NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

A new ontology and software for the automatic updating of the platform "Nomothesia"

Stavros D. Giannakis

Supervisors: **Manolis Koubarakis,** Professor
Ilias Chalkidis, PhD Candidate

ATHENS

OCTOBER
2017

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μια νέα οντολογία και ένα λογισμικό για την αυτόματη ενημέρωση της πλατφόρμας "Νομοθεσία

Σταύρος Δ. Γιαννάκης

Επιβλέποντες: **Μανώλης Κουμπαράκης,** Καθηγητής
Ηλίας Χαλκίδης, Υποψήφιος Διδάκτορας

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ
2017

BSc THESIS

A new ontology and software for the automatic updating of the platform "Nomothesia"

Stavros D. Giannakis
S.N.: 1115201100010

SUPERVISOR: **Manolis Koubarakis,** Professor
Ilias Chalkidis, PhD Candidate

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μια νέα οντολογία και ένα λογισμικό για την αυτόματη ενημέρωση της πλατφόρμας "Νομοθεσία

Σταύρος Δ. Γιαννάκης
Α.Μ.: 1115201100010

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανώλης Κουμπαράκης,** Καθηγητής
Ηλίας Χαλκίδης, Υποψήφιος Διδάκτορας

ABSTRACT

This work is part of the Nomothesi@ Project (http://legislation.di.uoa.gr) running under the hood of the Dep. of Informatics and Tel. at Kapodistrian University of Athens. The aim of this project is to contribute to the transparency of legal knowledge and make easier the public access. Nomothesi@ is a project designed to give access to Greek legislation using semantic web technologies. This work contributes to the continuous updating of the data library of Nomothesi@ platform, using tools that aid in its renewal and structuring. Using the National Printing Office's website as a resource provider, we can update our library with the latest daily issues, but also manage the legislative archive on its own platform. Recovering Government Gazette issues from the National Printing House is done by employing a Web crawler written in Python programming language. Additionally, we propose a new extension ontology, which describes the Government Gazette, its issues and the relation with the legal documents and we demonstrate the querying capabilities.

SUBJECT AREA: Semantic Web, Software Engineering

KEYWORDS: Web crawling, Web development, E-Government, Job scheduling, Greek Government Gazette, Legislative Knowledge Representation, Content Management Systems

ΠΕΡΙΛΗΨΗ

Η εργασία αυτή αποτελεί μέρος του έργου Νομοθεσί@ (http://legislation.di.uoa.gr) που λειτουργεί κάτω από την αιγίδα του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Καποδιστριακού Πανεπιστημίου Αθηνών. Στόχος αυτής της εργασίας είναι να συμβάλει στη διαφάνεια της νομικής γνώσης και να διευκολύνει την πρόσβαση του κοινού σε αυτή. Το έργο Νομοθεσί@ έχει σχεδιαστεί για να παρέχει πρόσβαση στην ελληνική νομοθεσία χρησιμοποιώντας τεχνολογίες σημασιολογικού ιστού. Η εργασία αυτή συμβάλλει στη συνεχή ενημέρωση της βιβλιοθήκης δεδομένων της πλατφόρμας Νομοθεσί@, χρησιμοποιώντας εργαλεία που βοηθούν στην ανανέωση και οργάνωση της. Χρησιμοποιώντας τον ιστότοπο του Εθνικού Τυπογραφείου ως πάροχο πόρων, μπορούμε να ενημερώνουμε καθημερινά τη βιβλιοθήκη μας με τα νέα τεύχη, αλλά επίσης να διαχειριζόμαστε το νομοθετικό αρχείο στην δική του πλατφόρμα. Η ανάκτηση των θεμάτων της Εφημερίδας Κυβερνήσεως από το Εθνικό Τυπογραφείο γίνεται με την χρήση ενός ανιχνευτή ιστού γραμμένο σε γλώσσα προγραμματισμού Python. Επιπλέον, προτείνουμε μια νέα οντολογία επέκτασης, η οποία περιγράφει την Εφημερίδα της Κυβερνήσεως, τα θέματα της και τη σχέση με τα νομικά έγγραφα και επιδεικνύουμε τις δυνατότητες διερεύνησης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Σημασιολογικός Ιστός, Μηχανική λογισμικού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Προγράμματα Ανίχνευσης Web, Ανάπτυξη Ιστοσελίδων Διαδικτύου, Προγραμματισμός Εργασιών, Κυβερνητική Εφημερίδα, HTML, Αναπαράσταση Γνώσης Νομοθεσίας, Σύστηματα Ελέγχου Περιεχομένου

AKNOWLEDGMENTS

Firstly, I would like to acknowledge my supervisor Professor, Mr. Manolis Koubarakis. His Artificial Intelligence course and his research about Semantic Web and Linked Data assisted me in deciding to contribute to the Nomothesi@ project (http://legislation.di.uoa.gr). Also, I would like to acknowledge MSc Student Ilias Chalkidis, whose guidance throughout the process of this thesis was tremendous, giving me ideas and advices and supervising me the whole time.

CONTENTS

1. INTRODUCTION 13

1.1 Objectives of the thesis 13

1.2 Organization of the thesis 14

2. RELATED WORK 15

2.1 Software Engineering Aspects 15

2.1.1 GoogleBot (Web Crawling) 15

2.1.2 WordPress (Content Management Systems) 16

2.1.3 jobscheduler (Job Scheduling) 17

2.2 Government Gazette Semantic Representation 17

2.2.1 European Legislation Identifier 17

2.2.2 Akoma Ntoso 17

2.3 Conclusion 18

3. BACKGROUND 19

3.1 Greek Government Gazette 19

3.1.1 Historical Introduction 19

3.1.2 Current Organization of the Greek Government Gazette 20

3.1.3 ET.gr, the Greek Government Gazette in the age of the World Wide Web.. 23

3.2 Web Retrieval and CMS 25

3.2.1 Web Crawling 25

3.2.2 CMS 26

3.3 The dynamics of RDF Data Models, OWL Ontologies and SPARQL - Semantic Perspectives 28

3.3.1 The new Data Model with Resource Description Framework (RDF) 28

3.3.2 OWL Ontologies and SPARQL 29

3.3.3 Semantic perspectives and future Web 29

3.4 Semantic Representation of Government Gazette 30

3.5 Conclusion 30

4. GREEK GOVERNMENT GAZETTE LIBRARY 31

4.1 Greek Government Gazette Crawler 31
4.1.1 Crawler Technologies 31
4.1.2 Crawler Demonstration 36

4.2 Greek Government Gazette Crawler Scheduler 40

4.3 Greek Government Gazette Library CMS 40
4.3.1 Architecture 40
4.3.2 Greek Government Gazette Library Demonstration 44

4.4 Conclusion 48

5. GREEK GOVERNMENT GAZETTE ONTOLOGY 49

5.1 Ontology 49

5.2 Greek Government Gazette URI Pattern 50

5.3 Greek Government Gazette Dataset 51

5.4 Conclusion 51

6. CONCLUSION AND FUTURE WORK 52

6.1 Conclusion 52

6.2 Future Work 52
6.2.1 Expanding the Content Management System’s features 52
6.2.2 Expanding the Web Crawler’s features 52

APPENDIX 55

A: Web Application 55

REFERENCES 58

LIST OF TABLES

Table 3. 1: Issue Names and Subjects 22

Table 3. 2: Crawler Policies 26

Table 3. 3: Roles and responsibilities in Content Management 27

Table 5. 1: URI Issue IDs 50

Table 6. 1: List of Abbreviations 54

Table 6. 2: Table of Translations 54

LIST OF FIGURES

Figure 3. 1: First ever Gazette release 20

Figure 3. 2: Contemporary Greek Government Gazette document (Issue B)................23

Figure 3. 3: ET.gr..24

Figure 3. 4: ET.gr search page 24

Figure 3. 5: Indexing process used by search engine crawlers. 26

Figure 3. 6: CMS Overview...27

Figure 4. 1: Daily Gazette release (Seed) webpage 32

Figure 4. 2: parse and dailyparse functions...34

Figure 4. 3: downloadfek visual representation 35

Figure 4. 4: parse Function's Response 36

Figure 4. 5: parse function's response elements 37

Figure 4. 6: result table...38

Figure 4. 7: prop Class (Issue) 38

Figure 4. 8: even/odd class (data) 39

Figure 4. 9: Scrapyd Web Interface 41

Figure 4. 10: Scrapyd Job Monitoring..42

Figure 4. 11: Search function...42

Figure 4. 12: URL patterns 44

Figure 4. 13: Homepage..45

Figure 4. 14: Crawling page...45

Figure 4. 15:Library page (directory view) 46

Figure 4. 16: Library page (document view) 46

Figure 4. 17: Search page 47

Figure 4. 18: Search results page...47

Figure 4. 19: Browser File Preview...48

Figure 5. 1:Greek Government Gazette Ontology 49

Figure 5. 2: Dataset RDF Example Graph 51

1. INTRODUCTION

People around the world have access to multiple sources of information. Specifically, we live in an era of information explosion, with information being measured in exabytes When it comes to the law, everyone seems to think that it should concern only lawyers and judges, while it's true purpose is to help and serve the public. Nomothesi@, being a project that aims to fulfil the demand of the public to acquire knowledge of what a legal document states [1], how it has been modified over the years, or even ask more complex questions. To serve this purpose, the Nomothesi@ platform needs to have an automatically updated Government Gazette library, fetched from the National Printing House's website daily. This way, every individual that needs to use legislation as a part of their job, or even just to gather knowledge, is able to do so without great effort.

1.1 Objectives of the thesis

As it stands at the moment, Nomothesi@'s library is stagnant. There is a small amount of Gazette issues populating it and the only process used to insert more issues is manually, by downloading them ourselves, parsing them using the Greek Government Gazette Parser in order to create the new RDF triples and then moving them into the platform's triplestore.

The main purpose of this thesis is to research and implement a web crawler, to retrieve Government Gazette issues from the National Printing House's website. This web crawler, or spider, should be capable of fetching Gazette issues based on their release date. With the construction of this spider, we will enhance the library's capabilities of being kept up to date.

Furthermore, the aforementioned web crawler should be activated automatically every day, so human intervention is not required for the data set to be updated. This matter calls for another script to be utilized, one that can execute the spider daily, at a specific hour and download all the new issues released on the current date.

While the web crawler and the scheduler script will be executed locally at the host of Nomothesi@'s server, the Library should be accessible at all times, from any place. To accomplish this goal, we will develop a Content Management System (CMS) using a Python Web Development framework, called Django[1] that will provide us with the ability to browse, retrieve and search for specific Government Gazette issues remotely, without requiring our physical presence at the main terminal, where the server is hosted. Since we will be able to scan the library, the web crawler will be integrated into our CMS, to simplify the process of issue retrieval. In case we require the Gazette issues before the scheduler runs and completes its task, we will be capable to fetch them via the CMS.

[1] See: https://www.djangoproject.com/

The last goal of our work is to formulate an ontology, describing in detail the Government Gazette, as an extension to the current Nomothesi@ Ontology. Ergo, this extension will populate our dataset and demonstrate the querying capabilities of the new schema.

1.2 Organization of the thesis

The organization of the thesis is separated in five chapters. In Chapter 2 we present some related work to our project, to show our basis. In Chapter 3 we discuss the background of three different subjects; the subjects that this thesis is revolving around. At first, we will have a glance into the history of the Greek Government Gazette and the National Printing House of Greece, to help us understand the structure of the Gazette issues’. After this topic, we will delve into the record of web crawling and CMS, as well as the importance of task scheduling in Computer Science (CS). The third topic concerns the dynamics of Semantic Web Technologies regarding Greek Legislation.

Furthermore, Chapter 4 furtherly elaborates on our implementation, providing an overview and demonstration. It will help the reader navigate through the platform that this thesis is based on. Moving on, Chapter 5 provides specific details on the Greek Government Gazette’s ontology and Uniform Resource Identifiers (URIs), while also an example of an RDF graph from our dataset. Lastly, we have included some conclusions and information about future projects and works, dealing with a variety of possible occasions.

2. RELATED WORK

Since the project revolving around this thesis involves not only the process automatic retrieval of the daily releases of the Government Gazette's issues, but also managing the library remotely, there have been plenty of research groups around the globe that have been providing and developing frameworks and platforms for similar tasks. These tasks are not confined in being used only on the object of legislation, since remote Content Management Systems and web crawling are of general nature around the internet. The usability of these applications has no borders. The user can select to crawl any kind of data she chooses while pertaining the ability to create a Content Management System for her own personal computer.

On the topic of legislation, many countries around the world have focused on the exact same thing Nomothesi@ is, simplifying legislative documents and serving them to the public to be queried and comfort the public by giving them the material they need without much endeavor. The set of data in their current form (plain text downloaded from the respective National Printing House of each country's website) do not help us to achieve our goals, so they must be transformed to be applicable to web standards. For example, RDF is a great data model in our case.

We will give a case of each element of the thesis (Web crawling, CMS, Government Gazette Representation) to be considered first and then we will move on to analyze our own. These examples will assist and aid the reader of this thesis to understand better the basic concepts of our work. Basic definitions will be given at the beginning of every subsection, while a complete background screening will follow at Chapter 3.

2.1 Software Engineering Aspects

2.1.1 GoogleBot (Web Crawling)

A crawler is a program that visits Web sites and reads their pages and other information in order to create entries for a search engine index. [2] The major search engines on the Web all have such a program, which is also known as a "spider" or a "bot."

Google, being one of the major engines in existence, is utilizing a bot to run automated tasks all over the Internet. The tasks are simple and usually repetitive This specific bot, or as it is called – Googlebot [3], is used to discover new and updated pages to be added to the Google index. This suggests that the software collects documents and data from websites to build a searchable index. Googlebot's crawl process is as follows: a huge set of computers are employed to fetch billions of pages on the web. Then, using an algorithmic process on a list of webpage URLs generated from previous crawls, Googlebot detects links (HREF and SRC) residing on each of these websites and adds them to the general list of pages to crawl. By doing that, the Google index is updated; changes to current active websites are noted, dead links are removed and new sites are appended. After plenty of experiments, there is increasing evidence that Googlebot does not only catch HREF and SRC links but is able to execute JavaScript code and dissect content generated by AJAX calls.

Googlebot crawls every website once every few seconds on average. Having this in mind, it has been reported that it sometimes overwhelms the server's bandwidth, resulting in temporary take downs. Google is providing some tools to assist website owners by letting them throttle the crawling rate of their site. There are also ways to restrict Googlebot from crawling websites, completely. This is achievable by the webmaster. Adding a specific tag into the HTML code of the webpage or excluding specific files and directories from crawling via the robots.txt file. The tag used for exclusion is:

- <meta name="Googlebot" content="nofollow" />

Furthermore, if the website crawled is not properly linked, Googlebot may not be able to discover it at its whole, resulting in pages not getting indexed. This can be resolved by creating and submitting a sitemap. A sitemap is a file that contains the organization of a site's content: the list of all webpages. Using this, Google's crawler will have an easier job moving through it without missing any content.

When someone utilizes the Google Search with a query, the Search looks for matching words and pages in the index, which is continuously updated by Googlebot. The search outcome is defined by a number of factors, one of them being PageRank. PageRank is the measure of importance of a page based on the incoming links from other pages. For example, if Googlebot finds that a million pages link to a specific one, this page will have a huge PageRank and get prioritized when displaying the search results. It is a hard task to achieve because not all links are equal. There are practices and spam links that negatively impact the quality of the search results.

2.1.2 WordPress (Content Management Systems)

Content Management Systems are programs that manage, organize, create and share digital content, such as media and documents [4]. Typically, they are utilized for managing Web or Enterprise content. One of the most widely known Web Content Management Systems is WordPress. While it became popular as a blogging system than a typical CMS, the huge number of plugins and modernization it received made it more CMS-like. It is Open-source and based on PHP and MySQL. As of February 2017, WordPress was used by more than 27.5% of the top 10 million websites. Many features are included like a plugin architecture and a template system. To get WordPress to operate you need to install it on a Web Server, provided by an internet hosting service. There are multiple reasons why WordPress is the leading CMS in the world.

