Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python

von Victor Westrich und Yannick Weber

Inhalt

Einführung	67
Die Daten im XML-Format	69
Aufbau der REST-Schnittstelle	71
Voraussetzungen für die Benutzung	72
Beschreibung der Benutzung der Skripte	75
Beschreibung der Funktionsweise der Skripte	79
Skripte	87

Einführung

Die sich inzwischen auch in den Geistes- und Kulturwissenschaften verbreitende freie Lizenzierung von Forschungsdaten eröffnet der Wissenschaft theoretisch die Möglichkeit der Nachnutzung derselben für andere Projekte. Praktisch steht der einzelne Forscher oftmals vor der Hürde, überhaupt an die Daten zu kommen, da oft einfache Zugangswege zu den Daten in geeigneten Formaten fehlen und so Anfragen an die datenhaltende Institution nötig sind, die eine mehr oder minder schwere Zugangsschranke darstellen können und so den Sinn der freien Lizenzierung konterkarieren. Im Folgenden soll am Beispiel des Akademieprojekts *Regesta Imperii (Quellen zur Reichsgeschichte)*[1] ein Weg aufgezeigt werden, diese Hürde zu überwinden.

[1] Regesta Imperii, Akademie Wissenschaft und Literatur Mainz, online verfügbar unter http://www.regesta-imperii.de/startseite.html/.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Ziel des Projekts ist es, sämtliche urkundlich und historiographisch belegten Aktivitäten der römisch-deutschen Könige und Kaiser von den Karolingern bis zu Maximilian I. (ca. 751-1519) sowie der Päpste des frühen und hohen Mittelalters in Form deutschsprachiger Regesten, also einer standardisierten wissenschaftlichen Aufarbeitung, zu verzeichnen. In dieser Unternehmung, angesiedelt insbesondere an der Mainzer Akademie der Wissenschaften sowie an der Berlin-Brandenburgischen Akademie und der Österreichischen Akademie, sind bisher über 90 gedruckte Bände erschienen sowie 135.000 Regesten zu den Aktivitäten erstellt worden, die online auf *Regesta Imperii Online*[2] zur Verfügung stehen.

Außerdem hat RI Online 2015 begonnen im Rahmen von RIplus[3] auch Regesten zu weiteren Fürsten des Reichs sowie work-in-progress Sammlungen zu noch nicht bearbeiteten Herrschern aufzubereiten, sodass die Datenbank nun über 180.000 Datensätze bereitstellt. Damit bietet sie einen gewaltigen Schatz an Daten, der enorme Potenziale für die Erschließung mittels digitaler Methoden und Verfahrensweisen bietet[4].

Um eine solche Erschließung zu ermöglichen, die sich nicht allein auf die Abfragemaske der Onlinerepräsentation stützen kann, wurden seit 2015 drei Maßnahmen ergriffen: Die Daten wurden unter eine CC BY 4.0[5] Lizenz gestellt. Damit dürfen sie zu jedem Zweck genutzt, bearbeitet, geteilt und veröffentlicht werden, solange der Urheber und die Lizenzierung angemessen und unverändert angegeben werden. Zweitens wurden die Daten für die

[2] Weitere Informationen zum Inhalt der Regestendatenbank finden sich online unter:http://www.regesta-imperii.de/unternehmen/ri-online.html/; s. Julian Schulz: Rezension zur Regesta Imperii Online, in RIDE 6 (2017), online publiziert unter http://ride.i-d-e.de/issues/issue-6/regesta-imperii-online/.

[3] Yannick Weber, Regesta Imperii plus. In: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte. 2016, online publiziert unter https://mittelalter.hypotheses.org/7911..

[4] Hier ist beispielsweise zu nennnen der Beitrag der Computerlinguisten Juri Opitz und Annete Frank zum Deep Learning mithilfe der RI-Daten, oder Joachim Lacznys Beitrag zur Itinerarforschung über Kaiser Friedrich III. In diesem vernetzte Lacny Daten der Regesta Imperii zu Ausstellungsorten Friedrichs III. mithilfe eines HGIS, eines Historischen Geoinformationssystems, vgl. Anette Frank und Juri Opitz, Deriving Players & Themes in the Regesta Imperii using SVMs and Neural Networks, in: Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH), hrsg. v. Association for Computational Linguistics, Berlin 2016, online verfügbar unter https://aclweb.org/anthology/W/W16/W16-2108.pdf, S. 74-83; Joachim Laczny, Friedrich III. (1440-1493) auf Reisen, in: Perzeption und Rezeption, Wahrnehmung und Deutung im Mittelalter und in der Moderne, hrsg. v. Joachim Laczny und Jürgen Sarnowsky, Göttingen 2014, S. 33-65.

[5] Creative Commons, CC BY 4.0, siehe https://creativecommons.org/licenses/by/4.0.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Transformation in das CEI-XML Schema[6] überführt und über eine REST-Schnittstelle[7] zur Verfügung gestellt, die allerdings nicht ohne vertiefte technische Kenntnisse genutzt werden kann.

Ziel des vorliegenden Guides ist es ohne großes technisches Vorwissen mithilfe eines Pythonskriptes ausgewählte Regesten herunterzuladen und am Ende einerseits die gewünschten Regesten als separate XML-Dateien zur Verfügung zu haben und andererseits bestimmte Informationen aus den einzelnen Regestendateien in einer CSV-Datei zusammenzufassen.

Die Daten im XML-Format

Neben dem reinen Regestentext bietet RI Online eine Reihe von Metadaten an, die teilweise über das analog und sogar das digital publizierte Regest hinausgehen. So sind für das Ausstellungsdatum bzw. den -zeitraum von Regesten Normdatierungen, für hoch frequentierte Ausstellungsorte Geokoordinaten, sowie Daten zu Literaturverweisen, Online-Drucken und Abbildungen der zugrundeliegenden Urkunde integriert. Alle diese Informationen sind in der XML-Datei des jeweiligen Regests hinterlegt. XML („Extensible Markup Language“)[8] ist, ähnlich wie HTML, eine Textauszeichnungssprache. Genauso wie HTML („Hyper Text Markup Language“)[9] verändert XML die Textgrundlage nicht. Im Gegensatz zu HTML, das vor allem zur Anzeige des Textes in einer bestimmten Form dient, liegt der Fokus bei XML darauf, relevante Informationen aus dem Text zu zu ordnen und maschinenlesbar zu machen.

