

- 1 Unicode-Unterstützung: Bisher war die erste Zeile von capella-Skripten in der Regel

```
# -*- coding: ISO-8859-1 -*-
```

Wenn Unicode genutzt werden soll, muss für capella 7 diese Zeile jetzt ersetzt werden durch

```
# -*- coding: UTF-8 -*-
```

Diese Zeile ist nötig nur für solche Skripten, die in ihrem Quelltext Texte mit Zeichen aus nicht westeuropäischen Zeichensätzen, enthalten: Kyrillisch, griechisch, etc., etwa für Dialogboxen. Wenn ausschließlich Texte eines bestimmten Sprachraums verarbeitet werden sollen, ist es auch möglich (aber nicht empfohlen), stattdessen die entsprechende Codierung anzugeben (z.B. ISO-8859-5 für kyrillisch). Diese Änderung ist nicht nur eine reine Textänderung, sondern erfordert auch eine tatsächliche Umcodierung. Wenn man sie mit dem Python-Editor Idle durchführt, wird die Umcodierung beim Speichern automatisch durchgeführt. Wenn man mit Windows Notepad arbeitet, muss man nach der Textänderung die Datei speichern mit Codierung UTF-8 (statt ANSI).

Falls die Python-Datei ein XML-Fragment mit <info>-Element enthält, das für die sprachabhängige Beschreibung im Script-Browser (Plugin – Script...) genutzt wird, muss genau so in der zugehörigen xml-Deklaration die Codierung umgesetzt werden. Das gleiche gilt für die separaten *.info-Dateien:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Diese Änderung ist leider nötig, weil capella 7 ja beliebige Texte (deutsche Umlaute auch mit eingestreuten griechischen oder kyrillischen Zeichen) verarbeiten kann, diese jedoch in ISO-8859-1 nicht dargestellt werden können, und außerdem dem codierten Text nachträglich nicht mehr sicher anzusehen ist, nach welcher Regel er codiert wurde. Python bietet zwar auch die Möglichkeit, über die Funktionen encode und decode mit den Codec-Bezeichnern Latin-1 (Synonym iso-8859-1) und utf-8 beliebige Umwandlungen programmgesteuert durchzuführen, das würde aber einen Eingriff ins Innere jedes Skriptes bedeuten.

- 2 Eine etwas unangenehme Konsequenz der Umstellung besteht darin, dass die Verwendung des Hilfsprogramms latin1, das in capDOM.py definiert ist, obsolet ist. Diese Funktion wird i.d.R. in externen Skripten verwendet in Zusammenhang mit den Basisklassen ScoreChange oder ScoreExport unter der Annahme, dass eine capella-Partitur in iso-8859-1 codiert ist. Diese Annahme ist nun hinfällig. Zum Ersatz enthalten ScoreChange und ScoreExport jetzt eine Funktion encode(), die die Kodierung der aktuellen Partitur kennt und dieselbe Kodierung dann wieder benutzt. In der Regel sollte es ausreichen, latin1 durch self.encode zu ersetzen, damit die Skripten wieder korrekte Umlaute anzeigen. Diese Funktion wird es (über einen Ersatz von capDOM.py) demnächst auch für ältere capella-Versionen geben können, so dass in dieser Hinsicht eine gewisse Rückwärtskompatibilität erreicht werden kann.
- 3 Externe Steuerung des Displays aus Skript (für Fußschalter z.B. für MusicPad von Klemm oder für frei programmierte Suchfunktionen): setSelection(sel), displayPage(iPage), displaySystem(iSystem). Die letzten beiden können mit displayTest.py ausprobiert werden.

- 4 Eine Editbox aus dem Python-Dialogbaukasten liefert jetzt nicht nur über `value()` den Textinhalt zurück, sondern auch über `sel()` die Selektion: ein Tupel mit Anfangs- und Endindex der Markierung.
- 5 Genau so kann im Edit-Konstruktor mit `"sel"=(s0, s1)` eine Selektion vorgegeben werden. Das geht natürlich nur für ein einziges Edit-Fenster, das dann den Input-Fokus kriegt, bzw. für das letzte Edit-Fenster, das vor dem Rücksprung aus der Dialogbox den Fokus hatte. Test mit `braille-view-newWindow.py`. Auch `VorspielAuswahl.py` ist in diesem Zusammenhang zu sehen.
- 6 Innerhalb Python die neue Klasse `CapXNode` mit den Funktionen `isEmpty()`, `getName()`, `getAttributes()`, `getChildNodeI(index)`, `getChildNode(name, index)`, `getParentNode()`. Die neue global verfügbare Funktion `rootNode()`, die den Wurzelknoten „score“ der aktiven capella-Datei als `CapXNode` zurückliefert. Weitere Funktionen zur Modifikation des XML-Datenbaums werden im Verlauf des Betatests noch eingeführt werden. Test mit `XMLTest.py`
- 7 Externe Skripten laufen jetzt auch in der Demoversion, solange die Partitur nicht verändert wurde (d.h. sie können fertige Beispielpartituren genau einmal verändern).

April 2010