1. It is used by a large pool of people, which make the WordPress community very helpful when in need of support. With the assistance of other users and WordPress being Open Source, while it is feature rich and probably offers everything a user might request out of the box, the 3rd party plugins will handle any demand, relieving the user of any coding procedure.

2. Created in 2003, WordPress has matured over the years. It is stable and is always in active development, with new versions being released continuously.

3. One of the most important considerations into selecting WordPress to manage your website content with, is safety and security. There has been a

number of hacks and security breaches targeted at websites using WordPress but have been patched and solved, as developers have been quick to react and stay on top of them.

2.1.3 jobscheduler (Job Scheduling)

Automation of jobs and tasks in modern day computer systems is of vital importance. Not requiring the physical presence or attention of a computer user to complete an action has be proven to be significantly decisive to the achievement of our goals [5]. A collection of different job scheduling applications exists, with various options regarding the kind of tasks they run and the amount of effort required into managing to understand the way they work. An example of a job scheduling application is the homonym jobscheduler[2]. It is an open source solution for workload automation, capable of executing files and scripts to run various procedures automatically.

JobScheduler stores all of the information generated into a back-end database system. Using priorities and timeslots, the application helps the user manage and organize the order and time of job executions to his demand.

2.2 Government Gazette Semantic Representation

2.2.1 European Legislation Identifier

The European Legislation Identifier (ELI)[3] [4] model is an initiative to improve access to European legislation and to the legislation of its Member States [6]. It is used to identify legislation with a unique identifier which is understandable by humans and computers, compatible with existing technological standards [7]. Also, it interprets legislation with a set of machine-readable metadata elements in compliance with a recommended ontology.

ELI offers a large number of benefits to everyone associated with legislation, as well as citizens in general. Being used all across Europe, it provides easy access to anyone. It reduces costs, promoting the linking and reuse of legal data, decreasing the burden of public administrations.

2.2.2 Akoma Ntoso

Another example of a representation of legislative and judiciary documents in XML format is Akoma Ntoso (Architecture for Knowledge-Oriented Management of African Normative Texts using Open Standards and Ontologies) [8]. The goal is identical to ELIs target. Organizing the legal materials in a way that can be read and understood by computers, to be queried by users. This can be achieved by making the semantic

[2] See: http://www.sos-berlin.com/jobscheduler

[3] See: http://www.eli.fr/en/

[4] See: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISSUM:jl0068

components of legislative documents visible to software applications, so the Official Journals and Government Gazette issues are not just plain undifferentiated text but analyzed to the core of their structure and have useful information extracted from them.

2.3 Conclusion

With our project revolving around three basic subjects (Web Crawling, CMSs, Semantic Representation of Government Gazette), the amount of related work in each sector is extensive. Each work presented in each category is intended to help us understand better the target and aim of our own. Googlebot helps us understand the way web crawlers work and our thought process while developing our own. Workbench and Jobscheduler show some basic features and ways to achieve great organization of files and documents and to automate tasks and functions. Inspired by all these related examples, we developed our own versions of a Spider, a CMS and a Scheduler. ELI and Akoma Ntoso gave us a great understanding of Semantic Representation of legislative documents, helping us generate our own ontology in Chapter 5.

3. BACKGROUND

In this Chapter, we are going to examine the backdrop of the many different elements that compose this thesis and give general information about them. Starting from the history of the Greek Government Gazette and the responsibilities of the National Printing House, moving on to discuss the great benefits that Web Crawling and CMS' offer to Computer Science, as well as all the advantages in semantic data designs, their dynamics and the future of Semantic Web.

3.1 Greek Government Gazette

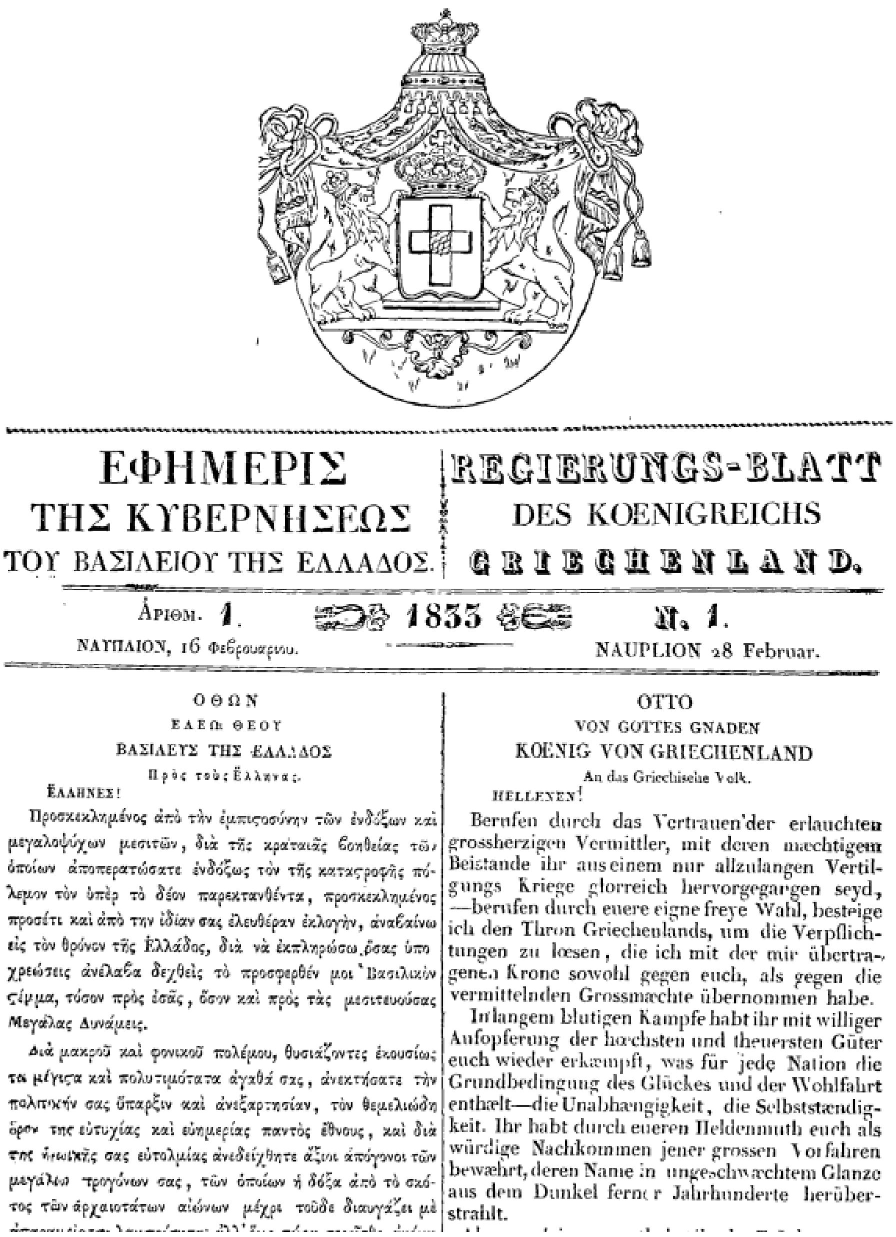
The National Printing House is a public service under the supervision of the Ministry of Administrative Reform and Electronic Government. It is housed in Athens and its main responsibility is the publishing of the Government Gazette [9]. By doing so, either electronically via its official website[5], either via the distribution of printed Gazette Issues, it provides the citizens with the ability to access the documents published in it. Furthermore, the National Printing House covers all the printing needs of the public services and also the printing of studies of scientists, scientific or research centers or institutes or other bodies of particular interest to the public and relating to political, social and administrative matters.

In this thesis we mention the Government Gazette a good deal of times, and it is because of the importance of these documents and the significant impact they have on our lives. The Greek Government Gazette, the document released by the National Printing House as mentioned above, is the only official instrument in the Constitution of Greece to publish the legal acts of the institutions of the Hellenic Republic, as well as the legal entities of public and private law, through which they become active.

3.1.1 Historical Introduction

Before the birth of the Government Gazette, there were some other documents that fulfilled the same purpose and had the same character as it and frequently devoted all their content to the official decisions of the state. Some examples are “The Friend of the State” (1824-1825), the “General Gazette of Greece” (1825-1832). The first ever Greek Government Gazette issue to be published was in 28 February 1833. It was released by King Otto’s Regency. The Government Gazette of the Kingdom of Greece as it was called, was printed in Nafplio until 5 November 1834 and after 21 of December 1834 continued its issuance in Athens, by the National Printing House until today. During Otto’s Regency, the Gazette of the Greek Kingdom was bilingual, in Greek and in German.

[5] See: www.et.gr



ΕΦΗΜΕΡΙΣ ΤΗΣ ΚΥΒΕΡΝΗΣΕΩΣ ΤΟΥ ΒΑΣΙΛΕΙΟΥ ΤΗΣ ΕΛΛΑΔΟΣ.

REGIERUNGS-BLATT DES KOENIGREICHS GRIECHENLAND.

Ἀριθμ. 1. 1833 N. 1.

ΝΑΥΠΛΙΟΝ, 16 Φεβρουαρίου. NAUPLION 28 Februar.

ΟΘΩΝ
ΕΛΕΩ ΘΕΟΥ
ΒΑΣΙΛΕΥΣ ΤΗΣ ΕΛΛΑΔΟΣ
Πρὸς τοὺς Ἕλληνας.

ἝΛΛΗΝΕΣ!

Προσκεκλημένος ἀπὸ τὴν ἐμπιστοσύνην τῶν ἐνδόξων καὶ μεγαλοψύχων μεσιτῶν, διὰ τῆς κραταιᾶς βοηθείας τῶν ὁποίων ἀποπερατώσατε ἐνδόξως τὸν τῆς καταστροφῆς πόλεμον τὸν ὑπὲρ τὸ δέον παρεκτανθέντα, προσκεκλημένος προσέτι καὶ ἀπὸ την ἰδίαν σας ἐλευθέραν ἐκλογὴν, ἀναβαίνω εἰς τὸν θρόνον τῆς Ἑλλάδος, διὰ νὰ ἐκπληρώσω ὅσας ὑποχρεώσεις ἀνέλαβα δεχθεὶς τὸ προσφερθέν μοι Βασιλικὸν στέμμα, τόσον πρὸς ἐσᾶς, ὅσον καὶ πρὸς τὰς μεσιτευούσας Μεγάλας Δυνάμεις.

Διὰ μακροῦ καὶ φονικοῦ πολέμου, θυσιάζοντες ἑκουσίως τὰ μέγιστα καὶ πολυτιμότατα ἀγαθά σας, ἀνεκτήσατε τὴν πολιτικήν σας ὕπαρξιν καὶ ἀνεξαρτησίαν, τὸν θεμελιώδη ὅρον της εὐτυχίας καὶ εὐημερίας παντὸς ἔθνους, καὶ διὰ της ἡρωϊκῆς σας εὐτολμίας ἀνεδείχθητε ἄξιοι ἀπόγονοι τῶν μεγάλων προγόνων σας, τῶν ὁποίων ἡ δόξα ἀπὸ τὸ σκότος τῶν ἀρχαιοτάτων αἰώνων μέχρι τοῦδε διαυγάζει μὲ

OTTO
VON GOTTES GNADEN
KOENIG VON GRIECHENLAND
An das Griechische Volk.

HELLENEN!

Berufen durch das Vertrauen der erlauchten grossherzigen Vermittler, mit deren mæchtigem Beistande ihr aus einem nur allzulangen Vertilgungs Kriege glorreich hervorgegangen seyd, —berufen durch euere eigne freye Wahl, besteige ich den Thron Griechenlands, um die Verpflichtungen zu lœsen, die ich mit der mir übertragenen Krone sowohl gegen euch, als gegen die vermittelnden Grossmæchte übernommen habe.

In langem blutigen Kampfe habt ihr mit williger Aufopferung der hœchsten und theuersten Güter euch wieder erkæmpft, was für jede Nation die Grundbedingung des Glückes und der Wohlfahrt enthælt—die Unabhængigkeit, die Selbststændigkeit. Ihr habt durch eueren Heldenmuth euch als würdige Nachkommen jener grossen Vorfahren bewæhrt, deren Name in ungeschwæchtem Glanze aus dem Dunkel ferner Jahrhunderte herüberstrahlt.

Figure 3. 1: First ever Gazette release

3.1.2 Current Organization of the Greek Government Gazette

Keeping the character of its ancestors intact, the Gazette publishes laws and legal acts. It is published in twelve issues or categories and their numeration begins at 1 at the beginning of each year. Each document has it is own issue/year/number combination and can be identified using that. Out of the twelve different issues, we will go into more details for the first four, with them being the most general and affecting the majority of citizens. The issues are numbered One, Two, Three and Four but are represented in Greek numbering, resulting in Issue one being Issue A, two being B, three being Γ and four being Δ. We will also present nominally the remaining 8 Gazette issues and review their content with a few words. Until 2006, there have been three issues that have been removed and their content has been placed into issues B and Γ.

The contents that each issue reviews are the following, as listed in the official website of the National Printing House:

3.1.2.1 Issue A:

a) the Constitution and the laws adopted and ratified in accordance therewith, as well as the international conventions or agreements of the Country with other States or organizations and the communications relating to such contracts or agreements,
b) the Rules of Procedure of the House of Representatives, as well as the acts of the Parliament or its President, which are published in accordance with the provisions of its Rules of Procedure,
c) the President's Protocol of Inauguration, the Presidential Appointments of the President, the Acts of Legislative Content issued pursuant to Article 44 (1) of the Constitution and the decrees for the announcement of referendums pursuant to Article 44 (2) of the Constitution,
d) Presidential decrees with normative content and presidential decrees issued in accordance with Articles 37, 38, 40, 41 and 43 (1) of the Constitution,
e) the acts of the Council of Ministers, the publication of which is provided for in the applicable legislation.

3.1.2.2 Issue B:

a) decrees approving the establishment of charities and modifying their organizations,
b) the regulations of the Prime Minister, Ministers, Deputy Ministers, as well as regulatory decisions of other Bodies of the Administration, provided that the existing legislation does not provide for public disclosure by another means and not otherwise provided for in this law,
c) the Bank of Greece Governor's Operations and the decisions of the Banking and Credit Committee of the Bank of Greece, which have a regulatory content,
d) ship registration decisions,
e) in summary, decrees and other acts relating to investment in development laws, conventions relating to development laws, decisions on the commencement of productive operation of investments and other acts referred to in development laws,
f) decisions approving the establishment of offices of foreign companies in Greece,
g) the acceptance of donations by the Greek State,
h) any other published act, the registration of which is not foreseen in another Issue.

3.1.2.3 Issue C (Γ):

a) in summary, individual decrees and acts of appointment, transfer, demotion, acceptance of resignation and dismissal of public servants, civilian or military officials,
b) in summary, acts appointing and accepting the resignation or termination of lawyers, notaries and chartered accountants,

c) in summary, acts of appointment and other changes of the Holy Clergy and of the staff of ecclesiastical organizations whose publication is required under the applicable legislation,
d) the announcement of positions of Teaching Researchers of Higher Education Institutions, the examination of candidates for attorneys and notaries,
e) the notices referred to in Article 3 (9) of Law 3429/2005 for the selection of directors of public enterprises and organizations,
f) the decisions of the disciplinary councils, which are required by law to be published in the "Government Gazette"
g) in summary, the Presidential Decrees of Grace or the Removal of Consequences of Condemnation, as well as Presidential Delegations of Flag or Decree.