Im Gegensatz zu anderen Auszeichnungssprachen liegt der Vorteil von XML darin, dass der Nutzer selbst entscheiden kann, was er als relevante Informationen sieht und wie er diese auszeichnen möchte. Voraussetzung einer funktionierenden XML-Datei ist lediglich, dass das Dokument *well formed* ist, das heißt, dass die Auszeichnung korrekt und kohärent erfolgt. Um eine Standardisierung verschiedener Dateien und so die notwendige Dokumentation für die

[6] CEI – Charters Encoding Initiative, online verfügbar unter http://www.cei.uni-muenchen.de/index.php.
[7] Siehe auch: Datenschnittstellen der Regesta Imperii – REST, siehe http://www.regesta-imperii.de/daten.html/.
[8] Siehe online https://www.w3.org/XML/.
[9] Siehe online https://www.w3.org/html/.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Nachnutzung zu gewährleisten, wird ein XML-Schema mit der jeweiligen Definition aller erfolgten Auszeichnungen bereitgestellt. RI Online verwenden hierzu das CEI-Schema. Das CEI-Schema entsprang 2004 dem Versuch, einen Standard zur Textcodierung speziell mittelalterlicher und frühneuzeitlicher Texte zu schaffen. Die CEI-Guidelines orientieren sich an denen der TEI. Der von der TEI („Text Encoding Initative“)[10] empfohlene Satz an XML-Elementen stellt einen führenden Standard[11] in den „Digital Humanities“ zur Textcodierung dar. Hierbei liegt der Fokus bei der Überführung bereits bestehender Texte in ein digitales Format und deckt sich damit mit den Aufgaben von RI Online. Der Header mit den Metadaten einer XML-Datei eines Regests der RI Online im CEI-Format sieht beispielsweise so aus:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <cei xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.cei.lmu.de/schema/cei060122.xsd">
        <teiHeader>
            <fileDesc>
                <titleStmt>
                    <title>Friedrich III. – [RI XIII] H. 14 n. 7</title>
                </titleStmt>
                <editionStmt>
                    <p n="volume">[RI XIII] H. 14 - Friedrich III., Nürnberg 1
(1440-1449)</p>
                    <p n="repository">Regesta Imperii Online: <ref
type="external" target="http://www.regesta-imperii.de/cei/013-014-
000/sources/1440-05-16_1_0_13_14_0_7_7"></ref></p>
                </editionStmt>
                <publicationStmt>
                    <p n="authority">
                        Deutsche Kommission für die Bearbeitung der Regesta
Imperii e.V. bei der Akademie der Wissenschaften und der Literatur | Mainz
                    </p>
[...]
```

Unter *http://www.cei.lmu.de/schema/cei060122.xsd* ist das referenzierte CEI-Schema zu finden, das den Aufbau der Datei beschreibt und die korrekte Anwendung des Schemas im konkreten Regestendatensatz kontrolliert.

[10] TEI Consortium, TEI P5: Guidelines for Electronic Text Encoding and Interchange. Version 3.0.0. 2016, online verfügbar unter http://www.tei-c.org/index.xml.
[11] Patrick Sahle: Digitale Editionsformen. Zum Umgang mit der Überlieferung unter den Bedingungen des Medienwandels. Teil 3: Textbegriffe und Recodierung (Schriften des Instituts für Dokumentologie und Editorik 9), Norderstedt 2013, S. 341-343, online verfügbar unter http://kups.ub.uni-koeln.de/5353/.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Man spricht beim Aufbau von XML-Dateien von einer Baumstruktur, die sich vom root-Element, dem Ausgangspunkt des Pfades, ausgehend, in diesem Falle „TEI“, immer weiter verästelt. Ein relevantes Konzept für das Skript stellt XPath („XML Path Language“) dar. XPath beschreibt den Ort eines Elements innerhalb der XML-Struktur anhand des dorthin führenden Pfades. Am Beispiel des obigen XML-Regests führt folgender XPath zum Inhalt des title-Elements:

teiHeader/fileDesc/titleStmt/title

Mithilfe dieses XPath könnte man sich „Friedrich III. [RI XIII] H. 14 n. 7“ auslesen lassen. Um auf andere Elemente zuzugreifen, als unten vorgesehen, muss der Pfad jeweils entsprechend angepasst werden.

Analog dazu wird die Struktur des ganzen XML-Dokuments als Baumstruktur bezeichnet, da sie, in ihren Verzweigungen vom root-Element hinweg, dieser ähnelt. Auf XPath wird später im Rahmen der Beschreibung der Funktionsweise des DownloadSkripts Bezug genommen.[12]

Aufbau der REST-Schnittstelle

Das REST („Representational State Transfer“)-Programmierparadigma fordert, dass jede online zur Verfügung gestellte Ressource eindeutig adressiert und somit auch von einer Maschine (= Computer) ohne Verwechslungsgefahr mit anderen Ressourcen eingelesen werden kann. Am Beispiel der *Regesta Imperii Online* heißt das: Jedes einzelne Regest kann mithilfe einer URI („Uniform Resource Identifier“), einem eindeutigen Identifikator einer digitalen Ressource, angesprochen werden.

Über die REST-Schnittstelle steht eine vollständige Liste aller Regesten-URIs, gebündelt nach Kollektionen, zur Verfügung. Diese Kollektionen sind zusammengestellt nach der herrscher- oder provenienzbezogenen Bandeinteilung der Regesta Imperii. So stehen zum Beispiel

[12] Für eine Einführung in XML: Transforming Data for Reuse and Re-publication with XML and XSL von The Programming Historian, einer Lernplattform für digitale Methoden für Geisteswissenschaftler, s. M. H. Beals, Transforming Data for Reuse and Republication with XML and XSL, in: The Programming Historian, 2016, online publiziert unter https://programminghistorian.org/lessons/transforming-xml-with-xsl/.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

