Example for changing characters in the TtM output of a LaTeX file

I use the example of the file https://www.rb-vps.de/?page_id=762 shown on my Website. **This is an ADHOC solution only** to cover the omitted calligraphic xml-symbols, which were not supported by browsers, when TtM was written many years ago. You can also include graphics (see the TtM manual for that).

Before TtM

I added in the LaTeX-File environments for \begin{theorem}, \begin{proof} and analogously for remarks and examples. Then the HowTo was as follows:

1) Find in the texfile all strings "**\mathcal{something}**" and delete \ , { and } in those strings.
   It is assumed that all calligraphic characters are edited **in that form in LaTeX**.
2) Analogously delete \ , { and } in all strings "**\mathbb{x}**" (in my case only x=R oder x=C).
   As above that editing \mathbb{x} is assumed in LaTeX.
   Generate a list1 with the new changed strings. I did this with the following bash script

change-texfile-and-list.sh / **call for example:** bash change-texfile-and-list.sh "your_file.tex" "**list1**"

```
#!/bin/bash

# Check if a filename is provided
if [ -z "$1" ]; then
    echo "Usage: $0 filename list_filename"
    exit 1
fi
# Check if the list filename is provided
if [ -z "$2" ]; then
    echo "Usage: $0 filename list_filename"
    exit 1
fi
# Assign the parameters to variables
filename="$1"
list_filename="$2"

# Create a temporary file to store new patterns
temp_file=$(mktemp)
# Perform the substitution using sed and capture the new patterns: replace \mathcal{..}
sed -i.bak -E 's/\\mathcal\{([^}]+)\}/ mathcal\1/g' "$filename"
# Extract new patterns from the modified file
grep -o 'mathcal[a-zA-Z]\+' "$filename" | sort | uniq > "$temp_file"
# Append new patterns to the list if not already present
while IFS= read -r line; do
    if ! grep -Fxq "$line" "$list_filename"; then
        echo "$line" >> "$list_filename"
    fi
done < "$temp_file"
# Clean up temporary file
rm "$temp_file"

# Create a temporary file to store new patterns
temp_file=$(mktemp)
# Perform the substitution using sed and capture the new patterns: replace \mathbb{..}
sed -i.bak -E 's/\\mathbb\{([^}]+)\}/ mathbb\1/g' "$filename"
# Extract new patterns from the modified file
grep -o 'mathbb[a-zA-Z]\+' "$filename" | sort | uniq > "$temp_file"
# Append new patterns to the list if not already present
while IFS= read -r line; do
    if ! grep -Fxq "$line" "$list_filename"; then
        echo "$line" >> "$list_filename"
    fi
done < "$temp_file"
# Clean up temporary file
rm "$temp_file"
```

3) Call TtM: **ttm your_file.tex**. It's assumed that all necessary files are in the same directory, inclusively **your_file.aux**. The modified strings are then clearly identifiable and can be changed to the desired calligraphic xml characters. This is done as follows:

AFTER TtM

4) Manually edit a new file "list2"

TtM codes the patterns mathL (originally \mathcal{L} in my example) (or other mathx) as
<mi fontstyle="italic">item of list1</mi> (see above);
Now generate list2 with pairwise lines of the type
<mi fontstyle="italic">item1 of list1</mi>
<mi>xml-character for item1</mi> **(see the xml alphabets)**
etc. for all items in list1. In most cases there are not very much of those.

5) Now replace in the TtM output your_file.xml the patterns in the odd numbered lines of list2 by the patterns in the according even numbered lines just below them.
This is done by the following python script

replace-strings.py / **call: python3 replace-strings.py** (Change to your filenames or parameterize them)

```
# Define the file paths
target_file_path = "your_file.xml"
list_file_path = "list2"

# Read the list file and parse it into pairs
with open(list_file_path, 'r') as list_file:
lines = list_file.readlines()

# Ensure there are an even number of lines in the list file
if len(lines) % 2 != 0:
raise ValueError("The 'list' file must contain an even number of lines.")

# Create a list of tuples (find_str, replace_str)
replacement_pairs = [(lines[i].strip(), lines[i+1].strip()) for i in range(0, len(lines), 2)]

# Read the target file
with open(target_file_path, 'r') as target_file:
target_content = target_file.read()

# Perform the replacements
for find_str, replace_str in replacement_pairs:
target_content = target_content.replace(find_str, replace_str)

# Write the modified content back to the target file
with open(target_file_path, 'w') as target_file:
target_file.write(target_content)

print("Replacements complete.")
```

That's it, you then have **your_file.xml** with the desired calligraphic xml characters.
When you edit it additionally with further HTML code, rename it to **your_file.html**,
to avoid xml parsing errors.