3.1.2.4 Issue D (Δ):

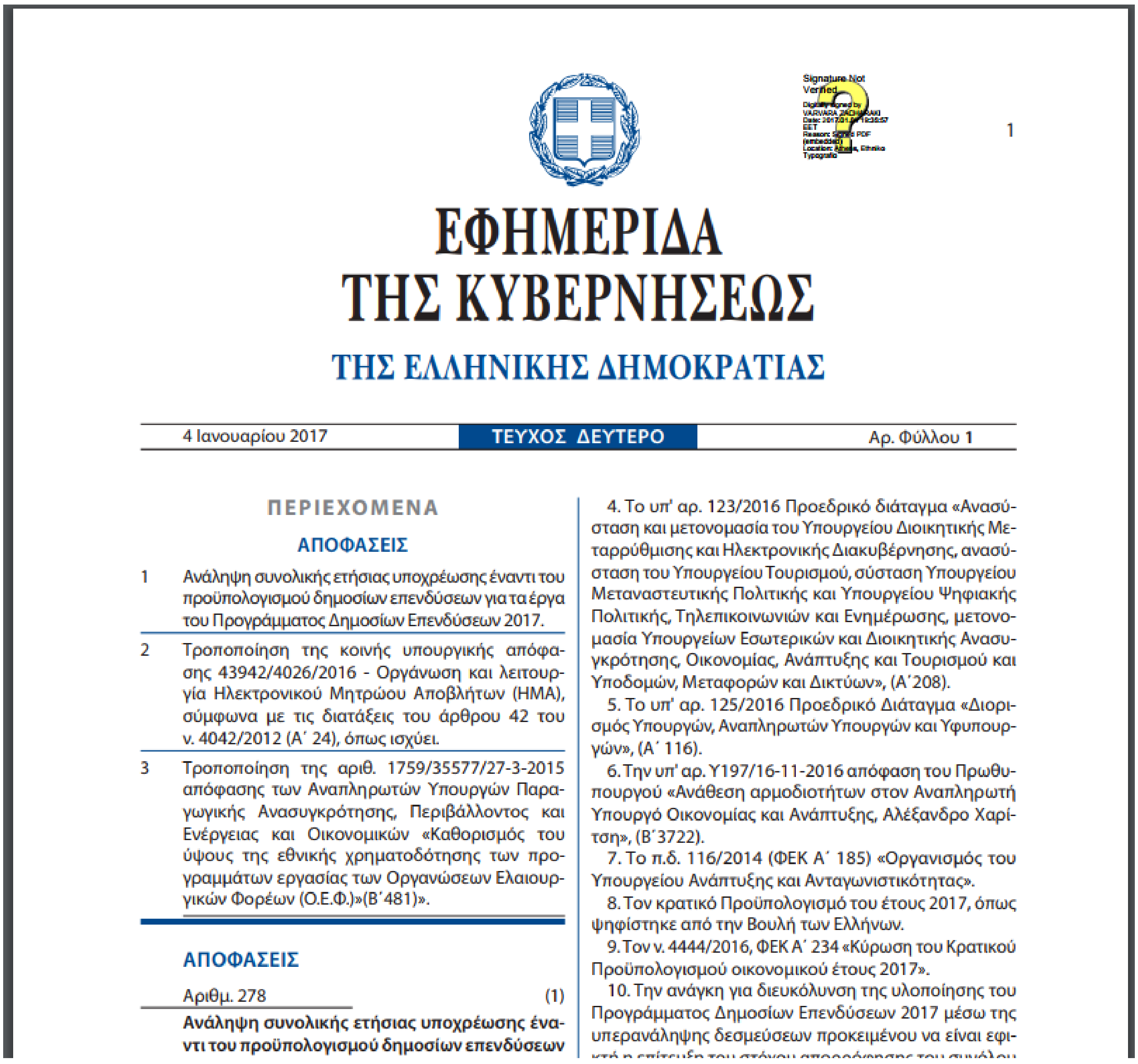
a) operations to declare extensible land as well as acts for the total or partial revocation of such operations,
b) acts of concession of public property,
c) acts for the definition of shoreline, beach and old shoreline and operations for defining lakes and rivers, laying down river and streamlines, industrial zones, national parks, public forests and forest areas in general,
d) acts relating to wildlife hideouts,
e) operations relating to the management of sewage,
f) any other act in relation to those referred to in the previous cases.

3.1.2.5 Remaining Issues

While the four major issues mentioned before cover a wide area of subjects, the remaining 8 issues examine more specific topics. Their issue names are initials in Greek, explaining the content they encompass. They are distinguished in the following way (Table 3.1)

Table 3. 1: Issue Names and Subjects

Issue Name	Subject
Α.Α.Π.	Forced expropriations and town planning issues
Α.ΕΙ.Δ	Supreme Special Court
Α.Σ.Ε.Π.	Statement of the Supreme Personnel Selection Board
ΠΡΑ.Δ.Ι.Τ.	Registration of transactions and data from other public and private sector entities
Δ.Δ.Σ.	Public Procurement notices
Ο.Π.Κ.	Finance of Political Parties and Political Parties Coalition
Υ.Ο.Δ.Δ.	Special Employees and Administration Bodies of Public and Wider Public-Sector Bodies
ΑΕ - ΕΠΕ / ΓΕΜΗ	Société Anonymes - Limited Liability Companies and General Commercial Registry



Signature Not Verified

Digitally signed by VARVARA ZACHARAKI
Date: 2017.01.04 18:35:57 EET
Reason: Signed PDF (embedded)
Location: Athens, Ethniko Typografio

1

ΕΦΗΜΕΡΙΔΑ ΤΗΣ ΚΥΒΕΡΝΗΣΕΩΣ

ΤΗΣ ΕΛΛΗΝΙΚΗΣ ΔΗΜΟΚΡΑΤΙΑΣ

4 Ιανουαρίου 2017 | ΤΕΥΧΟΣ ΔΕΥΤΕΡΟ | Αρ. Φύλλου 1

ΠΕΡΙΕΧΟΜΕΝΑ

ΑΠΟΦΑΣΕΙΣ

1 Ανάληψη συνολικής ετήσιας υποχρέωσης έναντι του προϋπολογισμού δημοσίων επενδύσεων για τα έργα του Προγράμματος Δημοσίων Επενδύσεων 2017.

2 Τροποποίηση της κοινής υπουργικής απόφασης 43942/4026/2016 - Οργάνωση και λειτουργία Ηλεκτρονικού Μητρώου Αποβλήτων (ΗΜΑ), σύμφωνα με τις διατάξεις του άρθρου 42 του ν. 4042/2012 (Α΄ 24), όπως ισχύει.

3 Τροποποίηση της αριθ. 1759/35577/27-3-2015 απόφασης των Αναπληρωτών Υπουργών Παραγωγικής Ανασυγκρότησης, Περιβάλλοντος και Ενέργειας και Οικονομικών «Καθορισμός του ύψους της εθνικής χρηματοδότησης των προγραμμάτων εργασίας των Οργανώσεων Ελαιουργικών Φορέων (Ο.Ε.Φ.)»(Β΄481)».

ΑΠΟΦΑΣΕΙΣ

Αριθμ. 278 (1)

Ανάληψη συνολικής ετήσιας υποχρέωσης έναντι του προϋπολογισμού δημοσίων επενδύσεων

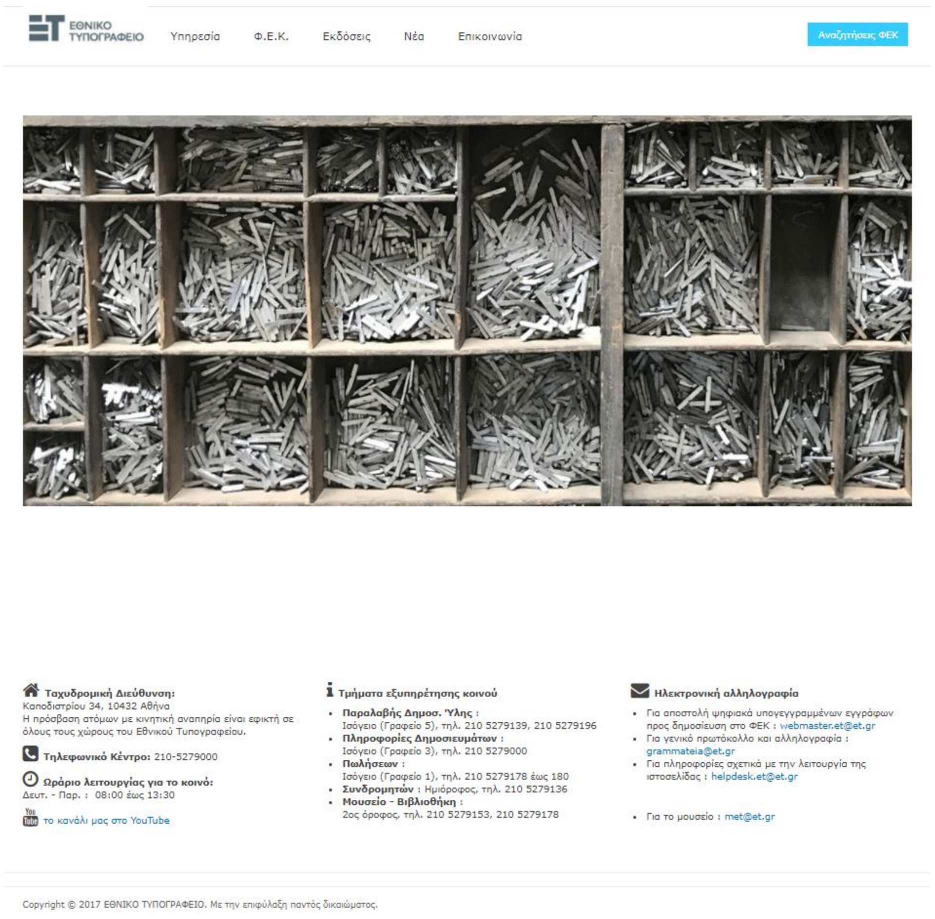
4. Το υπ' αρ. 123/2016 Προεδρικό διάταγμα «Ανασύσταση και μετονομασία του Υπουργείου Διοικητικής Μεταρρύθμισης και Ηλεκτρονικής Διακυβέρνησης, ανασύσταση του Υπουργείου Τουρισμού, σύσταση Υπουργείου Μεταναστευτικής Πολιτικής και Υπουργείου Ψηφιακής Πολιτικής, Τηλεπικοινωνιών και Ενημέρωσης, μετονομασία Υπουργείων Εσωτερικών και Διοικητικής Ανασυγκρότησης, Οικονομίας, Ανάπτυξης και Τουρισμού και Υποδομών, Μεταφορών και Δικτύων», (Α΄208).
5. Το υπ' αρ. 125/2016 Προεδρικό Διάταγμα «Διορισμός Υπουργών, Αναπληρωτών Υπουργών και Υφυπουργών», (Α΄ 116).
6. Την υπ' αρ. Υ197/16-11-2016 απόφαση του Πρωθυπουργού «Ανάθεση αρμοδιοτήτων στον Αναπληρωτή Υπουργό Οικονομίας και Ανάπτυξης, Αλέξανδρο Χαρίτση», (Β΄3722).
7. Το π.δ. 116/2014 (ΦΕΚ Α΄ 185) «Οργανισμός του Υπουργείου Ανάπτυξης και Ανταγωνιστικότητας».
8. Τον κρατικό Προϋπολογισμό του έτους 2017, όπως ψηφίστηκε από την Βουλή των Ελλήνων.
9. Τον ν. 4444/2016, ΦΕΚ Α΄ 234 «Κύρωση του Κρατικού Προϋπολογισμού οικονομικού έτους 2017».
10. Την ανάγκη για διευκόλυνση της υλοποίησης του Προγράμματος Δημοσίων Επενδύσεων 2017 μέσω της υπερανάληψης δεσμεύσεων προκειμένου να είναι εφι-

Figure 3. 2: Contemporary Greek Government Gazette document (Issue B)

3.1.3 ET.gr, the Greek Government Gazette in the age of the World Wide Web

As the years passed and the age of computers was upon us, the need for a digital release of the Government Gazette's documents was imperative. Hence, in 1997 the National Printing House's website, www.et.gr came online. The documents were published online as well and by 2010 their release became free and open to the public.

ET.gr offers a great amount of services. Besides the historical information of the National Printing House and the statistics regarding the Government Gazette, it also includes a Daily Release section. In this specific section you can browse the Gazette releases by their publish date, sorted by issues. Furthermore, a more advanced search function is included that allows the user to search by issue, Gazette ID or even specific keywords. While these functionalities are useful and advantageous, there are many concerns revolving around them. Specially, many older documents, released in 1980s and before, while existing in the database, they are in a scanned image form. Therefore, no search function can be practiced on them. Moreover, the search function operated on newer releases is totally built on the Gazettes and not the legislation, which would be the case of real benefit to the public.



Ταχυδρομική Διεύθυνση:
Καποδιστρίου 34, 10432 Αθήνα
Η πρόσβαση ατόμων με κινητική αναπηρία είναι εφικτή σε όλους τους χώρους του Εθνικού Τυπογραφείου.

Τηλεφωνικό Κέντρο: 210-5279000

Ωράριο λειτουργίας για το κοινό:
Δευτ. - Παρ. : 08:00 έως 13:30

το κανάλι μας στο YouTube

Τμήματα εξυπηρέτησης κοινού

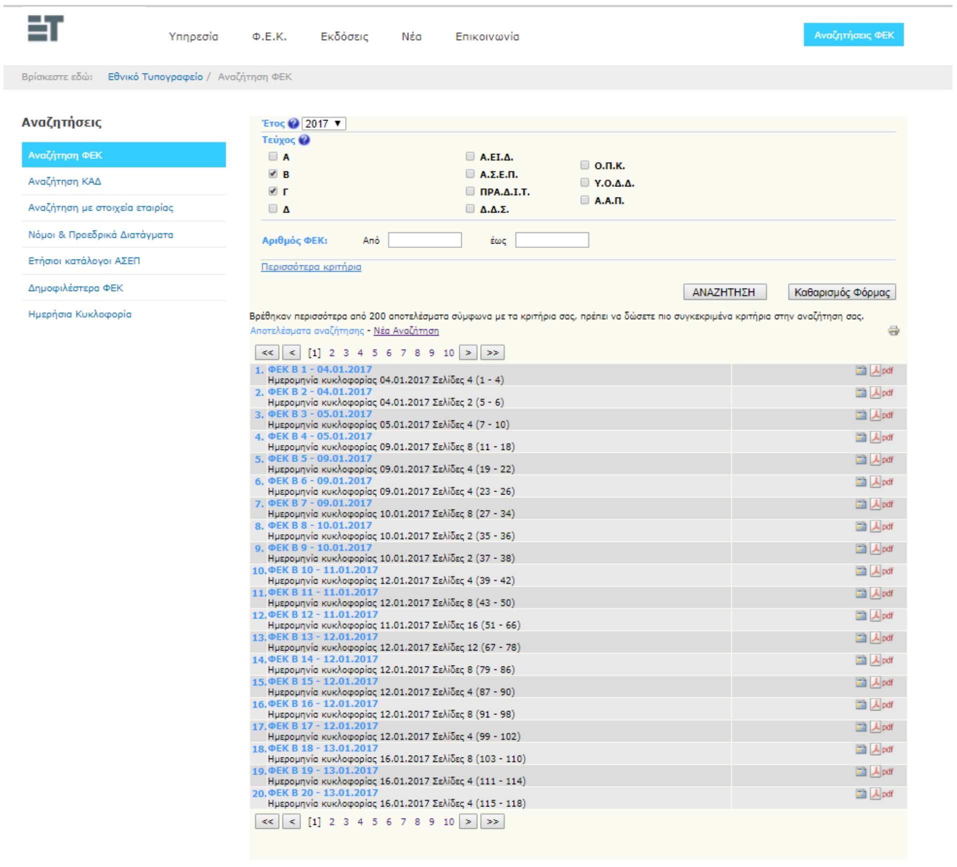
- **Παραλαβής Δημοσ. Ύλης :**
  Ισόγειο (Γραφείο 5), τηλ. 210 5279139, 210 5279196
- **Πληροφορίες Δημοσιευμάτων :**
  Ισόγειο (Γραφείο 3), τηλ. 210 5279000
- **Πωλήσεων :**
  Ισόγειο (Γραφείο 1), τηλ. 210 5279178 έως 180
- **Συνδρομητών :** Ημιόροφος, τηλ. 210 5279136
- **Μουσείο - Βιβλιοθήκη :**
  2ος όροφος, τηλ. 210 5279153, 210 5279178

Ηλεκτρονική αλληλογραφία

- Για αποστολή ψηφιακά υπογεγραμμένων εγγράφων προς δημοσίευση στο ΦΕΚ : webmaster.et@et.gr
- Για γενικό πρωτόκολλο και αλληλογραφία : grammateia@et.gr
- Για πληροφορίες σχετικά με την λειτουργία της ιστοσελίδας : helpdesk.et@et.gr
- Για το μουσείο : met@et.gr

Copyright © 2017 ΕΘΝΙΚΟ ΤΥΠΟΓΡΑΦΕΙΟ. Με την επιφύλαξη παντός δικαιώματος.

