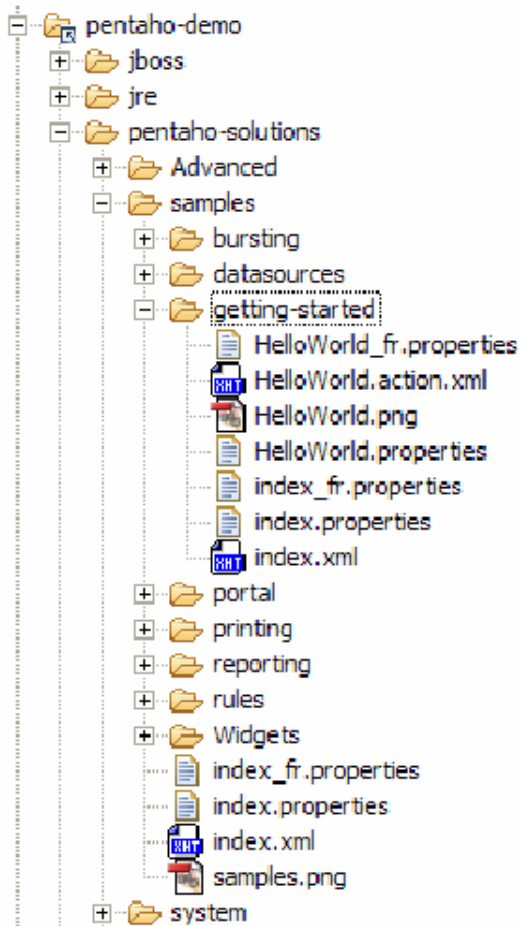
- **messages.properties:** messages 文件的默认英语版本
- **messages_fr.properties:** messages 文件的 French 版本
- **messages_de.properties:** messages 文件的 German 版本
- **messages_fr_CA.properties:** messages 文件的 French Canadian 版本

当用户发出一个请求后，平台探测那个用户的国家和语言，并定位文件的正确语言版本。

在运行时读取 **.properties** 文件，并不编译进平台。这使得执行翻译和查看结果是一个很简单的过程。

命名约定的完整解释请参考本文最后，**其他资源**一节中的参考文献。

4 翻译 Samples



开始翻译平台的最轻松方式是翻译 **samples**。预配置安装包的 **pentaho-solutions** 目录包含运行 **sample solutions** 的所有必需文件。用户可以看见 **.properties** 文件中的所有文本，他们均应该被翻译。这些文件的根目录是：

`<pentaho-demo>/pentaho-solutions/`

这张图显示了 `samples` 的标准位置，以及 `.properties` 文件的默认和 French 版本。在这个实例中，当一个英语用户请求 `samples` 目录下的 `index.xml` 文件时，平台会在此目录下寻找 `index.properties` 文件。当一个 French 用户登录后，它会寻找 `index_fr.properties`。对于一个 German 用户，找不到 `index_de.properties`，因此使用默认的 `index.properties`。

如果你是一个 German 翻译者，你需要创建一个名为 `index_de.properties` 的新文件，添加 German 版本的被翻译文本，并刷新你的浏览器，这样将会为 German 用户显示 German 信息。

为了选择正确的文本，在 `base document` 和 `.properties` 文件中的文本之间有一个 `mapping scheme`。对于实例 `index.xml`，以下是文件中的文本：

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<index>
  <name>%name</name>
  <description>%description</description>
  <icon>solutions.png</icon>
  <visible>true</visible>
</index>
```

以下是 `index.properties` 文件中的内容：

```
name=Pentaho BI Components Demo
description=These sample JSP pages show ...
```

我们已经自动为几种语言生成了 `resource` 包。我们也将运行这个生成工具，以将新的 `resources` 合并进本地化的包中。如果从基本 `resource` 包中删除一个 `resource`，它也将自动的从本地化版本的包中删除，并放置在 `bundle header` 的一个 `comment` 区域中。当一个新的 `resource` 被自动的添加进一个包时，它将包含一个唯一的标识符。例如：