```
<collection id="001-001-000" href="http://www.regesta-imperii.de/cei/001-001-000"/>
<collection id="001-002-001" href="http://www.regesta-imperii.de/cei/001-002-001"/>
<collection id="001-003-001" href="http://www.regesta-imperii.de/cei/001-004-002"/>
```

für drei karolingerzeitliche Regestenbände: Genauer gesagt steht 001-001-000 für RI I, für den „Böhmer-Mühlbacher“ mit den Herrscherregesten der Karolingerzeit, 001-002-001 für RI I,2,1, die erste Lieferung der Regesten Karls des Kahlen und 001-004-002 für RI I,4,2, den zweiten Teil der karolingerzeitlichen Papstregesten. Die erste Zahl im href-Element des Links steht jeweils für die Abteilung, die zweite für den Band und die dritte für den Teilband. Über die Eingabe von z.B.

```
http://www.regesta-imperii.de/cei/001-002-001/sources/
```

gelangt man zu einer Liste aller Regesten in der Kollektion „Karl der Kahle“. Als Beispiel sollen alle Regesten der Kollektion RI XIII H. 14 heruntergeladen werden, also dem 14. Heft der Regesten Kaiser Friedrichs III., oder:

```
http://www.regesta-imperii.de/cei/013-014-000/sources/
```

Bis hierhin erscheint die REST-Schnittstelle im Vergleich zur graphischen Nutzeroberfläche der Regestendatenbank sehr umständlich. Dies liegt daran, dass die Schnittstelle einen maschinen- und keinen menschenlesbaren Zugang zu den Regesten bietet. Die im Folgenden beschriebene Methode ließe sich auch auf einzelne Regesten oder auf ausgewählte Regesten, deren Links bestenfalls in einer Textdatei zusammengetragen wurden, anwenden. Außerdem ist die Methode auch auf andere Projekte übertragbar, die eine ähnliche Schnittstelle nutzen.

Voraussetzungen für die Benutzung

Rudimentäre Kenntnisse in Python, das für das Skript als Programmiersprache gewählt wurde, sind von Vorteil, aber nicht unbedingt für dessen Benutzung notwendig. Python wurde wegen seiner Nutzerfreundlichkeit gewählt, die eine individuelle Anpassung auch durch Nutzer ohne Vorkenntnisse ermöglicht. Zur Installation und Einführung ist zu empfehlen: Python Introduction

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

and Installation [13] von *The Programming Historian* und das Python Tutorial [14] von Codeacademy [15]. Als Basis für diesen Guide dient: Downloading Webpages with Python [16], ebenfalls von *The Programming Historian*. Es wurde die Python-Version 2.7.13 für Windows mit dem Windows x86 MSI installer verwendet. Diese ist unter *https://www.python.org/downloads/release/python-2713/* zu finden. Python und die zur Verfügung gestellten Skripte können in der jeweiligen Version analog auch auf Linux bzw. Mac OS-Systemen genutzt werden[17]. Die Skripte sind nicht Python 3-kompatibel. Mehrere Python-Versionen können aber parallel installiert werden.

Unter Windows ist relevant, in welchem Verzeichnis Python installiert ist, zum Beispiel:

```
C:/Python27/
```

Um Python-Skripte über die Windows-Kommandozeile auszuführen, muss nämlich „python.exe“ und der Pfad zu python.exe dem Namen des Skriptes vorgeschaltet sein. Unter anderen Betriebssystemen ist die Pfadangabe sowie „.exe“ nicht notwendig. Die Installation der Paketverwaltung pip („Pip installs Python“) soll im Folgenden als Beispiel dienen. Diese können wir dann dazu verwenden, weitere Python-Module zu installieren, die nicht bereits in der Installation enthalten sind. In unserem Fall wird das Modul lxml (*http://lxml.de/*) benötigt, das Python die Bearbeitung von XML-Dateien erlaubt.

Bevor wir pip installieren können, müssen wir die Installationsdatei *get-pip.py* herunterladen. Die Installationsdatei findet sich unter *https://pip.pypa.io/en/stable/installing/*. Nach dem Download von *getpip.py* müssen wir uns über die Windows-Eingabeaufforderung zum Speicherort der Datei bewegen. Zum Starten der Eingabeaufforderung ist entweder die Windows-Taste + R zu drücken oder in der internen Suche nach „Eingabeaufforderung“ zu suchen und die angezeigte Anwendung dann auszuführen. Nach dem Start der Eingabeaufforderung wird zunächst

[13] William J. Turkel und Adam Crumble, Python Introduction and Installation, in: The Programming Historian, 2012, online publiziert unter https://programminghistorian.org/lessons/introduction-and-installation/.
[14] Learn Python, hrsg. von Codeacademy, online publiziert unter https://www.codecademy.com/learn/learn-python/.
[15] Codeacademy, Inc, New York (NY), online unter https://www.codecademy.com/.
[16] William J. Turkel und Adam Crumble, Downloading Web Pages with Python, in: The Programming Historian, 2012, online publiziert unter https://programminghistorian.org/lessons/working-with-web-pages/.
[17] Eine Installationsanleitung für Python bzw. Pip auf Mac OS findet sich online unter http://docs.python-guide.org/en/latest/starting/install/osx/.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

```
Microsoft Windows [Version 10.0.15063]


(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.