Figure 3. 3: ET.gr



Έτος 2017

Τεύχος

- [ ] Α
- [✓] Β
- [✓] Γ
- [ ] Δ
- [ ] Α.ΕΙ.Δ.
- [ ] Α.Σ.Ε.Π.
- [ ] ΠΡΑ.Δ.Ι.Τ.
- [ ] Δ.Δ.Σ.
- [ ] Ο.Π.Κ.
- [ ] Υ.Ο.Δ.Δ.
- [ ] Α.Α.Π.

Αριθμός ΦΕΚ: Από ____ έως ____

Περισσότερα κριτήρια

ΑΝΑΖΗΤΗΣΗ | Καθαρισμός Φόρμας

Βρέθηκαν περισσότερα από 200 αποτελέσματα σύμφωνα με τα κριτήρια σας, πρέπει να δώσετε πιο συγκεκριμένα κριτήρια στην αναζήτηση σας.
Αποτελέσματα αναζήτησης - Νέα Αναζήτηση

<< < [1] 2 3 4 5 6 7 8 9 10 > >>

1. ΦΕΚ Β 1 - 04.01.2017
Ημερομηνία κυκλοφορίας 04.01.2017 Σελίδες 4 (1 - 4)
2. ΦΕΚ Β 2 - 04.01.2017
Ημερομηνία κυκλοφορίας 04.01.2017 Σελίδες 2 (5 - 6)
3. ΦΕΚ Β 3 - 05.01.2017
Ημερομηνία κυκλοφορίας 05.01.2017 Σελίδες 4 (7 - 10)
4. ΦΕΚ Β 4 - 05.01.2017
Ημερομηνία κυκλοφορίας 09.01.2017 Σελίδες 8 (11 - 18)
5. ΦΕΚ Β 5 - 09.01.2017
Ημερομηνία κυκλοφορίας 09.01.2017 Σελίδες 4 (19 - 22)
6. ΦΕΚ Β 6 - 09.01.2017
Ημερομηνία κυκλοφορίας 09.01.2017 Σελίδες 4 (23 - 26)
7. ΦΕΚ Β 7 - 09.01.2017
Ημερομηνία κυκλοφορίας 10.01.2017 Σελίδες 8 (27 - 34)
8. ΦΕΚ Β 8 - 10.01.2017
Ημερομηνία κυκλοφορίας 10.01.2017 Σελίδες 2 (35 - 36)
9. ΦΕΚ Β 9 - 10.01.2017
Ημερομηνία κυκλοφορίας 10.01.2017 Σελίδες 2 (37 - 38)
10. ΦΕΚ Β 10 - 11.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 4 (39 - 42)
11. ΦΕΚ Β 11 - 11.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 8 (43 - 50)
12. ΦΕΚ Β 12 - 11.01.2017
Ημερομηνία κυκλοφορίας 11.01.2017 Σελίδες 16 (51 - 66)
13. ΦΕΚ Β 13 - 12.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 12 (67 - 78)
14. ΦΕΚ Β 14 - 12.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 8 (79 - 86)
15. ΦΕΚ Β 15 - 12.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 4 (87 - 90)
16. ΦΕΚ Β 16 - 12.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 8 (91 - 98)
17. ΦΕΚ Β 17 - 12.01.2017
Ημερομηνία κυκλοφορίας 12.01.2017 Σελίδες 4 (99 - 102)
18. ΦΕΚ Β 18 - 13.01.2017
Ημερομηνία κυκλοφορίας 16.01.2017 Σελίδες 8 (103 - 110)
19. ΦΕΚ Β 19 - 13.01.2017
Ημερομηνία κυκλοφορίας 16.01.2017 Σελίδες 4 (111 - 114)
20. ΦΕΚ Β 20 - 13.01.2017
Ημερομηνία κυκλοφορίας 16.01.2017 Σελίδες 4 (115 - 118)

<< < [1] 2 3 4 5 6 7 8 9 10 > >>

Figure 3. 4: ET.gr search page

3.2 Web Retrieval and CMS

After our overview of the Greek Government Gazette’s structure and content information, we will now proceed to the more technical and Computer Science related part of this thesis. We will analyze the way Web Crawlers work as well as the level of organization we can achieve via Content Management Systems.

3.2.1 Web Crawling

As we explained during our Googlebot Example in Chapter 2.1.1, crawling the web is an essential part for feeding search engines with web-pages, enabling the engines to index the web-pages and making them searchable. They have been the leading approach to indexing since the very early days of search engines, dating back to 1995 and the birth of the AltaVista web crawler. As the number of webpages increases dramatically day by day, crawlers have had a hard job on making a complete index of all the webpages that existed back in the day. Googlebot, Bingbot and other modern and up to date web crawlers helped search engines to improve vastly.

After the year 2000, where programming and coding became easier to do at home with the introduction of easy to access and purchase personal computers and the addition of amateur-friendly programming languages like Python, web crawlers benefited users in a more direct manner; they were coded, and still are, to visit multiple websites and gather information and documents with a press of a button. [11] [12]

The way web crawlers, or as they are called, “spiders” work, is the following. Given a list of hyperlinks, known as “seed”, they begin their journey following all the other hyperlinks they come across, or they look for specific information based on HTML and CSS elements. For example, a crawler can be setup to look for picture files (JPG/PNG) or portable document files (PDF) that exist in a webpage. Of course, this is the method crawlers are also used by search engines for indexing. Crawling websites and discovering all the links that reside in the website, following them and indexing keywords and material to become searchable by the public. This function is repeated every few minutes to ensure the best results and avoid dead links. [11] [12]

With a first glance, web crawlers are pretty straight forward; visit websites and gather the material needed. Looking deeper into the subject, despite the coding issues that may occur to anyone, crawlers should follow specific policies that determine the behavior of the script. The policies are the following:

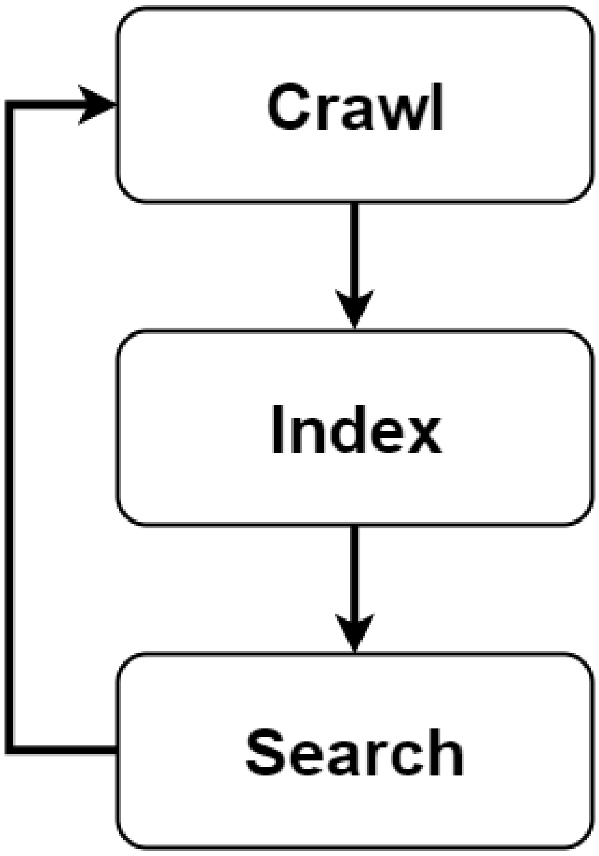


Figure 3. 5: Indexing process used by search engine crawlers.

Table 3. 2: Crawler Policies

Policy	**Description**
Selection Policy	Which pages are downloaded
Re-visit Policy	When to inspect for content changes
Politeness Policy	How to avoid overloading webpages
Parallelization policy	How to coordinate distributed crawlers

As the need for information and data by companies and individuals rises, many web crawling products have surfaced over the last few years. They are called “Visual web crawlers”, and will crawl pages and structure data into tables, based on the user’s demands. The difference between the classic, programmatic crawlers and their visual counterpart is that Visual Crawlers require no programming ability to be used. Inputting the data you desire, will return a list of results but without any effort. Of course, these services come with a price, since a paid subscription is required to use them, in contrast with the open source programmatic crawlers that can be coded at any time, without any cost.

3.2.2 CMS

Modern day Computer Systems have large directories crowded with data of any form, from text files such as electronic documents to multimedia files like video or audio files.

This huge amount of information needs to be managed for the system user not to be lost and overwhelmed by it. Content Management (CM), is a set of processes and technologies that supports the collection, managing, and publishing of information in any form or medium. CM is a collaborative process, often consisting of the following roles and responsibilities:

Table 3. 3: Roles and responsibilities in Content Management

Role	Responsibility
Creator	Content Creation
Editor	Content Tuning
Publisher	Content Release
Administrator	Managing permissions
Viewer	Read/View Content

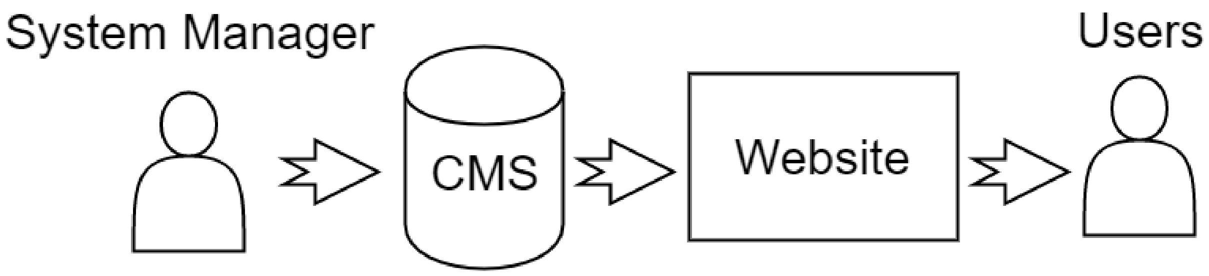


Figure 3. 6: CMS Overview

A Content Management System (CMS) is a computer system or application that allows everything we noted above about Content Management, as well as site maintenance from a webpage. They are also known as Web Content Management Systems (WCMS). [4] We have to note that not all CMS are WCMS, but the most popular and widely used, are. So, when we talk about CMS, we will be referring to WCMS as well. CMSs were created to develop and present information about content stored on a web server, on a website. From simple preview of files to more complex functions, CMS can be modified accordingly to the user's and website's needs. CMS can offer control access to data based on the user's role. For example, being member of a specific user group could prevent you from editing or even viewing some files that members of other groups can modify and retrieve.

CMS consist of two major components:

a) A content management application (CMA) is the front-end user interface that allows a user, even with limited expertise, to add, modify, and remove content from a website without the intervention of a webmaster.
b) A content delivery application (CDA) compiles that information and updates the website.

In most CMS scenarios, especially in already existing ones, files are stored into databases, usually SQL. Sometimes, if the project associated with the CMS is minor or insignificant, the files can be served from local directories, without even being saved into a database.

3.3 The dynamics of RDF Data Models, OWL Ontologies and SPARQL - Semantic Perspectives

World Wide Web has been designed in order to offer information understood by human beings. It consists by billions of documents, mainly HTML pages, interlinked via “lousy” hyperlinks. Current architecture offers very limited opportunities for querying and retrieving information tasks by machines. Semantic Web provides a state of the art framework that allows data to be shared and reused across application, enterprise, and community boundaries with great respect in machine automated processing. This framework has three crucial elements: the RDF data model, the OWL ontology and SPARQL, the query language for RDF data sets. [18]

3.3.1 The new Data Model with Resource Description Framework (RDF)

A data model is defined as “the data items of a certain part of the perceived reality (business domain) relevant for a specific application or a specific user in a structured way. This model includes the data relationships” [10]. Several data models have been implemented for management of changes in legislative document systems. But how our data model enables us to observe the versions or the metadata of a legal document and solve the issues mentioned in previous Sections. The Resource Description Framework (RDF) is a framework for representing information in the web. The core structure of the abstract syntax is a set of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph. An RDF graph can be visualized as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link. RDF data comply on RDF Schema, a semantic extension which mainly support classification and describe the interactions between resources. Conceptualizing legislation in a machine-readable format, RDF obviously seems to be a perfect fit. But before we can start talking about our new RDF data model, let us see why our old XML schema does not suits the purposes of our goals from now on and in the future. While the first problems addressed with semantic information like, the need to distinguish between information content and presentation in a legal document, have been dealt with XML (XML tags used to express the “semantics” of various pieces of information), different legal documents may represent in different ways semantically related pieces of information.

That brings to the surface a new problem that XML schemas and data models could not solve. So, such different XML documents might not share such common semantics. RDF in the contrary, is a data model that can express the “meaning” of such a piece of information in a shared way. With dropping the XML technology from our data model, we immediately dropped problems like querying the same semantics represented in different XML trees or the need of converting of all possible representations of a fact into one statement. RDF gives us a standard way of writing statements. Finally, we end up with a unifying model not only to itself but open to other data models as well, immediate expandable by adding simple statements without re-designing the whole model and fast in terms of querying and mining information. So in this project unlike the old version of Nomothesi@ we are trying a new approach. Instead of using RDF to store metadata for Greek Legislation, we are also using it to shape the whole model for two main reasons. First of all, to query them and infer the appropriate knowledge needed for advanced services. Secondly to publish our data sets and link those to third-party data sets across web.

3.3.2 OWL Ontologies and SPARQL

Web Ontology Language (OWL) designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to infer implicit knowledge explicit. Legislation is all about consistency and implicit knowledge, semantic information spelled in words, ruined in text documents. OWL documents, known as ontologies, can be published in the World Wide Web and may refer to or be referred from other OWL ontologies. The data described by an ontology in the OWL is interpreted as a set of “individuals” and a set of “property assertions” which relate these individuals to each other. An ontology consists of a set of axioms which place constraints on sets of individuals (called “classes”) and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided. We are certain to believe that we have established a strong and efficient data model given all the advantage above using RDF, RDFS and OWL. Now that we have added semantics to our RDF data, we must consider how is it published and queried. SPARQL is an RDF query language, that is, an SQL-like semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middle-ware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions, it also supports extensible value testing and constraining queries by source RDF graph. For the very first time in Greek Legislation and worldwide, our project innovates with the idea of serving a SPARQL Endpoint in a RESTful manner. Users can now take advantage of legal documents not only by reading them, but also querying all the semantic information. This forms a new path in the way legislation is being served and represented by any organization or official institution and we hope that many other will follow our way until we have unified government archives of all types. At this point we would like to present some technical information regarding the semantic information available from our Endpoint. The main query operations in this system are:

a) Obtaining the legislative document valid at a given date or a time space.
b) Historical evolution of a legislative document.
c) Full text search in the text of the legislative document.
d) Laws repealed by a law.
e) Obtaining all kinds of Metadata (the model allows new Metadata to be easily included).
f) Description queries on the RDF graph.

3.3.3 Semantic perspectives and future Web

The Semantic Web is a vision for the future of the web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the web. As Sir Tim Berners-Lee puts it:

“The concept of machine-understandable documents does not imply artificial intelligence which allows machines to comprehend human mumbling. It only indicates a machine’s ability to solve a well-defined problem by performing well-defined operation on existing well-defined data.”

We have to be then, as more efficient as it gets when we are about to represent complex documents on the web, and especially more careful when we talk about legislation. The legal systems are usually developed in several databases and formats that require integration. RDF offers the possibility of building a simple semantic data model with several advantages over traditional data models for legal systems. So, RDF helps to merge data from multiple sources and does it in such a way that data about the same things gets collated. This merge comes with the benefit of enabling cross-data source querying using SPARQL as we saw earlier, allowing content or relationships between content from several datasets to be discovered. Properties and values can be described with a shared schema or ontology, taking advantage of RDF in legal systems. But most importantly, this new era of modeling legislation semantically helps us develop a Restful API to serve different versions of a legislative document in different formats at the same URI, to give developers full and open access to the data, which concentrates the vision of this project.