```
name=[fr_57] Pentaho BI Components Demo
description=[fr_58] These sample JSP pages show ...
```

当你在页面上发现一个需要翻译的字符串时，这些唯一的标识符（例如 `fr_57`，`fr_58`）将会很有用。通过使用一个标识符，你可以搜索（使用一个 IDE，或 Windows: 开始 -> 搜索，或 `unix find`）包含这个 `resource` 的文件。这样你可以不必在大量 `resource` 文件中苦苦搜索，节省大量时间。当翻译完一个 `resource` 时，一定要删除标识符，因为它不再有用。

为翻译 XML 文件，我们使用 Eclipse 标准的 mapping XML 文本，在 `.properties` 文件中每个名字前使用一个 `%` 标记。在这个实例中，XML 文件中的 `%name` 将被 `.properties` 文件中的 `name= key` 后的文本所替换。`samples navigation JSP` 使用 `index.xml` 文件产生一套

solutions 所使用的 names, descriptions 和 image。以下是 [index_fr.properties](#) 文件的内容：

```
name=Démo De Composants de BI De Pentaho
description=Ces pages témoin JSP montrent comment ...
```

注意：这个实例 French 翻译是由一个翻译程序生成的，不要太较真。

翻译 samples 的步骤：

1. 从 samples 目录开始
2. 打开用于你的语言的每个 .properties 文件。
3. 编辑等号 ‘=’ 右边的英语文本，使用你的语言中的正确文本替换它。

重要：不要改变等号‘=’左边的文本。

例如：name=Démo De Composants

4.1 查看你的 samples 翻译结果

在你翻译每个文件，或每一行时，你可以查看结果，仅仅通过刷新你的浏览器，一些文本就会发生改变。Action documents 的 names 和 descriptions 被缓存，并必须被发布。为了发布，浏览到 Pentaho BI Components Demo 的顶层。默认 URL 是 <http://localhost:8080/pentaho>。点击 [Settings and Services](#)，然后选择 [Update Settings and Content](#)。为 “[Solution Repository](#)” publisher 点击 “[Publish](#)” 链接。在翻译时，保证这个页面可访问是很有用的。

在翻译 samples 时会遇到 3 种文件类型。在翻译完他们的 .properties 文件后，可以如下验证我们的修改：

index.xml：用于在 samples 的组之间进行浏览。你可以浏览至那个组，查看 index.properties 文件的改变结果。Getting Started 就是一个这样的组。

***.url**：定义和 samples 一起显示，用于发起 actions 的 URLs。Settings and Services” 组的 ” Update Settings and Content” 选择就是一个实例。

***.action.xml**：实现 BI solutions 的 Action documents。选择验证你的语言改变的 action。

当你准备共享你的工作时，你可以将之发送至 Pentaho，以包含在平台中。指令请参考[提交你的翻译一节](#)。

5 翻译 Pentaho Application

Pentaho [application](#) 目录树中的 .properties 文件包含应用程序文件中所使用的文本。这些包含在错误消息，debug 信息和用户接口文本。这些文件均命名为 [messages.properties](#)。如上节所述，我们已经自动生成了带有唯一标识符的 resource 包来加速翻译。如果你略过了上一节，可以返回去查看更多信息。在预配置安装中，[application resource 包](#)位于 [WEB-INF](#) 目录下。默认的，WEB-INF 目录的路径如下：