C:\\Users\\benutzername>
```

angezeigt, mit dem jeweiligen Windows-Benutzernamen anstelle von „benutzername“. Im Beispiel haben wir *get-pip.py* unter C:/Pip/ abgespeichert.

Der Befehl, um sich durch die Ordnerstruktur zu bewegen, ist *cd*. *cd* schickt uns ein Verzeichnis nach oben:

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.
C:\\Users\\benutzername>cd ..
C:\\Users>cd ..
C:\\>cd Pip
C:\\Pip>
```

Der Befehl *dir* listet den Inhalt des Verzeichnisses auf, und zeigt uns, ob wir an der richtigen Stelle sind:

```
C:\\Pip>dir
      Datenträger in Laufwerk C: ist Windows
      Volumeseriennummer: 2025-A075
      Verzeichnis von C:\\Pip
      07.12.2017  19:09    <DIR>          .
      07.12.2017  19:09    <DIR>          ..
      27.11.2017  03:32         1.595.408 get-pip.py
        1 Datei(en),      1.595.408 Bytes
        2 Verzeichnis(se), 20.852.379.648 Bytes frei
C:\\Pip>
```

get-pip.py ist ein Python-Skript, daher muss „python.exe“ vorgeschaltet werden, um das Skript ausführen zu können. Dazu muss das System wissen, wo sich dieser Befehl befindet, in unserem Fall in C:/Python27/. Daher lautet der vollständige Befehl:

```
C:\\Pip>C:/Python27/python.exe get-pip.py
```

Daraufhin wird pip installiert. pip dient dann dazu, weitere Module zu installieren. Zur Verarbeitung von XML benötigen wir das Modul lxml. Mit dem folgenden Befehl wird lxml installiert:

```
C:\\Pip>C:/Python27/pip.exe install lxml
```

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Der analoge Befehl für Linux-Systeme (auch Mac OS) lautet

```
$ pip install lxml
```

Das Python-Hauptskript wird an seinem Speicherort auf dieselbe Weise ausgeführt[18]. Die Benutzung wird vereinfacht, wenn zumindest python.exe, gegebenenfalls aber auch pip.exe zu der PATH-Systemumgebungsvariable hinzugefügt wird. Wird der Speicherort von python.exe zur PATH-Variable hinzugefügt, ist dieser dem System bekannt, ohne, dass er bei zukünftigen Befehlen spezifiziert werden muss[19], z.B.:

```
C:\\Pip>python get-pip.py
```

Um bei Linux/Mac OS-Systemen die richtige Version zu verwenden, wird bei den Befehlen statt „python“ „python2.7“ und statt „pip“ „pip2.7“ verwendet.

Beschreibung der Benutzung der Skripte

Es werden bereitgestellt

- *Windows_Regesten_Download_ueber_die_REST_Schnittstelle_der_RI_Online.py* (Hauptskript)
- *Windows_regesten_herunterladen.py*
- *Windows_informationen_aus_xml_extrahieren.py*
- eine CSV-Datei der Namen und jeweiligen Links zu den Kollektionen.

Diese müssen zusammen abgespeichert werden.

Um nun Regesten einer Kollektion gezielt als XML-Dateien herunterzuladen und zu einer CSV-Datei zusammenzufassen, ist nur ein Befehl und die Angabe von Speicherorten nötig. Alle

[18] Weitere Informationen zu pip s. Fred Gibbs, Installing Python Modules with pip, in: The Programming Historian, 2013, online publiziert unter https://programminghistorian.org/lessons/installing-python-modules-pip/. Für eine weitergehende Einführung in die Windows-Kommandozeile s. Ted Dawson, Introduction tot he Windows Command Line with PowerShell, in The Programming Historian, 2016, online publiziert unter https://programminghistorian.org/lessons/intro-to-powershell/.

[19] Falls unter Windows die Pfadvariable nicht automatisch erstellt wird, kann dies manuell erfolgen. Hierzu sucht man bei Windows 10 in der internen Suche nach „Systemeigenschaften“, dann unter „Erweitert“ nach „Umgebungsvariablen“ und wählt in der Liste der Systemvariablen die „Path“-Variable aus. Unter „Bearbeiten“ kann ein neuer Eintrag hinzugefügt werden. Dies stellt allerdings nur eine Erleichterung dar und ist für die Ausführung nicht essenziell.

weiteren folgenden Ausführungen dienen lediglich der Erklärung, um das Skript individuell anpassen zu können und eventuelle Fehler zu erkennen.

Zunächst ist die Windows-Shell in dem Ordner zu starten oder über die Kommandozeile der Speicherort der Skripte aufzusuchen, im Beispiel *C:/RI-Download-Tool/*. Dann kann über folgenden Befehl das Hauptskript gestartet werden:

```
C:/RI-Download-Tool>C:/Python27/python.exe
Windows_Regesten_Download_ueber_die_REST_Schnittstelle_der_RI_Online.py
```

Linux-Systeme (und Mac OS) benötigen vor dem Namen des Skripts lediglich das Kommando „python“. Die folgenden Befehle können ansonsten analog benutzt werden. Nur auf die Nutzung des richtigen Skripts ist zu achten, da hier wegen einer abweichenden Nomenklatur von Dateipfaden eine eigene Version des Skripts namens *Linux_Regesten_Download_ueber_die_REST_Schnittstelle _der_RI_Online.py* verwendet werden muss.

Zunächst kann man sich alle verfügbaren Kollektionen anzeigen lassen:

```
Moechten Sie sich alle verfuegbaren Kollektionen anzeigen lassen(j/n)? j
RI I,1          http://www.regesta-imperii.de/cei/001-001-000/sources
RI I,2,1        http://www.regesta-imperii.de/cei/001-002-001/sources
RI I,3,1        http://www.regesta-imperii.de/cei/001-003-001/sources
RI I,3,2        http://www.regesta-imperii.de/cei/001-003-002/sources
[...]
RIplus Regg. Baden 1,1  http://www.regesta-imperii.de/cei/020-019-
001/sources
RIplus Regg. Baden 2    http://www.regesta-imperii.de/cei/020-019-
002/sources
RIplus Regg. Baden 3    http://www.regesta-imperii.de/cei/020-019-
003/sources
RIplus Regg. Baden 1,2  http://www.regesta-imperii.de/cei/020-019-
005/sources
Geben Sie einen Link zu einer Kollektion ein:
```

