Character Encoding

Character Encoding Issues

Permalink to this page: https://cwiki.apache.org/confluence/x/liklBg

Questions

Answers

Why

What is the default character encoding of the request or response body?

If a character encoding is not specified, the Servlet specification requires that an encoding of ISO-8859-1 is used. The character encoding for the body of an HTTP message (request *or* response) is specified in the `Content-Type` header field. An example of such a header is `Content-Type: text/html; charset=ISO-8859-1` which explicitly states that the default (ISO-8859-1) is being used.

References: HTTP 1.1 Specification, Section 3.7.1

The above general rules apply to Servlets. The behaviour of JSP pages is further specified by the JSP specification. The request character encoding handling is the same, but response character encoding behaves a bit differently. See chapter "JSP.4.2 Response Character Encoding". For JSP pages in standard syntax the default response charset is the usual `ISO-8859-1`, but for the ones in XML syntax it is `UTF-8`.

Why does everything have to be this way?

Everything covered in this page comes down to practical interpretation of a number of specifications. When working with Java servlets, the Java Servlet Specification is the primary reference, but the servlet spec itself relies on older specifications such as HTTP for its foundation. Here are a couple of references before we cover exactly where these items are located in them. A more detailed list can be found on the Specifications page.

1. Java Servlet Specification 4.0
2. HTTP 1.1 Protocol: Message Syntax and Routing, HTTP 1.1 Protocol: Semantics and Content …
3. URI Syntax
4. ARPA Internet Text Messages
5. HTML 4, HTML 5

Default encoding for request and response bodies

See 'Default Encoding for POST' below.

Default encoding for GET

The character set for HTTP query strings (that's the technical term for 'GET parameters') can be found in sections 2 and 2.1 the "URI Syntax" specification. The character set is defined to be US-ASCII. Any character that does not map to US-ASCII must be encoded in some way. Section 2.1 of the URI Syntax specification says that characters outside of US-ASCII must be encoded using `%` escape sequences: each character is encoded as a literal `%` followed by the two hexadecimal codes which indicate its character code. Thus, a (US-ASCII character code 97 = 0x61) is equivalent to `%61`. Although the URI specification does not mandate a default encoding for percent-encoded octets, it recommends UTF-8 especially for new URI schemes, and most modern user agents have settled on UTF-8 for percent-encoding URI characters.

Some notes about the character encoding of URIs:

1. ISO-8859-1 and ASCII are compatible for character codes 0x20 to 0x7E, so they are often used interchangeably.
2. Modern browsers encoding URIs using UTF-8. Some browsers appear to use the encoding of the current page to encode URIs for links.
3. HTML 4.0 recommends the use of UTF-8 to encode the query string.
4. When in doubt, use POST for any data you think might have problems surviving a trip through the query string.

Default Encoding for POST

Older versions of the HTTP/1.1 specification (e.g. RFC 2616) indicated that ISO-8859-1 is the default charset for text-based HTTP request and response bodies if no charset is indicated. Although RFC 7231 removed this default, the servlet specification continues to follow suit. Thus the servlet specification indicates that if a `POST` request does not indicate an encoding, it must be processed as `ISO-8859-1`, except for `application/x-www-form-urlencoded`, which by default should be interpreted as {{`}}US-ASCII` (as it by definition should contain only characters within the ASCII range to begin with).

Some notes about the character encoding of a POST request:

1. RFC 2616 Section 3.4.1 stated that recipients of an HTTP message *must* respect the character encoding specified by the sender in the `Content-Type` header if the encoding is supported. A missing character allows the recipient to "guess" what encoding is appropriate.
2. Most web browsers today *do not* specify the character set of a request, even when it is something other than ISO-8859-1. This seems to be in violation of the HTTP specification. Most web browsers appear to send a request body using the encoding of the page used to generate the POST (for instance, the <form> element came from a page with a specific encoding... it is *that* encoding which is used to submit the POST data for that form).

Percent Encoding for `application/x-www-form-urlencoded`

The HTML 4.01 specification indicated that percent-encoding of any non alphanumeric characters of `application/x-www-form-urlencoded` (the default content type for HTML form submissions) should be performed using `US-ASCII` byte sequences. However HTML 5 changed this to use UTF-8 byte sequences, matching the modern percent encoding for URLs. Modern browsers therefore percent-encode UTF-8 sequences when submitting forms using `application/x-www-form-urlencoded`.

