Understanding Encoding

Different languages require different characters. Classic character encodings were limited to 255 characters per *character set* to reduce the amount of disk space required to store text. The original character set, ASCII, was limited to 128 characters and was suitable for English.

The Unicode effort and the UTF-8 encoding were created to support all characters for all languages as efficiently as possible.

Tiki uses UTF-8 internally for all manipulations and has done so for a very long time.

Encoding scenarios

However, MySQL allows storage to use different encodings for more efficiency. In the past, Tiki poorly informed MySQL of the content it provided to it. Several outcomes could appear.

The UTF-8 case

In this ideal scenario, the MySQL database server was configured to accept UTF-8 encoding and store the data as UTF-8. All features behaved correctly and data could freely be accessed from other applications. Some distributions provided settings leading to this configuration, but in most cases, it required intervention from the system administrator.
Sadly, what MySQL calls "utf8" does not cover the full range of UTF-8 encodings. Some languages are missing, as well as `emojis`. In order to support all possible characters, Tiki moved to `utf8mb4` since Tiki19. Since MySQL charset `utf8` is a subset of `utf8mb4` and not a different encoding, the data which is was stored in utf8 is still correctly encoded.

The latin case

In many cases, MySQL was configured to receive and store content as *latin1*, and encoding suitable for most western Europe languages. Under this set-up, Tiki behaved correctly from the front-end. However, there were consequences:

- Extracting the data stored by Tiki from other applications, such as phpMyAdmin, lead to strangely encoded characters.
- Moving the database to a different server could cause unexpected errors.
- Ordering in lists could appear to be random.

Double encoding

On most shared hosts, site administrators have no control over the default connection encoding on the server. The connection encoding would be used as latin1. However, when creating the database, the site administrator would create the database as UTF-8 to support all languages.

This configuration had a pervert effect. When Tiki would send UTF-8 data, MySQL would think it was latin1. Because the database requested UTF-8, it would perform a conversion, leading to doubly-encoded UTF-8.

In reality, the consequences were very similar to the latin case.

Information loss

If the server was configured by default to accept UTF-8 connections and the database happened to be created as latin1, Tiki would correctly send UTF-8 data and MySQL would recognize it. However, when time arrived to store it, the data would be converted to latin1. Because UTF-8 has more possible characters than other encoding types, unknown characters got converted to ? signs or some other unknown character.

Other than not supporting languages not supported by the database encoding, all features would work correctly in Tiki.

Changes

After a partially-failed attempt to resolve these issues in Tiki5, more options were added in Tiki5.1.

This section explains how the upgrade to 5.1 will affect your installation.

Upgrading from 2.x, 3.x or 4.x

For previous versions, Tiki will keep behaving as-is to avoid corrupting data.

Correcting the encoding issue will require manual intervention. The Tiki installer comes with tools to assist in the conversion. However, **you should have reliable backups before using them**. Other techniques involving exporting data, modifying the dump and re-importing it can be used as well.

To avoid data loss, you should make sure your tables use UTF-8. This can be done from the install/upgrade page in the installer.

Removing the double-encoding can be done from *Enter Your Tiki*. It will require the client charset to be forced to UTF-8.

Follow the steps described here:
http://doc.tiki.org/Upgrade#Fix_the_encoding_issue

Upgrading from 5.0

In default 5.0 installations and upgrades, the connection character set was specified automatically in the configuration file.

New 5.0 installations correctly specified the connection encoding and will lead to either the UTF-8 case or the information loss case.

Upgraded 5.0 installations will lead to different results. However, the upgrade procedure to 5.1 does not impact the changes that were made at that time. Some administrator judgment is advised if encoding issues appear. The client character set can be altered in the configuration file. Contact the developer mailing list if support is required.

New 5.1 installations

New 5.1 installations will specify the connection encoding to UTF-8 by default. If the database is not in UTF-8, this will lead to information loss. The installer will propose that you convert your database to UTF-8.

If you have no control over the database encoding and still need to use non-latin characters (ex.: Arabic) , it is best not to force the connection encoding to UTF-8 or to set it as the same encoding as the database itself. It will lead to the latin case, which comes with certain downfalls, but still allow for multiple languages to be used on a single site.

If you can control your database encoding

Pre-Tiki6, make sure to set your MySQL (via PhpMyAdmin for example) so that new tables are also created in UTF-8. In general, for your databases, look for "MySQL connection collation:" and if it's at latin_1, change it to utf8_general_ci. (Generally, this is already the case in typical shared hosting)

Then, in PhpMyAdmin, go to the database you'll be using your Tiki in Server: localhost -> Database: tiki -> Operations -> Collation, and also change from latin_1 to utf8_general_ci before running the installer. (Generally, this needs to be changed in typical shared hosting)

This is the equivalent of the following MySQL statement:

```
ALTER DATABASE DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Tiki 5.2 or 5.3

If you have file upload issues (just upload an image and you will see if it works or not), you can solve by

- applying revision 30522, which fixes it (and will be 5.4)

If you can't do that, you can try to

- add $api_tiki='adodb'; to your db/local.php,

and/or not forcing utf8

In db/local.php

```
$client_charset='utf8';
```

Now commented out

```
// $client_charset='utf8';
```

Tiki 5.4

On a fresh install, Tiki5.4 will by default set the tables to be UTF-8 and commit 30522 will be present.

Tiki 6

On a fresh install, Tiki6 will by default set the tables to be UTF-8.

Tiki 19

On a fresh install, Tiki19 will by default set the MySQL/MariaDB tables to utf8mb4 encoding to be fully UTF-8 compatible (including the Emojis support).

Old Tikis which are upgraded to Tiki19 will automatically change the table encodings from `utf8` to `utf8mb4` when you run the database upgrade scripts (either using the online installer or the usual command line `php console.php database:upgrade`).

Step by step procedure to fix the latin1 ⇒ UTF-8 encoding issues

See http://doc.tiki.org/Upgrade#Fix_the_encoding_issue

References

The following discusses the encoding issues in depth and describes how to successfully fix the issue after importing data into a new utf8 encoded database:
Getting out of MySQL Character Set Hell