3.4 Semantic Representation of Government Gazette

Regarding the Semantic Representation of Government Gazette documents, a complex model does not exist. Taking into consideration the two standards we introduced beforehand, ELI and Akoma Ntoso, we cannot represent Gazette information using their vocabulary and classes. Specifically, ELI, offers two properties (relations) to help illustrate Government Gazette details.

1. The Object Property “published_in”, that connects a Legal Resource format with an Official Gazette (Government Gazette) format.
2. The Data Property “published_in”, that associates a Legal Resource format with a string.

On the other hand, Akoma Ntoso’s take on Government Gazette representation is even more limited. To be exact, Akoma Ntoso’s vocabulary only offers one class that assists us, the “OfficialGazette” class.

3.5 Conclusion

In this Chapter we introduced the reader of this thesis to all the basic knowledge required to understand our work and our goals. Explaining the history and structure of the Greek Government Gazette provides assistance into the understanding of the Semantic Representation. We also explained how crawling and spiders function, clarifying some key components of a spider’s architecture, which will make the explanation of our own very straightforward. Last but not least, deconstructing a Content Management System like we did above will help us finding out which features we need for our own CMS.

4. GREEK GOVERNMENT GAZETTE LIBRARY

After a long introduction and some background information, now that we understand the concept of extracting documents and data via web crawling and managing our content through a Content Management System, we can proceed to explaining our project. Throughout this Chapter we will analyze and illustrate the main modules constituting our crawler and CMS while demonstrating the basic features.

4.1 Greek Government Gazette Crawler

The need of keeping Nomothesi@'s library up to date with the daily releases of Greek Government Gazette's issues required us to develop a program that can communicate with the National Printing House's website and retrieve old and new Gazette documents, adding them to the library. Web crawling is a very practical and straightforward way of achieving this goal. While using a Visual Web Crawler with a paid subscription would have been the easy way, developing and modifying the crawler to our needs and standards was a top priority, so we could retrieve exactly the type of data we wish, without any payment.

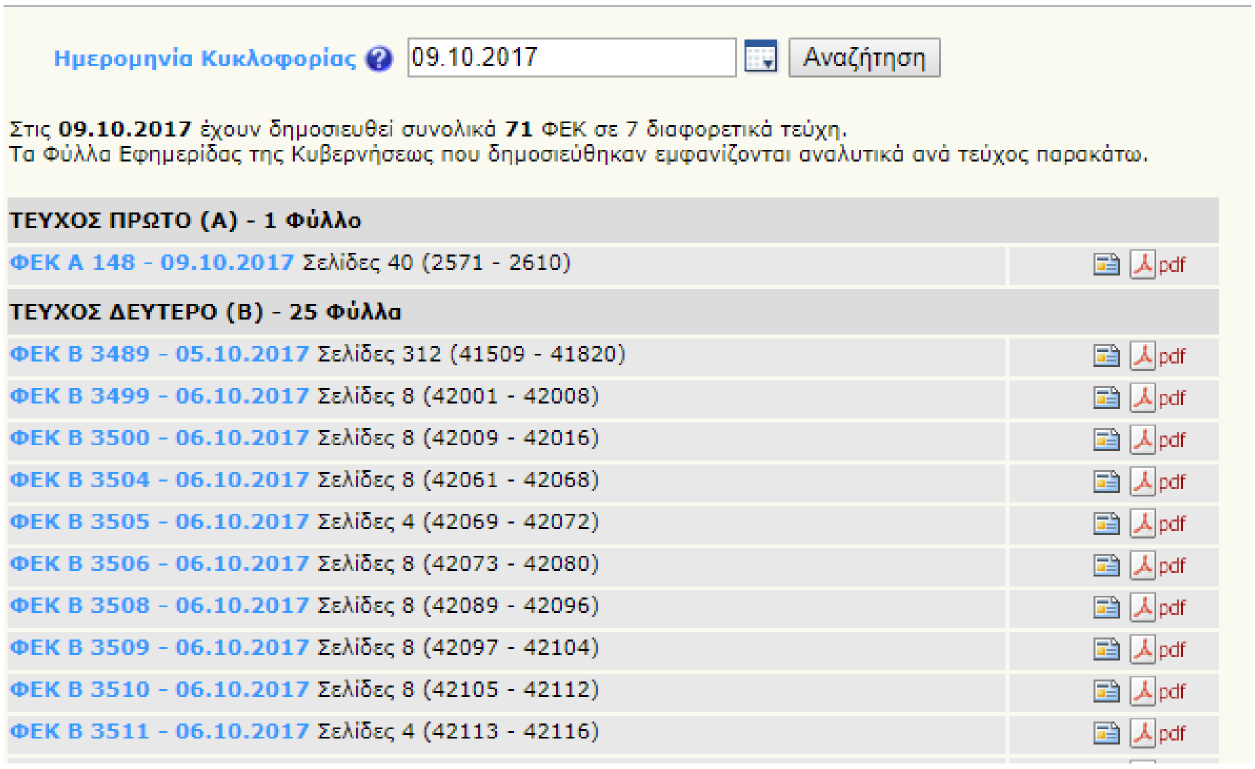
4.1.1 Crawler Technologies

Before starting our developing process, there were a number of considerations regarding the structure of the crawler. Firstly, deciding the programming language was a key factor, considering we would also be in need of a web crawling framework already existent in that language. The main two candidates were Java and Python, because after a brief research we discovered that the more suitable web crawling frameworks for our project were developed in these two particular languages. We picked Python due to its more simplistic command structure and because we thought it would be easier and more "painless" to work with a simple language for a project that we are not familiar with, that being web crawling. Having our programming language selected, the next step called for the selection of a web crawling framework. We considered Beautiful Soup[6], which may not be exactly a framework but a parsing library, which also does a pretty good job of fetching contents from URL and allows you to parse certain parts of them without any hassle. It only fetches the contents of the URL that you give and then stops. It does not crawl unless you manually put it inside an infinite loop with certain criteria. While Beautiful Soup would give us the tools to implement the script that we desire, we continued searching for a framework until we discovered Scrapy[7]. Scrapy [13] [14] is an application framework for crawling web sites and extracting structured data. It is very simple to use and build, test and debug your spiders taking advantage of its command line shell, as well as its great documentation and huge community. With that being said, deciding to work with Scrapy was an easy choice to make. To explain the algorithm that our spider works, we have to explain the way Scrapy spiders function in general.

[6] See: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[7] See: https://scrapy.org/

In the first place, creating a project with Scrapy [14] also generates a default spider that does not perform a specific function until you modify it. The configuration file of the spider is pretty big and anything can be adjusted to fit the requirements of the project. Like every web crawler, Scrapy spiders need an allowed domain, to stop crawling if you reach out of bounds and a "seed" website. The domain in our case, is the webpage of the National Printing House [8]. Going further from there, we needed specify the starting, seeding sub link of our main domain, since the daily releases of new Government Gazette issues appear on a specific link, so to save bandwidth, limit the execution time and get better results we used a direct link to the page that lists the new daily releases as a seed webpage for our spider[9] . After setting the seed website, we proceed with the actual development of our spider.



Στις **09.10.2017** έχουν δημοσιευθεί συνολικά **71** ΦΕΚ σε 7 διαφορετικά τεύχη.
Τα Φύλλα Εφημερίδας της Κυβερνήσεως που δημοσιεύθηκαν εμφανίζονται αναλυτικά ανά τεύχος παρακάτω.

ΤΕΥΧΟΣ ΠΡΩΤΟ (Α) - 1 Φύλλο	
ΦΕΚ Α 148 - 09.10.2017 Σελίδες 40 (2571 - 2610)	pdf
ΤΕΥΧΟΣ ΔΕΥΤΕΡΟ (Β) - 25 Φύλλα	
ΦΕΚ Β 3489 - 05.10.2017 Σελίδες 312 (41509 - 41820)	pdf
ΦΕΚ Β 3499 - 06.10.2017 Σελίδες 8 (42001 - 42008)	pdf
ΦΕΚ Β 3500 - 06.10.2017 Σελίδες 8 (42009 - 42016)	pdf
ΦΕΚ Β 3504 - 06.10.2017 Σελίδες 8 (42061 - 42068)	pdf
ΦΕΚ Β 3505 - 06.10.2017 Σελίδες 4 (42069 - 42072)	pdf
ΦΕΚ Β 3506 - 06.10.2017 Σελίδες 8 (42073 - 42080)	pdf
ΦΕΚ Β 3508 - 06.10.2017 Σελίδες 8 (42089 - 42096)	pdf
ΦΕΚ Β 3509 - 06.10.2017 Σελίδες 8 (42097 - 42104)	pdf
ΦΕΚ Β 3510 - 06.10.2017 Σελίδες 8 (42105 - 42112)	pdf
ΦΕΚ Β 3511 - 06.10.2017 Σελίδες 4 (42113 - 42116)	pdf

Figure 4. 1: Daily Gazette release (Seed) webpage

As shown in Figure 4.1, the seed webpage has a simple layout and is composed by the following parts:

a) A date input box
b) A submit button to retrieve the Gazette documents published on the requested date
c) A text area that mentions the total number of documents released on the specific date
d) A table that contains the list of documents, sorted by issue and ID number, a preview button, as well as a download link.

[8] See: http://www.et.gr

[9] See: http://www.et.gr/idocs-nph/search/dailyFekForm.html

Our spider takes advantage of these 4 parts and extracts the desired files and information.

The Greek Government Gazette web crawler consists of four different functions. We will dissect them and explain the way they work with each other. We present a table with a brief overview of the function’s purpose, before we analyze them.

Function Name	Function Overview
parse	Parses the date (or dates) the crawler will run for
dailyparse	Get the list of links and filenames to be downloaded
downloadfek	Downloads the documents
closed	Closes all the open files that were being used by the spider

Table 4. 1: Crawler Functions Overview

4.1.1.1 parse() function:

When generating a Scrapy spider, the parse() function is automatically created and filled with a Python “pass” statement. The pass statement is a null operation; nothing happens when it executes. By default, this is the function that gets called first for execution by Scrapy when running the web crawler. Our custom parse function creates a directory (if it does not already exist) for our soon to be Gazette documents to be saved into. In addition, it also generates a custom Log file that gets updated with information throughout the crawling process. The information residing into the log file includes execution time, number of files downloaded, number of files already existing, as well as the date and duration of the crawling process. Moving forward, the crawler can be used in three ways, depending on the arguments given at the time of the execution. It can either execute for the current date, a specific date given, or a range of dates. Also, likewise given when executing the spider, the user can select whether he would like to retrieve all the Gazette files released, or just the Α/Β/Γ/Δ issues. By default, when the necessary argument is not specified, the spider downloads only the Α/Β/Γ/Δ issues. The parse method also analyzes the date (or dates) that we want to crawl. If the user does not input a date, or inputs one date, parse calls a POST method, inputting at the date box of the seed website the specified date. The same procedure occurs when the user requests a range of dates, but the POST method is called in loop, for every date in between the starting and ending date given. Then, the response page returned by the POST request is passed to the dailyparse method to continue the series of steps towards the retrieval of the requested files. Note that in case of multiple dates, Scrapy does not crawl the pages in order. It is an asynchronous framework. It uses non-blocking IO, so it doesn't wait for a request to finish before starting the next one.

4.1.1.2 dailyparse() function:

The dailyparse function’s main objective is, after getting a response page from the previous parse() function, to gather the Gazette document names, as well as the download links for each one of them. This is where the actual crawling procedure begins; parse just used a POST method to fetch us the correct webpage to crawl from.

The crawling strategy is no different than the method used by all crawlers. Using CSS and xPath[10] selectors, and inspecting the HTML elements of the webpage, we are examining to find a pattern and retrieve the correct URLs, that link us to the download pages of the documents. As shown in Figure 4.1 and mentioned above, the released Gazette documents are sorted by issue. When the user requests all the documents, regardless of their issue, our job is limited and we just fetch all the list of results returned by the web server. If the user requests only the A/B/Γ/Δ types, we have to make sure that we only retrieve the links of these files.

After we find our way around the webpage, using the CSS selectors we can extract the links from the HREF elements, as well as the name of each document and save them into a Scrapy Item[11], which are basically simple containers used to collect scraped data. They provide a dictionary-like API with a convenient syntax for declaring their available fields. Also, they can easily be passed into another Scrapy method. With that in mind, our Scrapy item consists of two fields: Name and URL. After our items are complete, filled with the documents we want to retrieve, we call the next function, downloadfek, passing the Scrapy Items. Shown below (Figure 4.2) is a visual representation of the combination of parse and dailyparse functions. The item is then passed to downloadfek to take care of the download process as stated above.

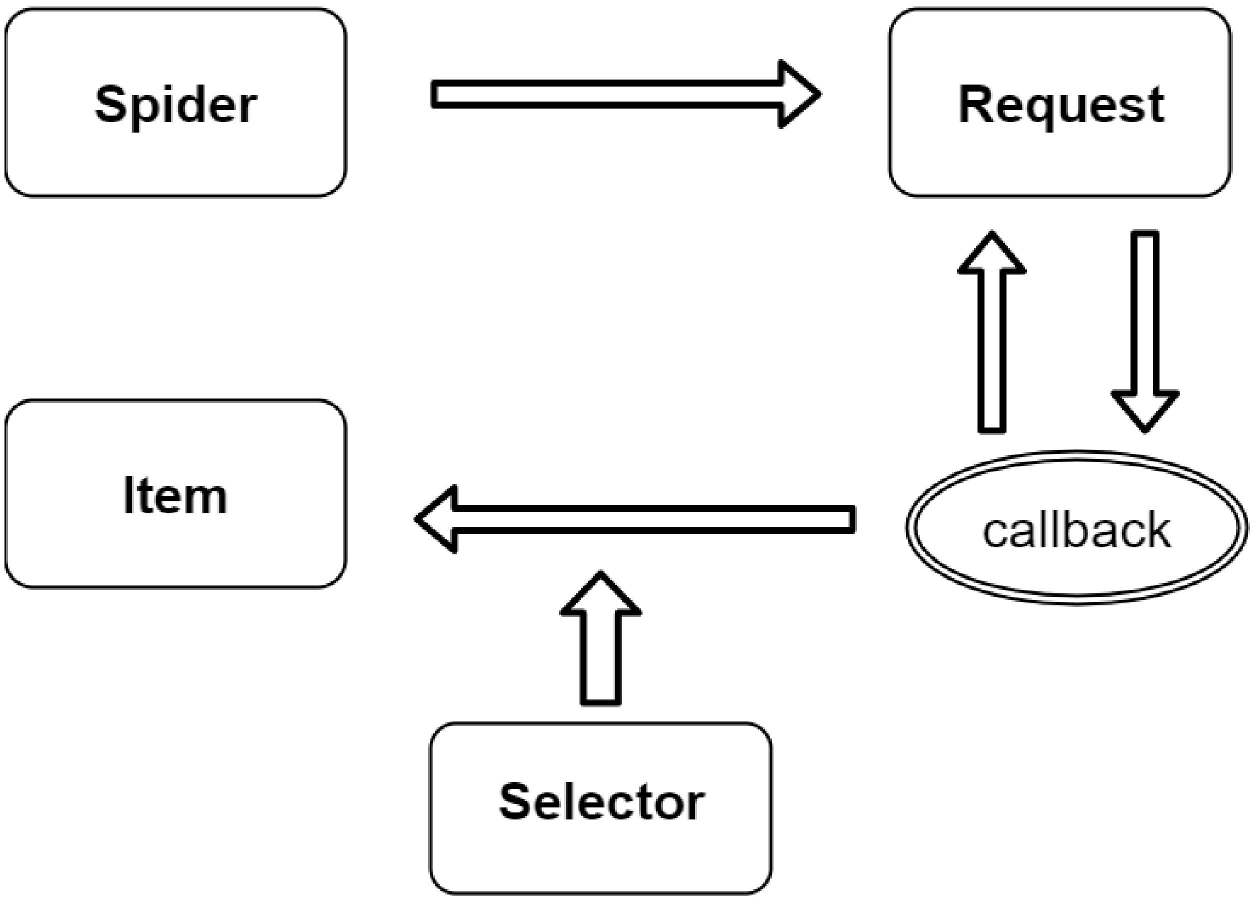


Figure 4. 2: parse and dailyparse functions

[10] See: https://www.w3schools.com/xml/xpath_syntax.asp

[11] See: https://docs.scrapy.org/en/latest/topics/items.html

4.1.1.3 downloadfek() function:

Being fed the hyperlinks of the target documents, the downloadfek function implements the actual downloading of the files. Utilizing Python's native urllib module[12] , which helps the process of fetching data across the World Wide Web, we are able to visit the links passed by dailyparse and retrieve the files we requested. Also, to organize and store the files according to their properties, downloadfek parses the name of the Gazette document, creating a subfolder for the issue of the document (if not existent) and inside that, another directory for the year of the document's release. This way the library is organized by issue/year/filename. Downloadfek also handles exceptions, like timeouts and unreachable host, requesting the files as many times as needed, until they are finally retrieved and stored into our library.