The servlet specification, however, requires servlet containers to interpret percent-encoded sequences in `application/x-www-form-urlencoded` as `ISO-8859-1`, which in a default configuration will result in corrupted content because of the charset mismatch. See below for how this can be reconfigured in Tomcat.

HTTP Headers

Section 3.1 of the ARPA Internet Text Messages spec states that headers are always in US-ASCII encoding. Anything outside of that needs to be encoded. See the section above regarding query strings in URIs.

How

How do I change how GET parameters are interpreted?

Tomcat will use ISO-8859-1 as the default character encoding of the entire URL, including the query string ("GET parameters") (though see Tomcat 8 notice below).

There are two ways to specify how GET parameters are interpreted:

1. Set the `URIEncoding` attribute on the <Connector> element in server.xml to something specific (e.g. `URIEncoding="UTF-8"`).
2. Set the `useBodyEncodingForURI` attribute on the <Connector> element in server.xml to `true`. This will cause the Connector to use the request body's encoding for GET parameters.

In Tomcat 8 starting with 8.0.0 (8.0.0-RC3, to be specific), the default value of `URIEncoding` attribute on the <Connector> element depends on "strict servlet compliance" setting. The default value (strict compliance is off) of `URIEncoding` is now `UTF-8`. If "strict servlet compliance" is enabled, the default value is `ISO-8859-1`.

References: Tomcat 7 HTTP Connector, Tomcat 7 AJP Connector, Tomcat 8.5 HTTP Connector, Tomcat 8.5 AJP Connector

How do I change how POST parameters are interpreted?

`POST` requests should specify the encoding of the parameters and values they send. Since many clients fail to set an explicit encoding, the default used is `US-ASCII` for `application/x-www-form-urlencoded` and `ISO-8859-1` for all other content types.

In addition, the servlet specification requires that percent-encoded sequences of `application/x-www-form-urlencoded` be interpreted as `ISO-8859-1` by default which, as explained above, does not match the HTML 5 specification and modern user agent practice of using UTF-8 to percent encode characters. Nevertheless the servlet specification requires the servlet container's interpretation of percent-encoded sequences of `application/x-www-form-urlencoded` to follow any configured character encoding. Thus appropriate intepretation of `application/x-www-form-urlencoded` byte sequences can be achieved by setting the request character encoding to `UTF-8`.

The container-agnostic approach for specifying the request character encoding for applications using Servlet 4.0 or later (which would correspond to Tomcat 9.0 and later) is to set the `<request-character-encoding>` element in the web application `web.xml` file:

```
<request-character-encoding>UTF-8</request-character-encoding>
```

Note: If you are using the Eclipse integrated development environment, as of Eclipse Enterprise Java Developers 2019-03 M1 (4.11.0 M1) the IDE does not recognize the `<request-character-encoding>` setting and will temporarily freeze the IDE and generate errors with any edit of web application files. You can track the latest status of this problem at Eclipse Bug 543377.

Otherwise one can employ a `javax.servlet.Filter`. Writing such a filter is trivial.

6.x, 7.x::
Tomcat already comes with such an example filter. Please take a look at `webapps/examples/WEB-INF/classes/filters/SetCharacterEncodingFilter.java`.

5.5.36+, 6.0.36+, 7.0.20+, 8.x and later::
Since Tomcat 7.0.20, 6.0.36 and 5.5.36 the filter became first-class citizen and was moved from the examples into core Tomcat and is available to any web application without the need to compile and bundle it separately, although this will not allow the web application to be deployed in non-Tomcat servlet containers that do not have this filter available, if the servlet is defined in the web application's own `web-xml` file. See documentation for the list of filters provided by Tomcat. The class name is `org.apache.catalina.filters.SetCharacterEncodingFilter`.

It is also possible to define such a filter in the Tomcat installation configuration file `conf/web.xml`, which would set the request character encoding across all web applications without the need for any `web.xml` modifications. In fact the latest Tomcat versions come with sections in `conf/web.xml` that already configure a filter to set the request character encoding to `UTF-8`. Simply edit `conf/web.xml` and uncomment both the definition and the mapping of the filter named `setCharacterEncodingFilter`.

