1. 更新履歴

- 2003/07/07
  新規作成(第 0.1 版)

2. 基本説明

2.1. 文字エンコーディングフィルタとは...

2.1.1. フィルタとは

フィルタとは、クライアントからのリクエストにフィルタ処理をして、サーバ側(サーブレット)にその要求を渡す機能を指す。このフィルタ機能は、Java Servlet API2.3 から導入された新機能である。

フィルタ機能を使用するには、

(1) フィルタ機能の API インタフェース(javax.servlet.Filter 等)を実装するクラスを作成する。
(2) リクエスト URL 毎に、どのクラスにフィルタ処理を行わせるかを設定する。

という作業を行えば良い。

2.1.2. 文字エンコーディングとは

文字エンコーディングは、それだけで本が書けるほど複雑で分りにくい話題なので、ここでは、非常に簡略化して説明してみる。

基本的に、

- 文字エンコーディングとは、文字列とバイト列との変換方式。
- コンピュータは、バイト列しか認識できない。
- コンピュータが文字列を処理(表示等)する場合、バイト列から文字列に変換する処理が必要。

と考えれば良い。

プログラム言語では「文字列型」というデータ型が存在する。そのため、「コンピュータは文字列を認識できる」と思いがちだが、そうではない。プログラム内部で、「文字列←→バイト列」変換を行っているのだ。しかし、「文字列とバイト列との変換方式」である文字エンコーディングを、誰も指定していないではないか。そう、指定しなくても良いのである。プログラム内部で、(勝手に)自動変換される場合「デフォルトエンコーディング」が使われる。これは OS や OS の言語設定に依存していて、例えば、Windows 日本語版だと「MS932」、Linux で日本語設定がしてあると、「EUC_JP」等となる。なお、このデフォルトエンコーディングが、文字エンコーディング問題をややこしくしている面もあるので、注意が必要である。

要するに、「文字列を扱うプログラムを書く」場合や「文字列を扱うアプリケーションを使う」場合には、文字エンコーディングを必ず指定する事を忘れないようにする事が重要なのである。

2.1.3. なぜ文字エンコーディングフィルタが必要なのか

2.1.2でも説明したように、文字列を扱うアプリケーションを使うには、文字エンコーディングを指定する必要がある。Web アプリケーションは、クライアントから入力された文字列(リクエストのパラメータ)を取得して処理を行うのが基本である。そのため、「クライアントから入力された文字列の文字エンコーディングは何か」が分らないと、当然、文字化けを起こしてしまう(デフォルトエンコーディングのおかげで、運良く文字化けを起こさない場合もある)。

特に、Web アプリケーションは、クライアント環境が様々であり、文字化けの危険性が高いと言える。そのため、クライアント環境に従って、適切な文字エンコーディングを設定しなければならない。そこで、文字エンコーディングフィルタを使って、サーバ側に、クライアントの文字エンコーディングを教えてあげよう、というのである。

フィルタ機能が採用されるまで(Java Servlet API2.2 以前)は、クライアントからのリクエストのパラメータを取得するのに、次のようなコーディングを強いられる場合もあった。

```
.....
    String s = "";
    if ((s = req.getParameter("_paramName_")) != null) {
        try {
            s = new String(s.getBytes("iso-8859-1"), "Shift_JIS");
        } catch (UnsupportedEncodingException e) {
            // エラー処理
        }
    }
.....
```

上記のようなコーディングをしない場合は、サーブレットコンテナに依存した設定が必要な場合が多い。例えば、Tomcat3 では、server.xml に次のように記述すれば、文字エンコーディングを指定できる。

```
.....
        <DecodeInterceptor defaultEncoding="Shift_JIS" />
.....
```

フィルタ機能では、リクエストごとにクライアントの文字エンコーディングを動的に切り替えるような、より柔軟な対応が可能となっている。

2.2. 目的

本ドキュメントは、文字エンコーディングフィルタの実装方法について説明することを目的とする。ここで説明する文字エンコーディングフィルタは、以下の機能を持つ。

- サーバが、クライアントから送信されるリクエストの文字エンコーディングを判断できる。
- 文字エンコーディングは、クライアントからのリクエストごとに動的に指定できる。また、web アプリケーションのデフォルト値(リクエストごとに指定しない場合に使う)も指定できる。

