

# **Zusätzliche Beteiligung**

Analyse der Artikel des Nachrichtensenders NTV

Kai Herbst, Manuel Zeh, Henrik Popp

August 2024

# Inhaltsverzeichnis

<b>1</b>	<b>Daten-Pipeline</b>	<b>1</b>
<b>A</b>	<b>Lambda-Funktion zur Sammlung der Artikel</b>	<b>1</b>
A.1	Index.mjs . . . . .	1
A.2	Categories.mjs . . . . .	2
A.3	S3Helper.mjs . . . . .	3
A.4	Helper.mjs . . . . .	4

# 1 Daten-Pipeline

## A Lambda-Funktion zur Sammlung der Artikel

### A.1 Index.mjs

---

Die Hauptfunktion, die von der Lambda-Funktion aufgerufen wird.

Listing 1: index.mjs

```
1 import helper from "./helper.mjs";
2 import categories from "./categories.mjs";
3 import S3Uploader from "./s3Helper.mjs";
4
5 export const handler = async (event, context) => {
6     const s3Uploader = new S3Uploader();
7     const bucket = "ntv-data";
8     const baseKey = helper.getYesterdaysDate().toISOString().split("T")[0];
9     for (let category of categories) {
10         let { links, rss } = await
11             helper.getLinksOfYesterdaysArticelsAndRssObjectOf(category);
12         if (links.length == 0) {
13             continue;
14         }
15         let rssKey = baseKey + "/" + category + "/rss.json";
16         let rssJSONString = JSON.stringify(rss);
17         await s3Uploader.uploadContent(bucket, rssKey, rssJSONString);
18         for (let link of links) {
19             let articleKey = baseKey + "/" + category + "/" +
20                 helper.getFileNameFromLink(link);
21             let htmlContent = await helper.downloadContentFrom(link);
22             let cleanedHTML = helper.cleanHTMLContent(htmlContent);
23             await s3Uploader.uploadContent(bucket, articleKey, cleanedHTML);
24         }
25     }
26     return {
27         statusCode: 200,
28         body: JSON.stringify("Done!"),
29     };
30 }
```

## A.2 Categories.mjs

---

Die verschiedenen NTV-Kategorien, die abgerufen werden sollen.

Listing 2: categories.mjs

```
1 export default [  
2   "dasbeste",  
3   "politik",  
4   "politik/US-wahl-2024",  
5   "wirtschaft",  
6   "boersenkurse",  
7   "sport",  
8   "panorama",  
9   "leute",  
10  "technik",  
11  "ratgeber",  
12  "wissen",  
13  "auto",  
14  "infografik",  
15  "regionales",  
16  "der_tag/alle_tage",  
17  "meinung",  
18 ];
```

### A.3 S3Helper.mjs

---

Funktionen zum Zugriff auf die entsprechenden AWS S3-Buckets.

Listing 3: s3Helper.mjs

```
1 import { S3Client, PutObjectCommand } from "@aws-sdk/client-s3";
2
3 export default class S3Uploader {
4   constructor() {
5     this.s3Client = new S3Client({ region: "eu-central-1" });
6   }
7
8   async uploadContent(bucket, key, contentString) {
9     const command = new PutObjectCommand({
10       Bucket: bucket,
11       Key: key,
12       Body: contentString,
13     });
14
15     try {
16       console.log("Uploading", bucket + "/" + key, "...");
17       const response = await this.s3Client.send(command);
18       console.log("Finished!");
19     } catch (err) {
20       console.log(err);
21     }
22   }
23 }
```

## A.4 Helper.mjs

---

Hilfsfunktionen zum Verarbeiten der von NTV empfangenen Daten.

Listing 4: helper.mjs

```
1 import { XMLParser } from "fast-xml-parser";
2
3 function buildRSSLinkFromCategory(category) {
4     let domain = "https://www.n-tv.de/";
5     let postfix = "/rss";
6     return domain + category + postfix;
7 }
8
9 function getYesterdaysDate() {
10     const date = new Date();
11     date.setHours(date.getHours() + 2);
12     date.setDate(date.getDate() - 1);
13     return date;
14 }
15
16 function dateWasYesterday(datestring) {
17     const yesterdaysDate = getYesterdaysDate();
18     const parsedDate = new Date(datestring);
19     return (
20         yesterdaysDate.getFullYear() == parsedDate.getFullYear() &&
21         yesterdaysDate.getMonth() == parsedDate.getMonth() &&
22         yesterdaysDate.getDate() == parsedDate.getDate()
23     );
24 }
25
26 function parseXMLString(xmlstring) {
27     const parser = new XMLParser();
28     return parser.parse(xmlstring);
29 }
30
31 function getRssItemsOfYesterday(rssObject) {
32     if (rssObject.rss.channel.item == undefined) {
33         return [];
34     }
35     let items = rssObject.rss.channel.item;
36     return items.filter((item) => dateWasYesterday(item.pubDate));
37 }
38
```

```

39 async function downloadRssXMLOfCategory(category) {
40     let url = buildRSSLinkFromCategory(category);
41     return await downloadContentFrom(url);
42 }
43
44 async function downloadContentFrom(url) {
45     let response = await fetch(url);
46     return await response.text();
47 }
48
49 function getFileNameFromLink(link) {
50     return link.split("/").at(-1);
51 }
52
53 function getLinksOfRssItems(rssItems) {
54     let links = [];
55     for (let item of rssItems) {
56         links.push(item.link);
57     }
58     return links;
59 }
60
61 function cleanHTMLContent(htmlString) {
62     let cleaned = htmlString.replace(
63         /<script\b[^\<]*(?:(!</script><[^\<]*)*</script\s*>/gi, "");
64     cleaned = cleaned.replace(
65         /<nav\b[^\<]*(?:(!</script><[^\<]*)*</nav\s*>/gi, "");
66     cleaned = cleaned.replace(
67         /<style\b[^\<]*(?:(!</script><[^\<]*)*</style\s*>/gi, "");
68     cleaned = cleaned.replace(
69         /<link\b[^\<]*(?:(!</script><[^\<]*)*</link\s*>/gi, "");
70     return cleaned;
71 }
72
73 async function getLinksOfYesterdaysArticelsAndRssObjectOf(category) {
74     let rssXMLString = await downloadRssXMLOfCategory(category);
75     let rss = parseXMLString(rssXMLString);
76     let yesterdaysItems = getRssItemsOfYesterday(rss);
77     let links = getLinksOfRssItems(yesterdaysItems);
78     rss.rss.channel.item = yesterdaysItems;
79     rss.rss.channel.category = category;
80     return { links: links, rss: rss };
81 }

```

```
82
83 export default {
84     getYesterdaysDate,
85     getLinksOfYesterdaysArticelsAndRssObjectOf,
86     getFileNameFromLink,
87     downloadContentFrom,
88     cleanHTMLContent,
89 };
```