Note: The request encoding setting is effective only if it is done earlier than parameters are parsed. Once parsing happens, there is no way back. Parameters parsing is triggered by the first method that asks for parameter name or value. Make sure that the filter is positioned before any other filters that ask for request parameters. The positioning depends on the order of `filter-mapping` declarations in the WEB-INF/web.xml file, though since Servlet 3.0 specification there are additional options to control the order. To check the actual order you can throw an Exception from your page and check its stack trace for filter names.

Tomcat 9.x and later: do not use a `<filter>` at all and instead specify `<request-character-encoding>` in your application's web.xml file.

What can you recommend to just make everything work? (How to use UTF-8 everywhere).

Using `UTF-8` as your character encoding for everything is a safe bet. This should work for pretty much every situation.

In order to completely switch to using UTF-8, you need to make the following changes:

1. Set `URIEncoding="UTF-8"` on your <Connector> in `server.xml`. References: Tomcat 7 HTTP Connector, Tomcat 7 AJP Connector, Tomcat 8.5 HTTP Connector, Tomcat 8.5 AJP Connector.
2. Set the default request character encoding either in the Tomcat `conf/web.xml` file or in the web app `web.xml` file; either by setting `<request-character-encoding>` (for applications using Servlet 4.0 / Tomcat 9.x+) or by using a character encoding filter.
3. Change all your JSPs to include charset name in their contentType. For example, use `<%@page contentType="text/html; charset=UTF-8" %>` for the usual JSP pages and `<jsp:directive.page contentType="text/html; charset=UTF-8" />` for the pages in XML syntax (aka JSP Documents).
4. Change all your servlets to set the content type for responses and to include charset name in the content type to be UTF-8. Use `response.setContentType("text/html; charset=UTF-8")` or `response.setCharacterEncoding("UTF-8")`.
5. Change any content-generation libraries you use (Velocity, Freemarker, etc.) to use UTF-8 and to specify UTF-8 in the content type of the responses that they generate.
6. Disable any valves or filters that may read request parameters before your character encoding filter or jsp page has a chance to set the encoding to UTF-8. For more information see https://www.mail-archive.com/users@tomcat.apache.org/msg21117.html.

How can I test if my configuration will work correctly?

The following sample JSP should work on a clean Tomcat install for any input. If you set the URIEncoding="UTF-8" on the connector, it will also work with method="GET".

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
     <title>Character encoding test page</title>
   </head>
   <body>
     <p>Data posted to this form was:
     <%
       request.setCharacterEncoding("UTF-8");
       out.print(request.getParameter("mydata"));
     %>

     </p>
     <form method="POST" action="index.jsp">
       <input type="text" name="mydata">
       <input type="submit" value="Submit" />
       <input type="reset" value="Reset" />
     </form>
   </body>
</html>
```

How can I send higher characters in my HTTP headers?

You have to encode them in some way before you insert them into a header. Using url-encoding (% + high byte number + low byte number) would be a good idea.

How to configure the BASIC authentication scheme to use UTF-8

If a web application is configured to use the BASIC authentication scheme (e.g. configured with `<auth-method>BASIC</auth-method>` in its web.xml file), it means that an instance of **BasicAuthenticator** will be automatically created and inserted into the chain of Valves for this web application (this Context), unless any other Authenticator valve has already been explicitly configured.

To enable support for UTF-8 in a BasicAuthenticator, you can configure it explicitly, by inserting the following line into the Context configuration file of your web application (usually META-INF/context.xml):

```
<Valve className="org.apache.catalina.authenticator.BasicAuthenticator" charset="UTF-8" />
```

If you do so, the BasicAuthenticator will append "charset=UTF-8" to the value of WWW-Authenticate header that it sends and will interpret the values sent by clients as UTF-8.

See also:

- Configuration Reference (Tomcat 9): Valves (Authentication), Context (Defining a context).
- Bug 61280
- Bug 66174
- Specifications (RFC 7617)

Troubleshooting

I'm having a problem with character encoding in Tomcat 5

In Tomcat 5 - there have been issues reported with respect to character encoding (usually of the the form "request.setCharacterEncoding(String) doesn't work"). Odds are, its not a bug. Before filing a bug report, see these bug reports as well as any bug reports linked to these bug reports:

- 23929
- 25360
- 25231
- 25235
- 22666
- 24557
- 24345
- 25848