```
/pentaho-demo/jboss/server/default/deploy/pentaho.war/WEB-INF
```

在典型使用期间，这些文件中的所有文本对于用户并不均是可见的。为了更清楚些，我们在 `.properties` 文件中为每个文本块均使用了一个命名约定。

前缀	意义
USER_	这段文本对用户可见。
ERROR_	这段文本表示系统中的一个错误。如果 log level 是 ERROR 或更高，则在 event logs 中可见。
WARN_	这段文本表示系统中的一个警告。如果 log level 是 WARN 或更高，则在 event logs 中可见。
INFO_	这段文本表示系统中的一个错误。如果 log level 是 INFO 或更高，则在 event logs 中可见。
DEBUG_	这段文本表示用于问题跟踪的详细系统信息。如果 log level 是 DEBUG 或更高，则在 event logs 中可见。

你翻译 messages 的优先级顺序如下：

1. **WEB-INF\portlet.properties** 中的所有文本
2. 对于以下路径，翻译所有带有 **USER_**，**ERROR_**，**WARN_**，**DEBUG_** 的文本名。
 - WEB-INF\classes\org\pentaho\locale
 - WEB-INF\classes\org\pentaho\ui\component
 - WEB-INF\classes\org\pentaho\ui\portlet
 - WEB-INF\classes\org\pentaho\ui\component\charting
 - WEB-INF\classes\org\pentaho\ui\portlet\charting
 - WEB-INF\classes\org\pentaho\ui\servlet

如果你仅想翻译预配置安装和 samples，你仅仅需要翻译

<pentaho-demo>/jboss/server/default/deploy/pentaho.war/WEB-INF 下的这些文件：

- \portlet.properties
- \classes\org\pentaho\locale\messages.properties

这和 Pentaho 预配置安装包之前的发布版本很不同。之前每个 Java 包拥有自己的 ResourceBundle 管理器 Messages.java 以及一套名为 messages.properties (每个本地版本一个) 的属性文件。如上所示，现在我们已经将我们的所有 resources 统一进了一个包，其使用一套通用的 manager 类，**org\pentaho\messages**。

每个本地版本都有一个 messages.properties，我们已经将之前翻译的所有包合并进他们的本地特定的 counterpart 中。

这应该更容易定位要翻译的字符串，因为他们均位于共同的位置。

如同 samples, **.properties** 文件的格式是 **name=message**。你必须小心不要修改等号‘=’左边的任何东西。**所有的 messages 必须在一行上。不要在 .properties 文件中添加回车键。**

在 application messages 中, 有特殊的**可替换型参数**标记, 它们像 {0} 和 {1} 等等。他们用于构建 messages, 例如: **HelloWorld.USER_HELLO_WORLD_TEXT=\nHello World. {0}\n**

{0} 表示一个占位符, 不应该被改变。这些占位符的文本来自于数据, 被 application 插入。原文的所有占位符必须出现在被翻译的 string 中。占位符的测试位置和他们出现的顺序可以被修改, 以匹配你的语言的语法。因此, 我们的第二个 example 可能如下所示:

HelloWorld.USER_HELLO_WORLD_TEXT =\nBonjour Monde. {0}\n

5.1 查看你的 application 翻译结果

为查看你的改变, 你必须重启应用服务器。使用 stop-pentaho 和 startpentaho 来完成。文本的 name 可以帮助断定从哪里寻找你的结果。在这个实例中, 当执行 Hello World 组件时, 你能看到显示给用户的一个 message。

从你的浏览器测试

如果你想在你的 web 浏览器的语言间进行切换, 你可以改变你的 web 浏览器的语言首选项。

5.2 在 Internet Explorer 中改变语言首选项

在 Internet Explorer 中, 按照以下步骤改变你的语言首选项:

1. 从主菜单, 浏览至**工具**菜单, 然后选择 **Internet 选项...**
2. 在**常规标签页**, 选择**语言...** 按钮。
3. 在**语言**对话框, 选择**添加...** 按钮。
4. 你将打开一个对话框, 从中你可以选择你需要的语言。选择你想首选使用的语言, 然后选择 OK。
5. 现在你选择的语言应该出现在语言对话框。选择你刚才添加的语言, 使用上移按钮, 直到它出现在列表的顶部。
6. 选择 OK 直到关闭了所有的对话框。
7. 刷新你的网页。

5.3 在 Firefox 中改变语言首选项

在 Mozilla Firefox 中, 按照以下步骤改变你的语言首选项:

1. 从主菜单, 浏览至 **Tools** 菜单, 然后选择 **Options...**
2. 在 **Advanced tab**, 选择 **Edit Languages...** 按钮。
3. 在 **Edit Languages** 对话框, 选择对话框底部的 **combo box**。

4. 你将打开一个可用语言的列表，选择你想使用的语言。
5. 选择 **Add** 按钮将你选择的语言添加进显示列表中。
6. 现在你的选择应该出现在 **Edit Languages** 对话框中的列表中。在列表中选择你刚添加的语言，然后选择上移按钮，直到它出现在列表的顶部。
7. 选择 **OK** 直到关闭所有对话框。
8. 刷新你的网页

6 Unicode 支持

Pentaho BI 平台支持 Unicode，成功完成翻译请按照以下指南。

6.1 Properties 文件

这些文件作为 ISO-8859-1 编码的文件读取。

为在这些文件中存储 Unicode 字符，它们必须使用 Java (\uNNNN) 或 XML (&#xNNNN;) 编码，并存储为 ASCII 文件。

你可以使用一个文本编辑器，例如 SC Unipad (<http://www.unipad.org>) 来编辑 Pentaho .properties 文件，并用正确的格式保存它们。

6.2 XML 文件

这些文件中的所有 Unicode 字符必须使用 XML (&#xNNNN;) 编码。

6.3 XSL 文件

XSLs 文件中的 Unicode 字符必须使用 XML (&#xNNNN;) 编码存储。

XSLs 中使用的，存储在 org/Pentaho/ui/messages.properties 中的 messages 的 Unicode 字符应该使用 msg:getXslString 访问，应该关闭输出 escaping。

例如：