Hier können wir nun den Link zur gewünschten Kollektion einfügen. Dadurch wird die Kollektionsübersicht als „index.xml“ heruntergeladen und die Links zu den einzelnen Regesten der Kollektion in eine Textdatei im CSV-Format extrahiert. Für das Beispiel von RI [XIII] H. 14 (http://www.regesta-imperii.de/cei/013-014-000/sources) sieht das wie folgt aus:

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

```
Geben Sie einen Link zu einer Kollektion ein: http://www.regesta-impe-
rii.de/cei/013-014-000/sources
Gueltige URI, Download beginnt
URI http://www.regesta-imperii.de/cei/013-014-000/sources: Download erfolg-
reich, Kollektion gespeichert unter "index.xml"
Links extrahiert, gespeichert in "Regesten_Links/regesten_links_013-014-
000.txt"
Moechten Sie die einzelnen Regesten jetzt herunterladen(j/n)?
```

Das Skript erzeugt ein Verzeichnis am Speicherort Regesten_Links, und speichert darin unter *regesten_links_013-014-000.txt* die Liste der URIs der gewünschten Regesten ab. Anschließend können diese heruntergeladen werden.

Für diesen Test editieren wir die Datei mit den Regesten-Links und laden nur die ersten zehn Regesten herunter.

```
Moechten Sie die einzelnen Regesten jetzt herunterladen(j/n)? j
Anzahl an Regesten: 10
1 von 10 Regesten heruntergeladen, gespeichert unter "Regesten/013-014-
000/1440-03-00_1_0_13_14_0_1_1.xml" (10.0%)
2 von 10 Regesten heruntergeladen, gespeichert unter "Regesten/013-014-
000/1440-03-00_2_0_13_14_0_2_2.xml" (20.0%)
[...]
9 von 10 Regesten heruntergeladen, gespeichert unter "Regesten/013-014-
000/1440-05-16_3_0_13_14_0_9_9.xml" (90.0%)
10 von 10 Regesten heruntergeladen, gespeichert unter "Regesten/013-014-
000/1440-05-16_4_0_13_14_0_10_1.xml" (100.0%)
Regesten gespeichert in Ordner: Regesten_013-014-000
Moechten Sie jetzt die XML-Regesten in einer CSV-Datei zusammenfassen(j/n)?
```

Die Regesten sind nun als einzelne XML-Dateien gespeichert und unter ihrer ID im Ordner „Regesten_013-014-000“ zu finden. Als nächstes haben wir die Option, aus diesen Informationen zu extrahieren und in einer CSV-Datei („comma separated values“) zusammenzufassen:

```
Moechten Sie jetzt die XML-Regesten in einer CSV-Datei zusammenfassen(j/n)?
j
Geben sie den Namen des Ordners mit ihren Regesten ein:Regesten_013-014-000
Folgende Informationen werden aus den Regestendateien extrahiert:
ID, Startdatum, Enddatum, Ausstellungsort, Regestentext, URI
[...]
Regesten-CSV gespeichert in Regesten_CSV/regesten_csv_013-014-000.txt.
```

Voreingestellt ist die Extraktion der Regesten-ID, die nicht zu verwechseln ist mit der REST-ID, des Startdatums, Enddatums, Ausstellungsorts, Regestentexts und der URI. Als Spaltentrenner wird die Pipe („|“) verwendet, da Kommata und Anführungsteichen im Regestentext vorkommen können.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

So sieht das Ergebnis aus:

```
id|start|end|place|abstract|uri
"[RI XIII] H. 14 n. 1"|"1440-03-01",1440-03-31"|"[-]"|"Kg."|"http://www.re-
gesta-imperii.de/id/1440-03-00_1_0_13_14_0_1_1"
"[RI XIII] H. 14 n. 2"|"1440-03-01",1440-03-31"|"[-]"|"Kg. F. desgleichen
an Hz. Johann (IV.) von Bayern(-München)."|"http://www.regesta-impe-
rii.de/id/1440-03-00_2_0_13_14_0_2_2"
"[RI XIII] H. 14 n. 3"|"1440-03-01",1440-03-31"|"[-]"|"Kg. F. desgleichen
an Bf. Albrecht von Eichstätt."|"http://www.regesta-imperii.de/id/1440-03-
00_3_0_13_14_0_3_3"
```

Diese Datei kann anschließend mithilfe eines gängigen Tabellenkalkulationsprogrammes geöffnet werden.Wie zu sehen ist, enthalten nicht alle Regesten alle diese Informationen. Den XML-Dateien ist außerdem zu entnehmen, dass die Regesten in der Regel je nach Abteilung wesentlich mehr Informationen enthalten. Möchte man diese Auswahl ändern oder ergänzen, kann man im Skript *Windows_informationen_aus_xml_extrahieren.py* entsprechend den lxml-XPath ändern oder einen neuen hinzufügen. Wenn man die Regesten-Links zu einem späteren Zeitpunkt herunterladen möchte, kann man auch nur das Skript *Windows_regesten_herunterladen.py* bzw. *Linux_regesten_herunterladen.py* ausführen. Diese stellen eigenständige Teile des Hauptskripts dar.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Beschreibung der Funktionsweise der Skripte

Vor der Beschreibung der einzelnen Funktionen ist hier zunächst der Code zum Import der Kollektionenliste und zur Erstellung der Ordner für die Links zu den Regesten und die CSV-Dateien der Regesten zu sehen:

```
#Variablendefinition
f = open('kollektionen_links.txt', 'r')
kollektionen_links = f.readlines()
Names = []
Links = []
for x in kollektionen_links:
      Names.append(x.split('\t')[1])
      Links.append(x.split('\t')[2])
f.close()
Links = [line.replace('\n', '') for line in Links]
#Ordner anlegen
if os.path.isdir('Regesten_Links') == False:
      csv_dir = os.mkdir('Regesten_Links')
      if os.path.isdir('Regesten_CSV') == False:
            csv_dir = os.mkdir('Regesten_CSV')
```

Über „kollektionen_links.txt“ werden Namen und Links einzeln extrahiert. Zum Start werden beide zusammen ausgegeben, wenn dies gewünscht wird. Die Teilliste Links wird außerdem dazu verwendet, zu überprüfen, ob tatsächlich ein gültiger Link zu einer bereitgestellten Kollektion der RI Online angegegeben wurde. Dies bringt uns zur ersten Funktion.