3. 前提環境

本ドキュメントが前提とする動作環境を以下に示す。

- JVM：j2sdk1.4.1
- Servlet コンテナ：Tomcat4.1.27

4. 実装方式

4.1. フィルタの作成

まず、文字エンコーディングフィルタのプログラムを作成する。ここでは、Tomcat3.3.1 と同様の方式を採用する。即ち、

- リクエスト URL に「;charset=***」を埋め込む。
- サーバ側で URL の解析を行う事により、文字エンコーディングを特定する。

という方式を実装する。文字エンコーディングフィルタのソースコードを以下に示す。

```
/*
 * @(#) jp.co.bbreak.jfusion.dv.CharsetFilter
 * Copyright (c) 2003 bBreak Systems Co, Ltd. All Rights Reserved.
 *
 * THE SOFTWARE IS PROVIDED BY bBreak Systems Co, Ltd.,
 * WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
 * NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
 * A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDER BE LIABLE FOR ANY
 * CLAIM, DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING
 * OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */

package jp.co.bbreak.jfusion.dv;

/* for Tomcat4 */
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

// You can set charset daynamically by attribute on request.
// Or, You can set The Default Charset as follow in [web.xml].
//      <filter>
//          <filter-name>Path Mapped Filter</filter-name>
//          <filter-class>jp.co.bbreak.jfusion.dv.CharsetFilter</filter-class>
//          <init-param>
//              <param-name>defaultCharset</param-name>
//              <param-value>shift_jis</param-value>
//          </init-param>
//      </filter>
//
//      <filter-mapping>
//          <filter-name>Path Mapped Filter</filter-name>
//          <url-pattern>/*</url-pattern>
//      </filter-mapping>
public final class CharsetFilter implements Filter {
    public static final String DEFAULT_CHARACTER_ENCODING = "ISO-8859-1";
    private String charsetURIAttribute = ";charset=";
```

```
private String charset = null;
private FilterConfig filterConfig = null;

public void destroy() {
    this.charset = null;
    this.filterConfig = null;
}

public void doFilter(
    ServletRequest request,
    ServletResponse response,
    FilterChain chain)
    throws IOException, ServletException {

    String charEncoding = null;

    // This source is from Tomcat 3.3.1. -
    // Special trick: ;charset= attribute ( similar with sessionId )
    // That's perfect for multibyte chars in URLs
    int idxCharset =
        ((HttpServletRequest)request).getServletPath().indexOf(
            charsetURIAttribute);

    if (idxCharset >= 0) {
        String uri = ((HttpServletRequest)request).getServletPath();
        int nextAtt = uri.indexOf(';', idxCharset + 1);
        String next = null;
        String forwardUrl = null;
        String queryString = ((HttpServletRequest)request).getQueryString();

        if (nextAtt > 0) {
            next = uri.substring(nextAtt);
            charEncoding =
                uri.substring(
                    idxCharset + charsetURIAttribute.length(),
                    nextAtt);
            forwardUrl = uri.substring(0, idxCharset) + next;
        } else {
            charEncoding =
                uri.substring(idxCharset + charsetURIAttribute.length());
            forwardUrl = uri.substring(0, idxCharset);
        }
        if (queryString != null) {
            forwardUrl += "?" + queryString;
        }

        if (charEncoding == null) {
            request.setCharacterEncoding(getDefaultCharset());
        } else if (charEncoding.trim().length() == 0) {
            request.setCharacterEncoding(getDefaultCharset());
        } else {
            request.setCharacterEncoding(charEncoding);
        }

        request.getRequestDispatcher(forwardUrl).forward(request, response);
    }
    // - This source is from Tomcat 3.3.1.

    else {
        request.setCharacterEncoding(getDefaultCharset());
```

```
                chain.doFilter(request, response);
            }

        }

        public void init(FilterConfig filterConfig) throws ServletException {
            this.filterConfig = filterConfig;
        }

        private String getDefaultCharset() throws ServletException {
            String defaultCharset = filterConfig.getInitParameter("defaultCharset");
            if (defaultCharset == null) {
                defaultCharset = DEFAULT_CHARACTER_ENCODING;
            }
            return defaultCharset;
        }
}
```

4.2. フィルタのデプロイ

4.1で作成した文字エンコーディングフィルタを、web アプリケーションからロードできる場所に配備する。具体的には、「$WEBAPP_HOME/WEB-INF/classes/($WEBAPP_HOME/WEB-INF/lib/)」や「$CATALINA_HOME/common/WEB-INF/classes/($CATALINA_HOME/common/WEB-INF/lib/)」等に class ファイル(jar ファイル)を置けばよい。

※　$WEBAPP_HOME：web アプリケーションのベースディレクトリ
※　$CATALINA_HOME：Tomcat のホームディレクトリ

4.3. web.xml の設定

文字エンコーディングフィルタを使用する web アプリケーションの web.xml に、以下の記述を追加する(追加する位置は、参考文献参照)。

なお、「defaultCharset」には、URL に文字エンコーディングの指定がない場合のデフォルト文字エンコーディングを指定すればよい。

```
        <filter>
                <filter-name>Path Mapped Filter</filter-name>
                <filter-class>jp.co.bbreak.jfusion.dv.CharsetFilter</filter-class>
                <init-param>
                        <param-name>defaultCharset</param-name>
                        <param-value>Windows-31J</param-value>
                </init-param>
        </filter>
        <filter-mapping>
                <filter-name>Path Mapped Filter</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>
```

4.4. 使用方法

クライアントへのレスポンスとなる、jsp(または Servlet)が出力する HTML に、URL を記述する際に、予め「charset=***」部分を埋め込んでおけば良い。

Web アプリケーションで、文字エンコーディングが固定でよい場合(クライアントのリクエスト毎に指定する必要がない場合)は、web.xml にデフォルト値を記述するだけで良い。

5. 参考文献

- Apache Jakarta Tomcat3.3.1 のソースコード

- web.xml 要素リファレンス
  http://www.sk-jp.com/java/servlet/webxml.html

- ServletとJSPにおける文字化けについて
  http://www.ingrid.org/java/jserv/i18n/corruptedchar.html