```
<xsl:value-of
  select="msg:getXslString('UI.USER_FILTER_PANEL_HINT')"
  disable-output-escaping="yes"
/>
```

6.4 JFreeReport 定义

JFreeReport 中的报表定义的 Unicode 字符应该使用 XML (&#xNNNN;) 编码存储。

而且，报表配置区域应该指定 PDF 输出所使用的 Identity-H 编码和嵌入式字体。

例如：

```
<configuration>
  <property
name="org.jfree.report.modules.output.pageable.pdf.Encoding">Identity-H</property>
```

```
<property name="org.jfree.report.modules.output.pageable.pdf.EmbedFonts">true</property>
</configuration>
```

你需要使用支持 Unicode 字符的字体，例如‘Ariel Unicode MS’。

7 提交你的翻译

如果你想提交任何文件，请发邮件至：communityconnection@pentaho.org。你将需要签署一份 Contributor Agreement (http://www.pentaho.org/contributor_agreement)，然后我们将告知你如何提交。

8 此文档没有覆盖的内容

这个国际化指南并没有覆盖如何翻译 samples 使用的 sample 数据的信息，以后的版本中将会提供。

9 其他资源

在构造 Pentaho BI 平台时，我们一直遵循 Sun 的国际化一个 Java 应用程序的标准，这使得准备任何语言的平台仅仅是翻译的事情，而不需要修改代码。有关 Sun 的如何国际化一个 Java 应用程序的国际标准，推荐阅读以下资料：

- Internationalize Your Software, Jeff Friesen, Java World
<http://www.javaworld.com/javaworld/jw-12-1998/jw-12-internationalize.html>
- Java Internationalization: An Overview, John Connor, Sun Microsystems Inc.
<http://java.sun.com/developer/technicalArticles/Intl/IntlIntro>
- Java Internationalization: Localization with Resource Bundles, John Connor, Sun Microsystems Inc. <http://java.sun.com/developer/technicalArticles/Intl/ResourceBundles>
- Internationalization, O’Rielly ONJava.com, David Flanagan
http://www.onjava.com/pub/a/onjava/excerpt/javaexIAN3_chap8/