Kollektionen herunterladen

Über die Python-library urllib2 lassen sich ganze Webseiten über ihre URI herunterladen. Die Funktion download_collection(link) übergibt eine solche URI über die Variable „link“ und speichert den Inhalt zunächst in der Datei *index.xml* ab. Die Funktion wird in der Windows-Version nur aufgerufen, wenn der User-Input einen gültigen Link zu einer Kollektion darstellt. In der Linux-Version wird nur überprüft, ob es sich um einen gültigen Link handelt. Sollte es sich also nicht um einen Kollektionen-Link handeln, wird trotzdem versucht, Daten herunterzuladen. Da diese nicht mit dem weiteren Skript kompatibel sind, wird die Ausführung an dieser Stelle mit einer Fehlermeldung beendet.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

```
#Funktionsdefinition
def download_collection(link):
      #Funktion zum Download der Kollektionsuebersicht als XML-Datei
      response = urllib2.urlopen(link)
      webContent = response.read()
      response.close()
      f = open('index.xml', 'w')
      f.write(str(webContent))
      f.close
      print 'URI '+ link +': Download erfolgreich, Kollektion gespeichert
unter "index.xml"'
      return
[...]
#Funktionsaufrufe
#Anfrage nach der URI als Kollektion
while True:
      kollektion_link = raw_input('Geben Sie einen Link zu einer Kollektion
ein: ')
      if kollektion_link in Links:
            break
      else:
            print 'Keine gueltige Kollektionen-URI!'
            continue
      print 'Gueltige URI, Download beginnt'
      download_collection(kollektion_link)
```

Links extrahieren

In *index.xml* befinden sich die URIs der Regesten. Sie sind allerdings noch, wie in der Schnittstelle angezeigt, im XML-Format in Tags eingebunden, sodass urllib2 die URI nicht als solche erkennt. Daher werden nun in der zweiten Funktion extract_links(link) die Links zu den einzelnen Regesten bereinigt und in einer Textdatei abgespeichert:

```
#Funktionsdefinition
def extract_links(link):
      #Funktion zur Extraktion der Links ueber die Übergabe des Kollektio-
nen-Links
      tree = etree.parse(kollektion_link)
      root = tree.getroot()
      #Textdatei fuer Speicherung von Links mit collections-ID erstellen
      file = open('Regesten_Links' + '/' + 'regesten_links' + '_' +
str(kollektion_link[34:45]) + '.txt', 'w')
      #Fuellen der Textdatei mit den URI's der einzelnen Regesten
      for child in root:
            file.write(child.attrib['href'] + '\n')
            file.close
      print 'Links extrahiert, gespeichert in "' + 'Regesten_Links' + '/' +
'regesten_links' + '_' + str(kollektion_link[34:45]) +  '.txt"'
      return
[...]
#Funktionsaufruf
#Extraktion der Links
if os.path.isfile('index.xml') == True:
      extract_links(kollektion_link)
```

Wir übergeben hierzu wieder den vom Nutzer eingegebenen Link zur Kollektion. Insofern wäre es nicht unbedingt notwendig, die Links im XML-Format in *index.xml* abzuspeichern. Eventuell können dadurch aber schon mögliche Probleme abgefangen werden. Deswegen wird die Funktion zur Extraktion der Links erst aufgerufen, wenn sichergestellt ist, dass *index.xml* existiert.

Mithilfe der lxml-library für Python können XML-Dateien verarbeitet werden. Über *ET.parse* wird durch die XML-Baumstruktur (*tree*) der Datei geparst und über *tree.getroot()* der root-Tag, in unserem Fall „collection“, der allen Regesten-URI’s übergeordnet ist, eingelesen. Über *child in root* können wir nun in einem for-loop auf alle einzelnen URI’s in der Datei zugreifen und für jede einzelne das „href“-Attribut mit der URI in die neue Datei *regesten_links_013-014-000.txt* schreiben. Damit besitzen wir nun eine bereinigte Textdatei aller Regesten-URIs in der Kollektion RI XIII H. 14, insgesamt 501:

```
http://www.regesta-imperii.de/cei/013-014-000/sources/1440-03-
00_1_0_13_14_0_1_1
http://www.regesta-imperii.de/cei/013-014-000/sources/1440-03-
00_2_0_13_14_0_2_2
http://www.regesta-imperii.de/cei/013-014-000/sources/1440-03-
00_3_0_13_14_0_3_3
http://www.regesta-imperii.de/cei/013-014-000/sources/1440-03-
00_4_0_13_14_0_4_4
[...]
```

