

01-clean-juris-docs

2024-09-01

```
[ ]: %run "./common.py"
```

```
[ ]: import os
import pandas as pd
from bs4 import BeautifulSoup
```

```
[ ]: def extract_meta(soup):
    colnames = [x.get_text().strip() for x in soup.find('div', {'class':
    ↳ 'documentHeader'}).table.find_all('td', {'class': 'TD30'})]
    colnames = list(filter(lambda elem: not elem.startswith('ECLI'), colnames))
    colvals = [x.get_text().strip() for x in soup.find('div', {'class':
    ↳ 'documentHeader'}).table.find_all('td', {'class': 'TD70'})]
    assert len(colnames) == len(colvals)
    return {x:y for x, y in list(zip(colnames, colvals))}

def get_meta(doc):
    docid = doc.split('.')[0]
    with open(f'{path_to_docs}/{doc}') as f:
        soup = BeautifulSoup(f.read(), 'lxml')
    return **extract_meta(soup), 'jurisidentifier':docid

def get metas(docs):
    meta = []
    for doc in docs:
        meta.append(get_meta(doc))
    return meta
```

```
[ ]: def create_meta_dataframe(meta):
    df = pd.DataFrame.from_records(meta)
    df = df.rename({col:col.lower().replace(':', '') for col in df.columns},
    ↳ axis=1)
    df['entscheidungstag'], df['entscheidungsmonat'], df['entscheidungsjahr'] =
    ↳ (list(zip(*[[int(y) for y in x.split('.')]
    ↳
    ↳ for x in df.entscheidungsdatum]]))
    df['eingangszeichen'], df['eingangsjahr'] = list(zip(*[[int(y) for y in x.
    ↳ split(' ')[-1].split('/')]
```

```

for x in df.
    aktenzeichen]))
    df['eingangsjahr'] = [x + 2000 if x <= 20 else x + 1900 for x in
df['eingangsjahr']]
    df['dateiname'] = [f"{y}-{m:02}-{d:02} {a.replace('/', '-')} {j} {t}" for
y,m,d,a,j,t
                        in zip(df['entscheidungsjahr'],
df['entscheidungsmonat'], df['entscheidungstag'],
                        df['aktenzeichen'], df['jurisidentifizier'],
df['dokumenttyp'])]
    return df[sorted(df.columns)].set_index('jurisidentifizier')

```

0.0.1 Set up access to docs

```

[ ]: path_to_newdocs = ensure_exists('../data/BGH-XI-cleaned')
path_to_helper_data = ensure_exists('../data/BGH-XI-helpers')
path_to_docs = '../data/BGH-XI-decdata'
docs = [x for x in os.listdir(path_to_docs) if x.endswith('.html')]
len(docs)

```

0.0.2 Create CSV with decision metadata

```

[ ]: meta = get metas(docs)

```

```

[ ]: df = create_meta_dataframe(meta).sort_values('dateiname')
df.to_csv(f'{path_to_helper_data}/bgh-xi-entscheidungen.csv')

```

0.0.3 Parse content (this will include Orientierungssätze as they are not structurally distinguishable from the rest)

```

[ ]: def get_doc_text(doc, path_to_docs):
    with open(f'{path_to_docs}/{doc}') as f:
        soup = BeautifulSoup(f.read(), 'lxml')
        # 'dd' seems to be the content identifier tag
        return '\n'.join([x.get_text().strip() for x in soup.find_all('dd')])

def write_doc_text(doc, metadf, path_to_docs, path_to_newdocs):
    text = get_doc_text(doc, path_to_docs)
    docid = doc.split('.')[0]
    with open(f'{path_to_newdocs}/{metadf.at[docid, "dateiname"]}.txt', 'w') as
f:
        f.write(text)

def write_doc_texts(docs, metadf, path_to_docs, path_to_newdocs):
    for doc in docs:
        write_doc_text(doc, metadf, path_to_docs, path_to_newdocs)

```

```
[ ]: assert len(df) == len(set(df.dateiname))
```

```
[ ]: write_doc_texts(docs, df, path_to_docs, path_to_newdocs)
```

```
[ ]:
```