There is a particular and special occasion, that some Government Gazette documents are re-released with modifications and changes. In this case, the National Printing House's website replaces the old, outdated document with the new one, so there is no need to delete old files, since the site itself takes care of our documents to always be the newly released ones.

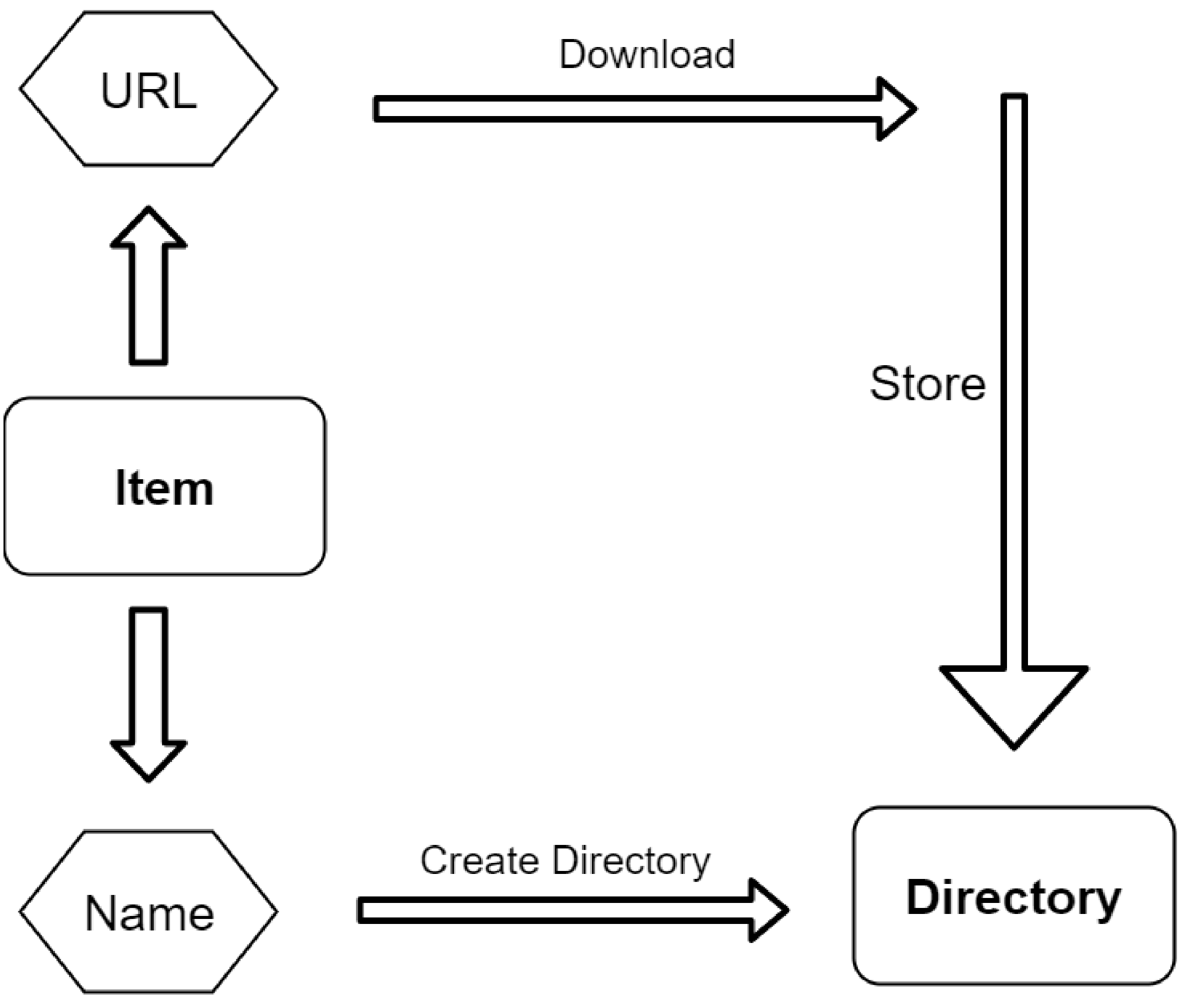


Figure 4. 3: downloadfek visual representation

[12] See: https://docs.python.org/2/library/urllib.html

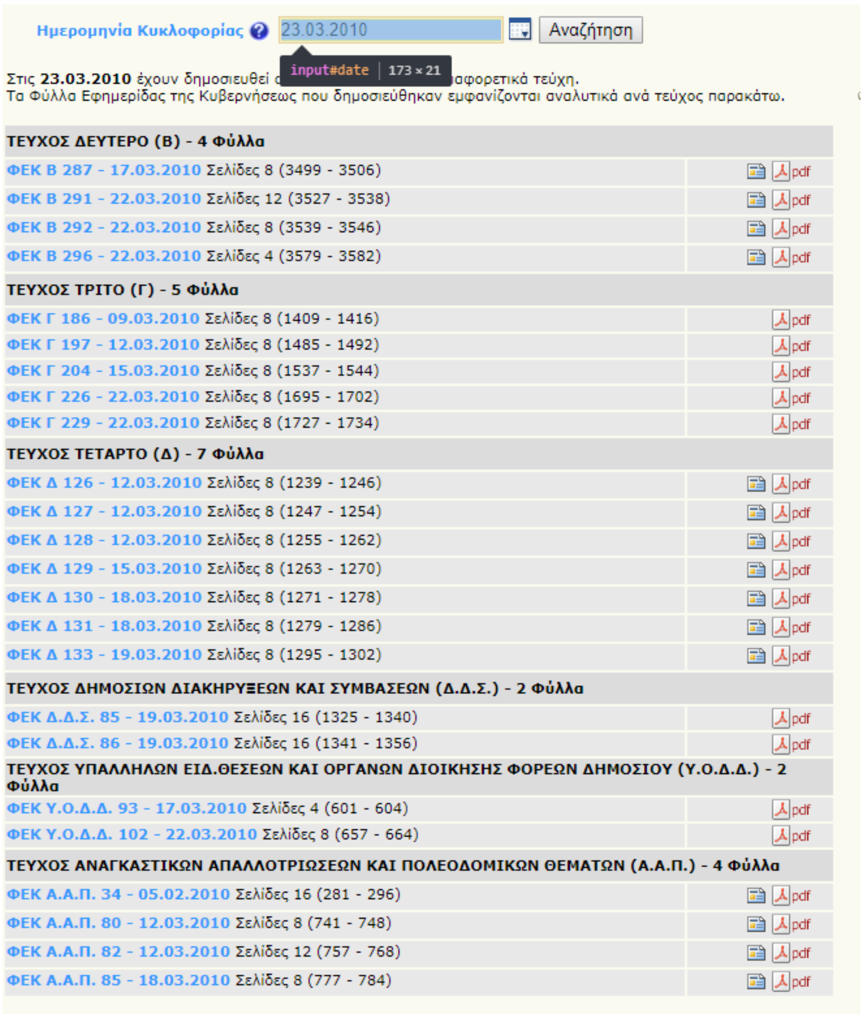
4.1.1.4 closed() function:

When the downloading finishes and the crawler must terminate its session, by default, even when not implemented, Scrapy calls the closed function. This function terminates all the ongoing Scrapy processes and when modified, can execute some final commands to wrap up the crawling process. In our case, closed updates the log file with the time that the execution finished, as well as the numbers of files downloaded or already existed. When closed functions execution ends, the crawler exits as well.

4.1.2 Crawler Demonstration

After analyzing the functionality of the web crawler, we would like to present an example of its usage. We chose to execute the crawler for a specific date and for all the filetypes. Executing for a range of dates would be just a repetition of the same process.

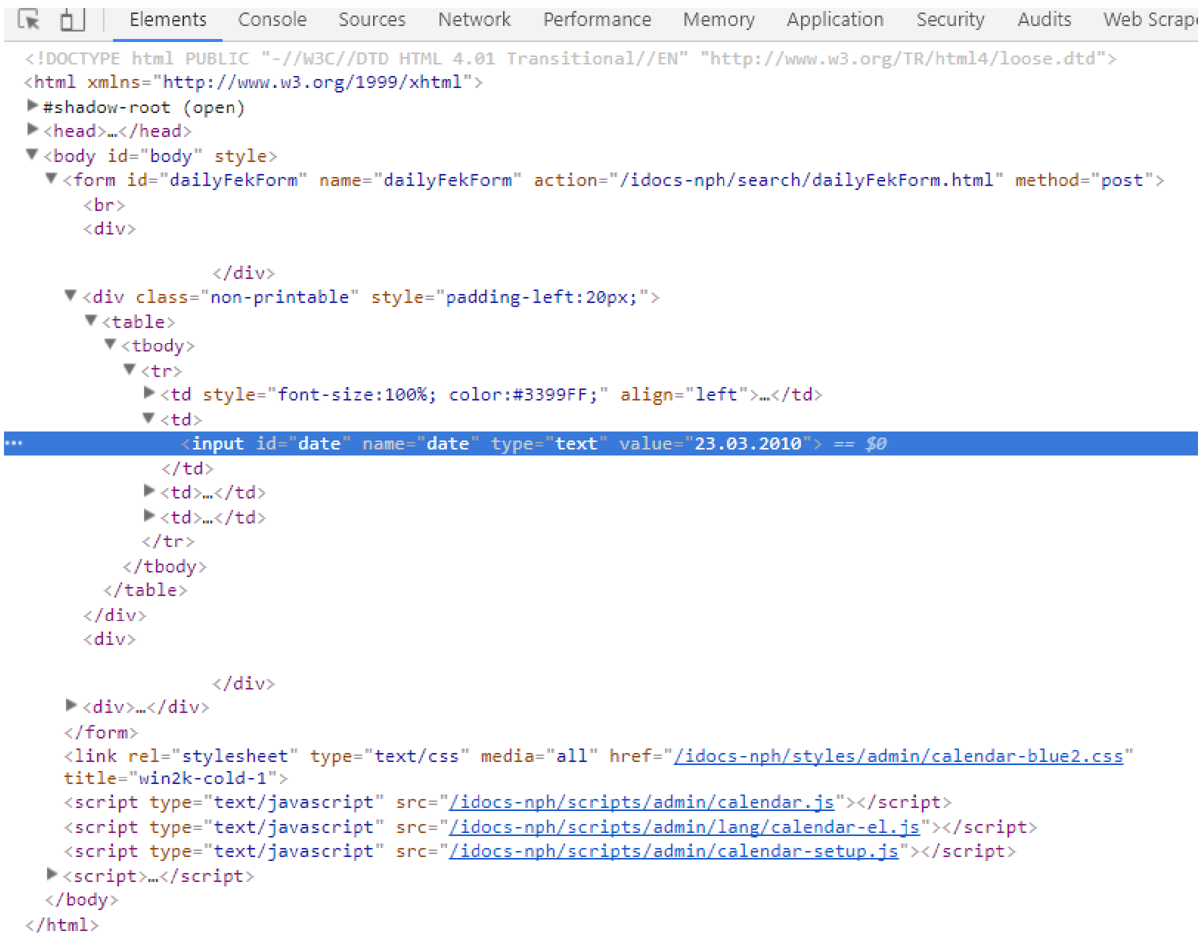
Calling the spider with the arguments “date1=23.03.2010” and “FEKtype=default” to download all the issues released, the parse function uses a POST method input data into the date submission HTML element, to redirect us from the starting, seeding webpage that shows the current date’s releases to the date we requested, in the present example, 23.03.2010. (Figure 4.4 and Figure 4.5)



Στις **23.03.2010** έχουν δημοσιευθεί ... αφορετικά τεύχη.
Τα Φύλλα Εφημερίδας της Κυβερνήσεως που δημοσιεύθηκαν εμφανίζονται αναλυτικά ανά τεύχος παρακάτω.

ΤΕΥΧΟΣ ΔΕΥΤΕΡΟ (Β) - 4 Φύλλα	
ΦΕΚ Β 287 - 17.03.2010 Σελίδες 8 (3499 - 3506)	pdf
ΦΕΚ Β 291 - 22.03.2010 Σελίδες 12 (3527 - 3538)	pdf
ΦΕΚ Β 292 - 22.03.2010 Σελίδες 8 (3539 - 3546)	pdf
ΦΕΚ Β 296 - 22.03.2010 Σελίδες 4 (3579 - 3582)	pdf
ΤΕΥΧΟΣ ΤΡΙΤΟ (Γ) - 5 Φύλλα	
ΦΕΚ Γ 186 - 09.03.2010 Σελίδες 8 (1409 - 1416)	pdf
ΦΕΚ Γ 197 - 12.03.2010 Σελίδες 8 (1485 - 1492)	pdf
ΦΕΚ Γ 204 - 15.03.2010 Σελίδες 8 (1537 - 1544)	pdf
ΦΕΚ Γ 226 - 22.03.2010 Σελίδες 8 (1695 - 1702)	pdf
ΦΕΚ Γ 229 - 22.03.2010 Σελίδες 8 (1727 - 1734)	pdf
ΤΕΥΧΟΣ ΤΕΤΑΡΤΟ (Δ) - 7 Φύλλα	
ΦΕΚ Δ 126 - 12.03.2010 Σελίδες 8 (1239 - 1246)	pdf
ΦΕΚ Δ 127 - 12.03.2010 Σελίδες 8 (1247 - 1254)	pdf
ΦΕΚ Δ 128 - 12.03.2010 Σελίδες 8 (1255 - 1262)	pdf
ΦΕΚ Δ 129 - 15.03.2010 Σελίδες 8 (1263 - 1270)	pdf
ΦΕΚ Δ 130 - 18.03.2010 Σελίδες 8 (1271 - 1278)	pdf
ΦΕΚ Δ 131 - 18.03.2010 Σελίδες 8 (1279 - 1286)	pdf
ΦΕΚ Δ 133 - 19.03.2010 Σελίδες 8 (1295 - 1302)	pdf
ΤΕΥΧΟΣ ΔΗΜΟΣΙΩΝ ΔΙΑΚΗΡΥΞΕΩΝ ΚΑΙ ΣΥΜΒΑΣΕΩΝ (Δ.Δ.Σ.) - 2 Φύλλα	
ΦΕΚ Δ.Δ.Σ. 85 - 19.03.2010 Σελίδες 16 (1325 - 1340)	pdf
ΦΕΚ Δ.Δ.Σ. 86 - 19.03.2010 Σελίδες 16 (1341 - 1356)	pdf
ΤΕΥΧΟΣ ΥΠΑΛΛΗΛΩΝ ΕΙΔ.ΘΕΣΕΩΝ ΚΑΙ ΟΡΓΑΝΩΝ ΔΙΟΙΚΗΣΗΣ ΦΟΡΕΩΝ ΔΗΜΟΣΙΟΥ (Υ.Ο.Δ.Δ.) - 2 Φύλλα	
ΦΕΚ Υ.Ο.Δ.Δ. 93 - 17.03.2010 Σελίδες 4 (601 - 604)	pdf
ΦΕΚ Υ.Ο.Δ.Δ. 102 - 22.03.2010 Σελίδες 8 (657 - 664)	pdf
ΤΕΥΧΟΣ ΑΝΑΓΚΑΣΤΙΚΩΝ ΑΠΑΛΛΟΤΡΙΩΣΕΩΝ ΚΑΙ ΠΟΛΕΟΔΟΜΙΚΩΝ ΘΕΜΑΤΩΝ (Α.Α.Π.) - 4 Φύλλα	
ΦΕΚ Α.Α.Π. 34 - 05.02.2010 Σελίδες 16 (281 - 296)	pdf
ΦΕΚ Α.Α.Π. 80 - 12.03.2010 Σελίδες 8 (741 - 748)	pdf
ΦΕΚ Α.Α.Π. 82 - 12.03.2010 Σελίδες 12 (757 - 768)	pdf
ΦΕΚ Α.Α.Π. 85 - 18.03.2010 Σελίδες 8 (777 - 784)	pdf