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Regesten herunterladen

Den eigentlichen Download der Regesten übernimmt die Funktion download_regests(file), der wir die Textdatei mit den Links übergeben.

```
#Funktionsdefinition
def download_regests(file):
      #Funktion zum Download der einzelnen Regesten aus einer Textdatei mit
Regestenlinks#Download der einzelnen Regesten
      f = open(file, 'r')
      links = f.readlines()
      print "Anzahl an Regesten: " + str(len(links))
      f.close()
      name_link = links[0]
      if os.path.isdir('Regesten' + '_' + str(name_link[34:45])) == False:
            download_dir = os.mkdir('Regesten' + '_' +
str(name_link[34:45]) + '/')
      for link in links:
            response = urllib2.urlopen(link)
            webContent = response.read()
            response.close()
            f = open('Regesten' + '_' + str(name_link[34:45]) + '/' +
str(link[54:-1]) + '.xml', 'w')
            f.write(webContent)
            f.close
            num_files = len([f for f in os.listdir('Regesten' + '_' +
str(name_link[34:45]) + '/')
            if os.path.isfile(os.path.join('Regesten' + '_' +
str(name_link[34:45]) + '/', f))])
                  print str(num_files) + ' von ' + str(len(links)) + ' Re-
gesten heruntergeladen, gespeichert unter "Regesten/'    +
str(name_link[34:45]) + '/' + str(link[54:-1]) + '.xml" '+ '(' +
str(round(num_files * 100/(len(links)),1)) + '%' + ')'
                  time.sleep(1)
                  print 'Regesten gespeichert in Ordner: ' + 'Regesten' +
'_' + str(name_link[34:45])
                  return
[...]
#Funktionsaufruf
#Anfrage, ob Regesten jetzt heruntergeladen werden sollen
if os.path.isfile('Regesten_Links' + '/' + 'regesten_links' + '_' +
str(kollektion_link[34:45]) + '.txt') == True:
      download_bestaetigung = raw_input('Moechten Sie die einzelnen Reges-
ten jetzt herunterladen(j/n)? ')
      #Wenn 'j', Download der Regesten
      if download_bestaetigung == 'j':
            download_regests('Regesten_Links' + '/' +  'regesten_links' +
'_' + str(kollektion_link[34:45]) + '.txt')
```

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Die Funktion öffnet die soeben erstellte Textdatei und liest sie Zeile für Zeile als Python-Liste *links* ein. In einem weiteren for-loop[20] wird nun für jeden einzelnen „link in links" die jeweilige URI an urllib2 übergeben und das einzelne Regest ins Unterverzeichnis Regesten als XML-Datei heruntergeladen. Der Datensatz wird benannt nach der internen Regesten-ID, die mit der 54. Stelle in der URI beginnt *link[54:-1]*. Darüber hinaus ist der for-Loop mit *time.sleep(1)* um eine Sekunde verzögert, um einen denial of service zu vermeiden und den Server nicht mit zu vielen Anfragen auf einmal zu überlasten.

Hierbei ist zu beachten, dass durch diese Verzögerung der Download größerer Kollektionen mit mehreren tausend Regesten unter Umständen mehrere Stunden in Anspruch nehmen kann.[21] Daher sollte der Download unter Berücksichtigung der Auslastung der Server der RI-Online am besten in den Randzeiten durchgeführt werden, sonst sind Fehler beim Download möglich.

Die Funktion wird nur aufgerufen, wenn die entsprechende Datei mit den Regesten-Links existiert und die Bestätigung zum Download gegeben wurde. Ansonsten wird das Programm an dieser Stelle beendet. Es ist empfehlenswert den Download in diesem Fall über

Windows_Regesten_herunterladen.py

durchzuführen. Hierbei handelt es sich um ein Teilskript, dem man direkt die Textdatei mit den Regesten-Links übergeben kann, ohne noch einmal die Kollektion herunterzuladen und bereinigen zu müssen.

Damit enthält das Unterverzeichnis *Regesten_013-014-000* nun alle Regesten der Kollektion RI XIII H.14 als einzelne XML-Datensätze. Das Skript wird nun fragen, ob die Regesten in einer CSV-Datei zusammengefasst werden sollen. Wer dies nicht möchte, kann den Guide und das Skript an dieser Stelle mit der Eingabe von „n" beenden. Ansonsten folgt die jetzt zu erläuternde

[20] Ein for-loop führt einen Befehl für eine bestimmte Anzahl von Wiederholungen aus, i. E. für die Anzahl der URIs in der Textdatei, siehe https://wiki.python.org/moin/ForLoop/.

[21] Es sollten andere Möglichkeiten gewählt werden, wenn mit dem Gesamtbestand der RI-Daten gearbeitet werden soll. Diese werden beispielsweise hier als gepackte Datei zum Donwload angeboten (allerdings nicht immer auf dem aktuellen Stand): http://www.regesta-imperii.de/fileadmin/CEI-Regesten.tgz. Auch mittels einer Spark-Umgebung werden für den Download des Bestandes mindestens 2 Tage benötigt, schnellere Ergebnisse sind mit einem in Java programmierten Thread parallelisierter Crawler möglich, wie ein Team um Michael Haft im Rahmen eines Praktikums an der ADWL Mainz austestete.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Extraktion von Informationen aus den XML-Dateien aller Regesten und die Zusammenfassung in einer einzelnen CSV-Datei.

Transformation von XML in CSV

Bevor wir fortfahren, betrachten wir zunächst die Struktur des Regests im CEI-Format. Um über die Baumstruktur beispielsweise auf die Regesten-IDs des gedruckten Regests zuzugreifen, müssen wir dem Pfad charter/chDesc/head/idno folgen:

```
[...]
<charter>
    <chDesc>
        <head>
            <idno>[RI XIII] H.14 n. 136</idno>
[...]
```

Wir greifen für die CSV-Datei nun ID-Nummer („idno“), Startdatum des Zeitraums auf den sich das Regest bezieht bzw. in dem es ausgestellt worden sein könnte („dateRange“ mit dem Attribut „from“), Enddatum („dateRange“ mit dem Attribut „to“), Ausstellungsort („placeName“), Regestentext („abstract“) und URI (ebenfalls „idno“, unter einem anderen XPath: „teiHeader/fileDesc/sourceDesc/bibl/idno“) auf.

Die Funktion hierzu ist informationen_extrahieren(path) und übernimmt als Variable das Verzeichnis mit den XML-Dateien der Regesten.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

```
#Funktionsdefinition
def informationen_extrahieren(path):
      #Funktion zur Extraktion von Informationen aus der Regestendatei
      file = open(os.path.dirname(os.path.abspath(__file__)) + '\\' + 'Re-
gesten_CSV' + '\\' +  'regesten_csv'+ '_' + str(path[9:20]) +'.csv'| 'w')
      #CSV-Datei mit Ueberschriften schreiben
      file.write('"' + 'id' + '"' + '|' + '"' + 'start' + '"' + '|' + '"'
+ 'end' + '"' + '|'+ '"' + 'place' + '"' + '|' + '"' + 'abstract' + '"' +
'|' + '"' + 'uri' + '"' + '\n')
      file.close
      os.chdir(os.path.dirname(os.path.abspath(__file__)) + '\\' + path)
      #Parsen durch XML-Dateien einzelner Regesten im Zielordner
      for filename in os.listdir(os.path.dirname(os.path.ab-
spath(__file__))):
            doc = etree.parse(filename)
            #Extrahieren der ID
            idno = doc.findtext('charter/chDesc/head/idno')
            #Extrahiern des Startdatums
            start = doc.find('charter/chDesc/head/issued/issueDate/p/dat-
eRange').attrib['from']
            #Extrahieren des Enddatums
            end = doc.find('charter/chDesc/head/issued/issueDate/p/dat-
eRange').attrib['to']
            #Extrahieren der Ortsangabe
            place = doc.findtext('charter/chDesc/head/issued/issue-
Place/placeName')
            #Extrahieren des Abstracts
            abstract = doc.findtext('charter/chDesc/abstract/p')
            #Extrahieren der URI
            uri = doc.findtext('teiHeader/fileDesc/sourceDesc/bibl/idno')
            line = ''
            #CSV-Datei
            if place != None:
                  line = '"' + idno + '"' + '|' + '"' + start + '"' + '|'+
end + '"' + '|' + '"' + place + '"' + '|' + '"' + abstract + '"' + '|' +
'"' + uri + '"' + '\n'
            elif place == None:
                  line = '"' + idno + '"' + '|' + '"' + start + '"' + '|' +
end + '"' + '|'  + '"' + '[None]' + '"' + '|' + '"' + abstract + '"' + '|'
+ '"' + uri + '"' + '\n'
            os.chdir('..')
            file = open(os.path.dirname(os.path.abspath(__file__)) + '\\' +
'Regesten_CSV' + '\\' + 'regesten_csv'+ '_' + str(path[9:20]) +'.txt'| 'a')
            file.write(line.encode('utf-8'))
            file.close
            os.chdir(os.path.dirname(os.path.abspath(__file__)) + '\\' +
path)
      print 'Regesten-CSV gespeichert in Regesten_CSV/regesten_csv_' +
str(path[9:20]) +'.csv' + '.'
      return
[...]
#Funktionsaufruf
if informations_extraktion == 'j':
      pfad_anfrage = raw_input('Geben sie den Namen des Ordners mit ihren
Regesten ein:')
[...]
informationen_extrahieren(pfad_anfrage)
```

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Im Beispiel werden die *Regesten_013-014-000* als Input und damit als Ordnerpfad angegeben. Zuerst wird nun die CSV-Datei angelegt, die die zu extrahierenden Informationen aufnehmen soll, mit den entsprechenden Überschriften. Dann wechseln wir in das Unterverzeichnis mit den Regesten (*os.chdir([...])*).Für jede einzelne Datei in diesem Verzeichnis werden nun die über XPath definierten Informationen extrahiert (*for filename in os.listdir([...])*). Die einzelnen Variablen zusammengefasst als String bilden eine Zeile für ein Regest in der neuen zusammengefassten CSV-Datei im Textformat. Über ein if/else Statement wird für alle Fälle, in denen der Ausstellungsort im XML-Regest leer ist, der jeweilige Ausstellungsort in der zusammengefassten Datei mit „[o.O.]“ benannt. Dies ist nicht unbedingt notwendig, ermöglicht aber die Unterscheidung zwischen Fehlern in der Datei und fehlenden Ausgangsinformationen.

Zu Fehlern bei der CSV-Erstellung kann es kommen, wenn bei einzelnen Dateien das CEI-Schema nicht korrekt angewandt wurde. Bei der großen Anzahl der Datensätze, die auch durch Nutzerinput via Nachträgen[22] Veränderungen unterworfen sind, ist dies leider nicht ganz zu auszuschließen. In dem Fall ist sich an die RI-Online Redaktion zu wenden.

Damit besitzen wir nun eine CSV-Datei der Kollektion RI XIII H. 14, in der jede einzelne Zeile ein Regest mit den von uns extrahierten Informationen repräsentiert und die als Grundlage für eine Nachnutzung der Daten für eigene Forschungsprojekte dienen kann. Zur Vereinfachung der Benutzung wurde das hier präsentierte Skript aber auf die REST-Schnittstelle der Regesta Imperii Online zugeschnitten. Mit denselben Methoden lassen sich auch Informationen aus anderen Datenbanken, die ähnlich aufgebaut sind, anfragen. Es ist lediglich eine REST-Schnittstelle nötig, die die Links zu den Datensätzen im XML-Format anbietet. Die Details können problemlos im Skript angepasst werden. Die CSV-Erstellung setzt natürlich eine entsprechende Strukturierung der XML-Dateien voraus, doch durch die Anpassung der XPATH-Ausdrücke können auch anders strukturierte Datensätze ausgelesen werden. Genauso lassen sich auch andere Informationen der RI-Regesten auslesen.

[22] Online einzusehen unter: http://www.regesta-imperii.de/regesten/nachtraege.html/.

Zitation:

Victor Westrich und Yannick Weber, Der Weg zu den Forschungsdaten. Ein Beispielguide für die Nutzung der REST-Schnittstelle der Regesta Imperii mithilfe von Python, in: Mittelalter. Interdisziplinäre Forschung und Rezeptionsgeschichte 1 (2018), S. 67-87, https://mittelalter.hypotheses.org/11794.

Skripte

Die vollständigen Skripte können unter den folgenden Links von der Regesta Imperii Online heruntergeladen werden:

RI-Download-Tool Windows: http://www.regesta-imperii.de/fileadmin/user_upload/downloads/RI-Downloadtool_Windows.zip

RI-Download-Tool Mac OS bzw. Linux: www.regesta-imperii.de/fileadmin/user_upload/downloads/RI-Downloadtool_Linux.zip