Figure 4. 4: parse Function's Response



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 ▶#shadow-root (open)
 ▶<head>…</head>
 ▼<body id="body" style>
   ▼<form id="dailyFekForm" name="dailyFekForm" action="/idocs-nph/search/dailyFekForm.html" method="post">
      <br>
      <div>

                  </div>
     ▼<div class="non-printable" style="padding-left:20px;">
       ▼<table>
         ▼<tbody>
           ▼<tr>
             ▶<td style="font-size:100%; color:#3399FF;" align="left">…</td>
             ▼<td>
                <input id="date" name="date" type="text" value="23.03.2010"> == $0
              </td>
             ▶<td>…</td>
             ▶<td>…</td>
            </tr>
          </tbody>
        </table>
      </div>
      <div>

                  </div>
     ▶<div>…</div>
    </form>
    <link rel="stylesheet" type="text/css" media="all" href="/idocs-nph/styles/admin/calendar-blue2.css"
    title="win2k-cold-1">
    <script type="text/javascript" src="/idocs-nph/scripts/admin/calendar.js"></script>
    <script type="text/javascript" src="/idocs-nph/scripts/admin/lang/calendar-el.js"></script>
    <script type="text/javascript" src="/idocs-nph/scripts/admin/calendar-setup.js"></script>
   ▶<script>…</script>
  </body>
</html>
```

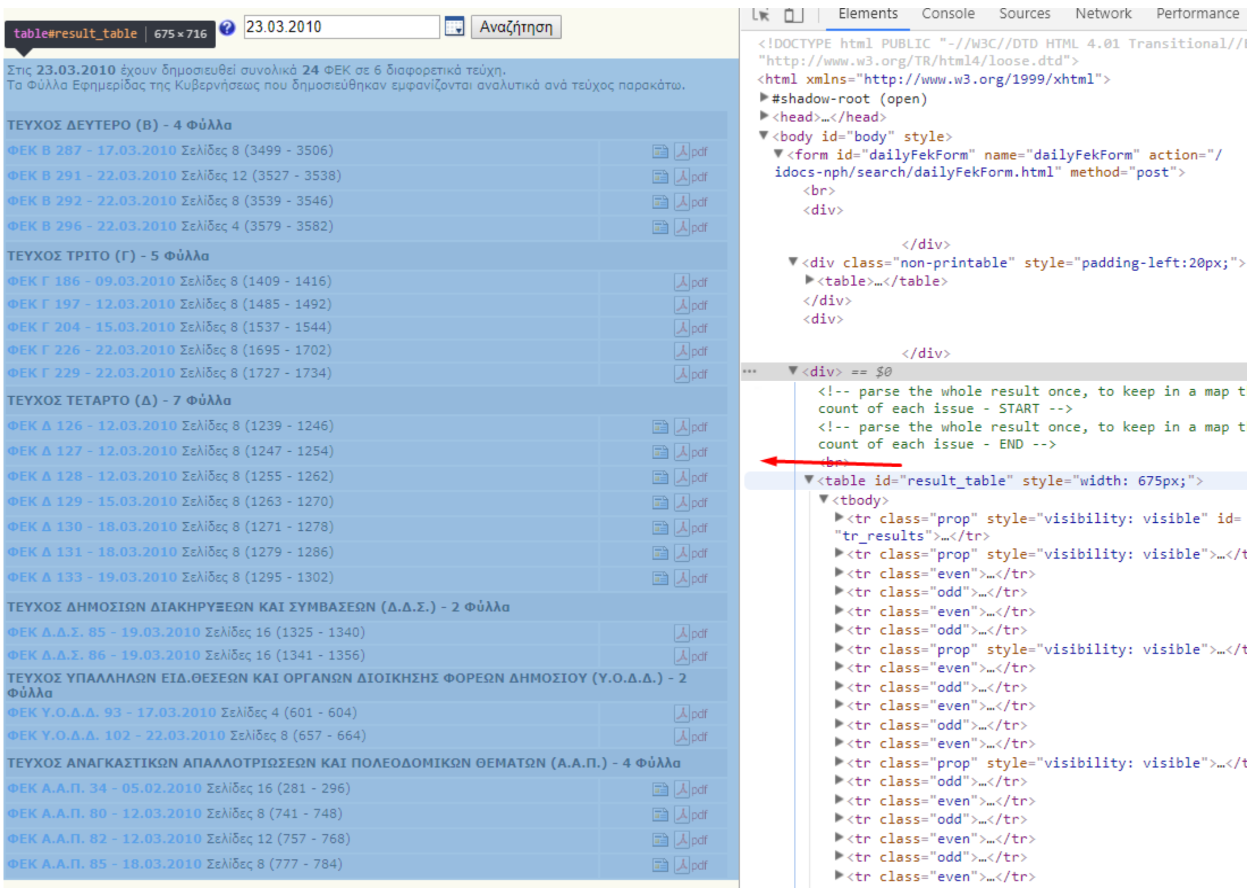
Figure 4. 5: parse function's response elements

The next step is to gather the links and filenames of the documents requested. As we can see in Figure 4.6 below, the results are grouped in an HTML div element with the id of “result_table”. Our crawler, using CSS selectors will isolate the table, and look into it for the elements that will guide us to the data we need.

In the result_table div element, the rows of the table are divided into 3 classes:

a) Class “prop”, which holds the issue name (Figure 4.7)
b) Class “even” and class “odd” which hold the data we need to extract. (Figure 4.8)

Since the crawler was executed with the “default” type issues to be downloaded, the function will check the name of the issue before moving on with the extraction of the download link and name of the document. In our scenario, we are looking only for A/B/Γ/Δ issues. If the issue name is valid for our case, we proceed with the extraction of the data. In our case, only the first 16 documents should get downloaded, since they are of the issues B/Γ/Δ. The selectors extract the data from the “odd” and “even tr” HTML elements and save them into a Scrapy item to pass them for downloading by the next function called, the downloadfek function.



Στις **23.03.2010** έχουν δημοσιευθεί συνολικά **24** ΦΕΚ σε 6 διαφορετικά τεύχη.
Τα Φύλλα Εφημερίδας της Κυβερνήσεως που δημοσιεύθηκαν εμφανίζονται αναλυτικά ανά τεύχος παρακάτω.

ΤΕΥΧΟΣ ΔΕΥΤΕΡΟ (Β) - 4 Φύλλα
ΦΕΚ Β 287 - 17.03.2010 Σελίδες 8 (3499 - 3506) pdf
ΦΕΚ Β 291 - 22.03.2010 Σελίδες 12 (3527 - 3538) pdf
ΦΕΚ Β 292 - 22.03.2010 Σελίδες 8 (3539 - 3546) pdf
ΦΕΚ Β 296 - 22.03.2010 Σελίδες 4 (3579 - 3582) pdf

ΤΕΥΧΟΣ ΤΡΙΤΟ (Γ) - 5 Φύλλα
ΦΕΚ Γ 186 - 09.03.2010 Σελίδες 8 (1409 - 1416) pdf
ΦΕΚ Γ 197 - 12.03.2010 Σελίδες 8 (1485 - 1492) pdf
ΦΕΚ Γ 204 - 15.03.2010 Σελίδες 8 (1537 - 1544) pdf
ΦΕΚ Γ 226 - 22.03.2010 Σελίδες 8 (1695 - 1702) pdf
ΦΕΚ Γ 229 - 22.03.2010 Σελίδες 8 (1727 - 1734) pdf

ΤΕΥΧΟΣ ΤΕΤΑΡΤΟ (Δ) - 7 Φύλλα
ΦΕΚ Δ 126 - 12.03.2010 Σελίδες 8 (1239 - 1246) pdf
ΦΕΚ Δ 127 - 12.03.2010 Σελίδες 8 (1247 - 1254) pdf
ΦΕΚ Δ 128 - 12.03.2010 Σελίδες 8 (1255 - 1262) pdf
ΦΕΚ Δ 129 - 15.03.2010 Σελίδες 8 (1263 - 1270) pdf
ΦΕΚ Δ 130 - 18.03.2010 Σελίδες 8 (1271 - 1278) pdf
ΦΕΚ Δ 131 - 18.03.2010 Σελίδες 8 (1279 - 1286) pdf
ΦΕΚ Δ 133 - 19.03.2010 Σελίδες 8 (1295 - 1302) pdf

ΤΕΥΧΟΣ ΔΗΜΟΣΙΩΝ ΔΙΑΚΗΡΥΞΕΩΝ ΚΑΙ ΣΥΜΒΑΣΕΩΝ (Δ.Δ.Σ.) - 2 Φύλλα
ΦΕΚ Δ.Δ.Σ. 85 - 19.03.2010 Σελίδες 16 (1325 - 1340) pdf
ΦΕΚ Δ.Δ.Σ. 86 - 19.03.2010 Σελίδες 16 (1341 - 1356) pdf

ΤΕΥΧΟΣ ΥΠΑΛΛΗΛΩΝ ΕΙΔ.ΘΕΣΕΩΝ ΚΑΙ ΟΡΓΑΝΩΝ ΔΙΟΙΚΗΣΗΣ ΦΟΡΕΩΝ ΔΗΜΟΣΙΟΥ (Υ.Ο.Δ.Δ.) - 2 Φύλλα
ΦΕΚ Υ.Ο.Δ.Δ. 93 - 17.03.2010 Σελίδες 4 (601 - 604) pdf
ΦΕΚ Υ.Ο.Δ.Δ. 102 - 22.03.2010 Σελίδες 8 (657 - 664) pdf

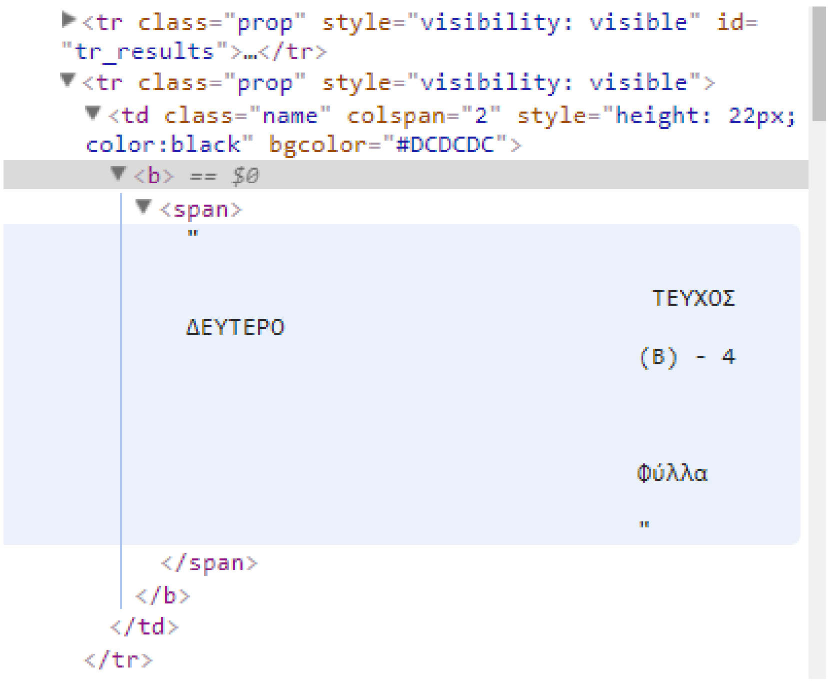
ΤΕΥΧΟΣ ΑΝΑΓΚΑΣΤΙΚΩΝ ΑΠΑΛΛΟΤΡΙΩΣΕΩΝ ΚΑΙ ΠΟΛΕΟΔΟΜΙΚΩΝ ΘΕΜΑΤΩΝ (Α.Α.Π.) - 4 Φύλλα
ΦΕΚ Α.Α.Π. 34 - 05.02.2010 Σελίδες 16 (281 - 296) pdf
ΦΕΚ Α.Α.Π. 80 - 12.03.2010 Σελίδες 8 (741 - 748) pdf
ΦΕΚ Α.Α.Π. 82 - 12.03.2010 Σελίδες 12 (757 - 768) pdf
ΦΕΚ Α.Α.Π. 85 - 18.03.2010 Σελίδες 8 (777 - 784) pdf

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 ▶#shadow-root (open)
 ▶<head>…</head>
 ▼<body id="body" style>
   ▼<form id="dailyFekForm" name="dailyFekForm" action="/
   idocs-nph/search/dailyFekForm.html" method="post">
       <br>
       <div>

                    </div>
     ▼<div class="non-printable" style="padding-left:20px;">
       ▶<table>…</table>
       </div>
       <div>

                    </div>
…    ▼<div> == $0
         <!-- parse the whole result once, to keep in a map t
         count of each issue - START -->
         <!-- parse the whole result once, to keep in a map t
         count of each issue - END -->
         <br>
       ▼<table id="result_table" style="width: 675px;">
         ▼<tbody>
           ▶<tr class="prop" style="visibility: visible" id=
           "tr_results">…</tr>
           ▶<tr class="prop" style="visibility: visible">…</t
           ▶<tr class="even">…</tr>
           ▶<tr class="odd">…</tr>
           ▶<tr class="even">…</tr>
           ▶<tr class="odd">…</tr>
           ▶<tr class="prop" style="visibility: visible">…</t
           ▶<tr class="even">…</tr>
           ▶<tr class="odd">…</tr>
           ▶<tr class="even">…</tr>
           ▶<tr class="odd">…</tr>
           ▶<tr class="even">…</tr>
           ▶<tr class="prop" style="visibility: visible">…</t
           ▶<tr class="odd">…</tr>
           ▶<tr class="even">…</tr>
           ▶<tr class="odd">…</tr>
           ▶<tr class="even">…</tr>
           ▶<tr class="odd">…</tr>
           ▶<tr class="even">…</tr>
```

Figure 4. 6: result table

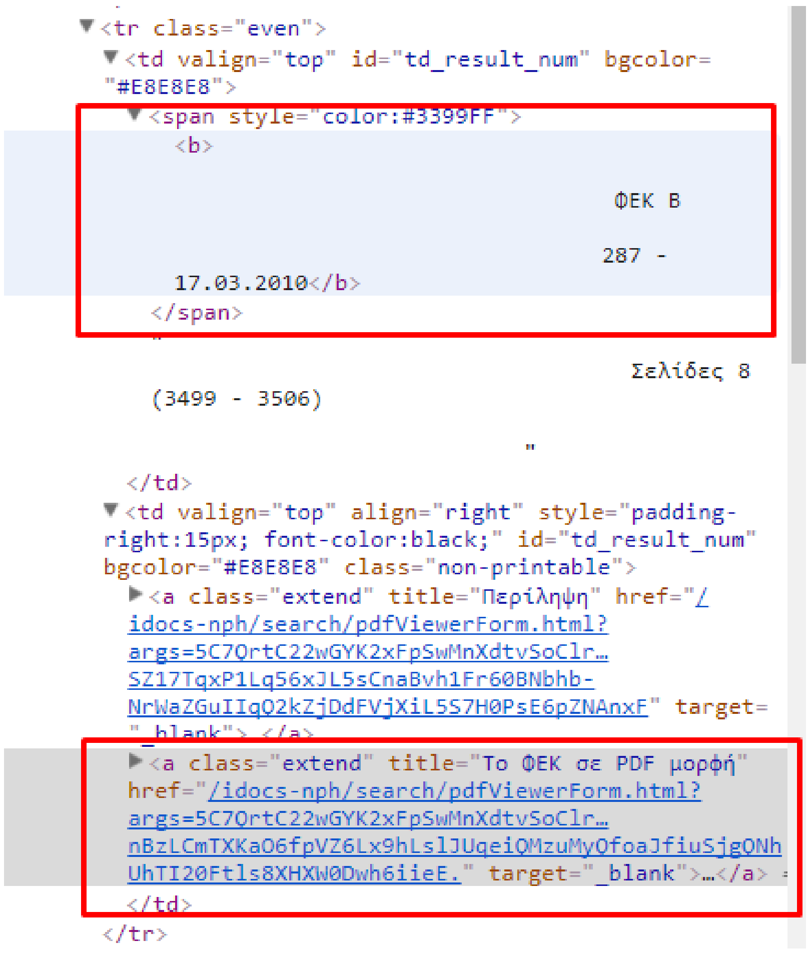


```
▶<tr class="prop" style="visibility: visible" id=
"tr_results">…</tr>
▼<tr class="prop" style="visibility: visible">
  ▼<td class="name" colspan="2" style="height: 22px;
  color:black" bgcolor="#DCDCDC">
    ▼<b> == $0
      ▼<span>
          "

                                                  ΤΕΥΧΟΣ
          ΔΕΥΤΕΡΟ
                                                 (Β) - 4


                                                 Φύλλα
                                                 "
        </span>
      </b>
    </td>
  </tr>
```

Figure 4. 7: prop Class (Issue)



```
▼<tr class="even">
  ▼<td valign="top" id="td_result_num" bgcolor=
  "#E8E8E8">
    ▼<span style="color:#3399FF">
       <b>
                                        ΦΕΚ Β

                                       287 -
       17.03.2010</b>
     </span>
     "
                                                Σελίδες 8
     (3499 - 3506)

                                  "
    </td>
  ▼<td valign="top" align="right" style="padding-
  right:15px; font-color:black;" id="td_result_num"
  bgcolor="#E8E8E8" class="non-printable">
    ▶<a class="extend" title="Περίληψη" href="/
    idocs-nph/search/pdfViewerForm.html?
    args=5C7QrtC22wGYK2xFpSwMnXdtvSoClr…
    SZ17TqxP1Lq56xJL5sCnaBvh1Fr60BNbhb-
    NrWaZGuIIqQ2kZjDdFVjXiL5S7H0PsE6pZNAnxF" target=
    "_blank">…</a>
    ▶<a class="extend" title="Το ΦΕΚ σε PDF μορφή"
    href="/idocs-nph/search/pdfViewerForm.html?
    args=5C7QrtC22wGYK2xFpSwMnXdtvSoClr…
    nBzLCmTXKaO6fpVZ6Lx9hLslJUqeiQMzuMyOfoaJfiuSjgQNh
    UhTI20Ftls8XHXW0Dwh6iieE." target="_blank">…</a>
    </td>
  </tr>
```

Figure 4. 8: even/odd class (data)

The files are being downloaded by the urllib.urlretrieve Python function by visiting the links that we crawled in the previous step. Then, when the script terminates its execution, we simply navigate to the directory created by the parse function to see if our files were correctly downloaded. We can also check the log file created by the web crawler. For this demonstration, the log file generated is the following:

```
Execution started: 20:19:39
Script executed for: 23.03.2010 and the default types.
>> 23.03.2010: 16 FEK files.
Total files: 16
Total files downloaded: 16
Total files already existed: 0
Execution ended: 20:20:26
```

Which means that our script was executed correctly, since the total number of files of issues Α/Β/Γ/Δ were 16, as stated previously.

4.2 Greek Government Gazette Crawler Scheduler

After achieving the retrieval of the Government Gazette documents successfully, the issue left unsolved was the daily execution of the web crawler. While it could be done just with the press of a button, the process would require the physical presence of a system administrator in the main server’s computer, or remotely via SSH. Wanting to simplify the task, we started thinking about scheduling the job to execute automatically every day at a specific time. Since we wanted something universal, to run on all operating systems (OS), we developed the script using Python. We could have conveniently used the native Windows Task Scheduler[13] or cron by Linux[14] depending on the operating system we are working, but creating a program that is executable regardless of the OS was a sounder approach.

Having all the above in mind, we researched and discovered an existing Python module, named “schedule”. An in-process scheduler for periodic jobs that uses the builder pattern for configuration. Schedule lets you run Python functions (or any other callable) periodically at pre-determined intervals using a simple, human-friendly syntax. Being very simple and uncomplicated to use, we simply select the particular command that executes the script we want, in our case, the Greek Government Gazette web crawler, at a specific time every day and our task is complete.

4.3 Greek Government Gazette Library CMS

Having our Gazette library updated and well organized, we are in need of a tool to manage and view our stored files, as well as perform some specific actions like downloading them remotely to another computer, or searching for a specific Gazette document. For the creation of this tool, we considered constructing a simple web CMS which will help us accomplish the actions we previously stated. We also thought of the fact that implementing the web crawler into the CMS would benefit us in the interest of executing and updating the library with past files that might be missing, or even with new files that have not been yet downloaded by our automatic scheduled crawling procedure analyzed in Chapter 3.2.

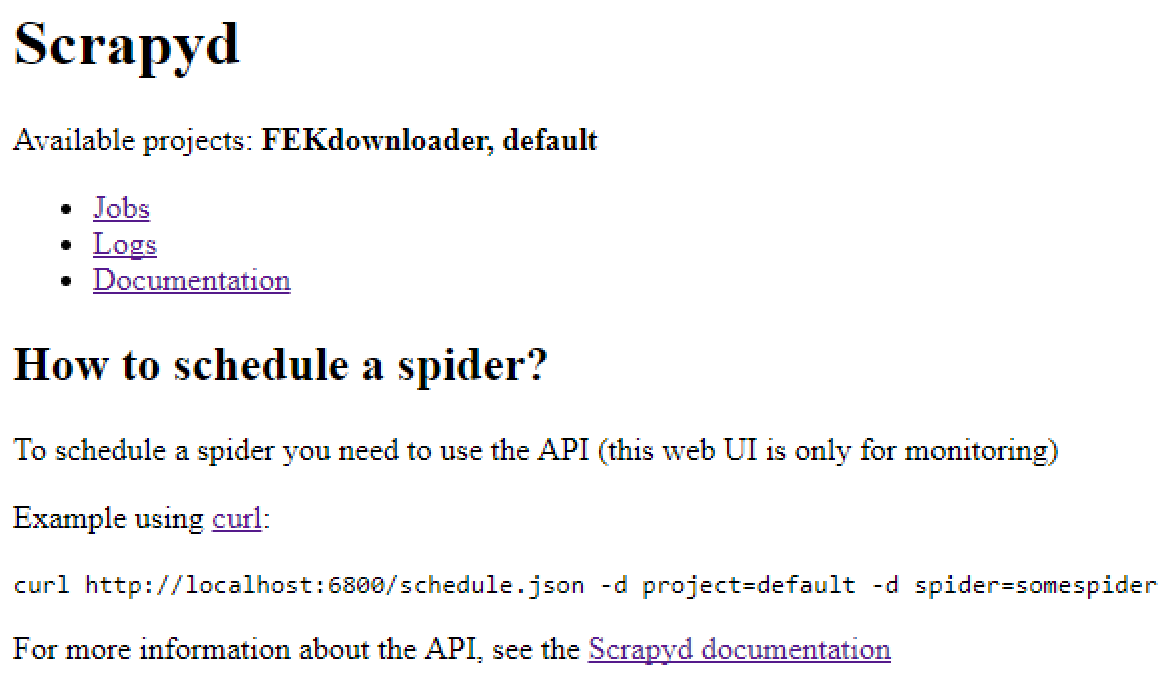
4.3.1 Architecture

The fact that our crawler was programmed using Python led us to select Python again for the development of our website/CMS. Choosing a web development framework was easy, since Django, being the most popular, was our first thought and choice. After creating the HTML files needed and modifying them using CSS and Bootstrap, we went on to generate our Django Project. Django works with the concept of “apps” [16] [17]. A Django App is a Web Application that does a particular job. So, in our case, the apps we created were the “Crawl” app and the “Manage” app.

[13] See: https://technet.microsoft.com/en-us/library/cc721931(v=ws.11).aspx

[14] See: https://www.ibm.com/developerworks/library/l-job-scheduling/index.html

The “Crawler” app is the ensemble of all the required functions and methods which help us use our web crawler through our website. Specifically, we created three different forms, with three different submit buttons, for each distinct use case of our spider (Current date / Specific date / Range of dates). Having dropdown menus for the date selection and the issues of Government Gazette we want to add into our library, we used a POST method to parse the data inputted from these forms and use them as arguments for our spider call. An issue presented itself in this situation. While parsing the forms and retrieving the arguments was a simple operation, the execution process of our spider through Django was complicated, as the Scrapy module could not be called through Django. To overcome this situation, we used a third-party application called Scrapyd. Scrapyd [15] is an application for deploying and running Scrapy spiders. It enables you to deploy (upload) your projects and control their spiders using a JSON API. It listens to requests for spiders to run and spawns a process for each one, executing the spider with the arguments given. Using Scrapyd, we were able to deploy our spider to our local host and start the scrapyd service as a daemon, waiting for requests via its JSON API. Wherever we submit crawl request through our website, a POST method calls a cURL command[15] to add our spider to the execution list of scrapyd, which starts the crawling process. Then, it redirects us to a confirmation page, that presents us with a hyperlink to scrapyd’s minimal web interface, where we can monitor the pending/running/finished processes as well as accessing the default Scrapy log files, not to be confused with the ones we are producing through our crawler. Our files are stored into the library directory that our crawler creates, which is located into the Django project’s root directory.


Scrapyd

Available projects: **FEKdownloader, default**

- Jobs
- Logs
- Documentation

How to schedule a spider?

To schedule a spider you need to use the API (this web UI is only for monitoring)

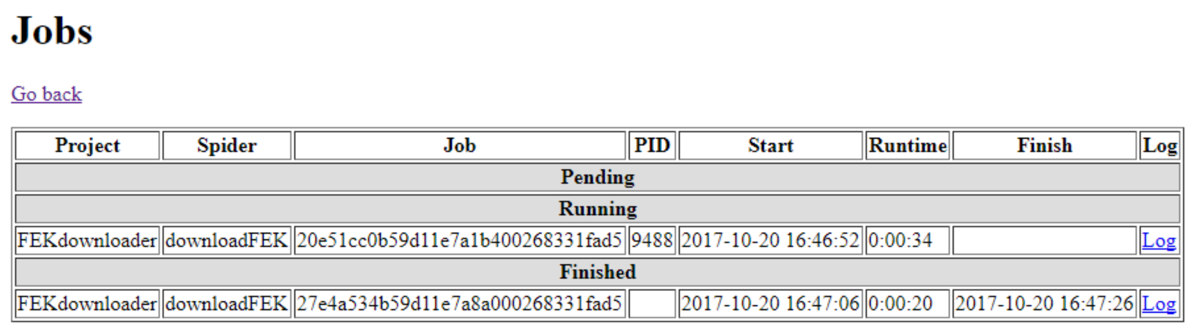
Example using curl:

```
curl http://localhost:6800/schedule.json -d project=default -d spider=somespider
```

For more information about the API, see the Scrapyd documentation

Figure 4. 9: Scrapyd Web Interface

[15] See: https://en.wikipedia.org/wiki/CURL



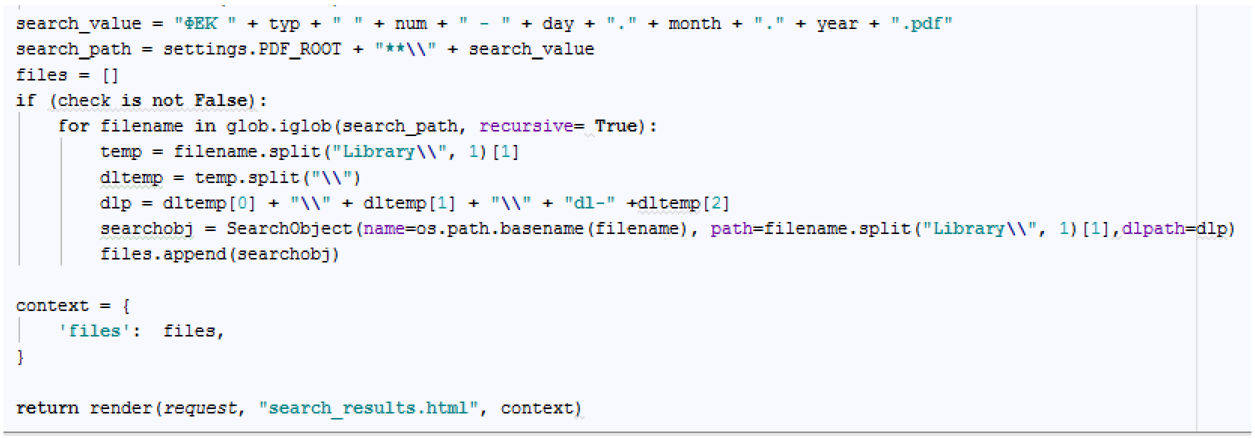
Jobs

Go back

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
FEKdownloader	downloadFEK	20e51cc0b59d11e7a1b400268331fad5	9488	2017-10-20 16:46:52	0:00:34		Log
Finished							
FEKdownloader	downloadFEK	27e4a534b59d11e7a8a000268331fad5		2017-10-20 16:47:06	0:00:20	2017-10-20 16:47:26	Log

Figure 4. 10: Scrapyd Job Monitoring

The "Manager" app includes the methods that assemble the browsing of the library functionality, as well as the search service provided by our project. Our file browser is fairly simple. Using Django's ListView class based functions, we list into our template the directories subsiding into our library's root folder. Organized as they are by the web crawler, the root folder includes the directories of each different issue, as well as the folder which contains the log files we generate during the crawling procedure. Clicking a folder takes us a level deeper, showing us the release year of the issues and then, after selecting a year, listed into our template is the list of Gazette documents that exist into our file system. Since the number of files is immense, we take advantage of Django's pagination feature and show 20 results on each page, to make the browsing session more pleasant and smoother. Clicking the name of a document will request it from our library and present it into the browser, while clicking the download button on the right of the name will retrieve the selected document and save it. Except from the Gazette documents, our template can also render the list of Log files, following the exact same steps. Moving on, we also mentioned of a search function. The user can input a wide variety of different criteria like issue, release day, release month, release year, Gazette ID or a combination of any of those, and using Python's glob module, we can look through our library to scan if a document (or documents) with the requested attributes exist and then list it into a template. Figure 3.11 shows the search functions core. The criteria given by the user are passed via a POST method and parsed by our search function, before assembling the file name to look for and performing the search function.



```
search_value = "ΦΕΚ " + typ + " " + num + " - " + day + "." + month + "." + year + ".pdf"
search_path = settings.PDF_ROOT + "**\\" + search_value
files = []
if (check is not False):
    for filename in glob.iglob(search_path, recursive= True):
        temp = filename.split("Library\\", 1)[1]
        dltemp = temp.split("\\")
        dlp = dltemp[0] + "\\" + dltemp[1] + "\\" + "dl-" +dltemp[2]
        searchobj = SearchObject(name=os.path.basename(filename), path=filename.split("Library\\", 1)[1],dlpath=dlp)
        files.append(searchobj)

context = {
    'files':  files,
}

return render(request, "search_results.html", context)
```

Figure 4. 11: Search function

For all of these operations to work harmoniously together, we need to set our URL patterns. Like all websites, ours too, has a number of different pages which serve different purposes, like the search and crawl pages we mentioned before. The URL patterns are regular expressions that Django uses to redirect to a specific webpage. The URL patterns of our basic pages (homepage, search, crawl) are pretty simple.

Eg: Search page URL: domain/search/

When we move on to the documents, we need some more complex regexes. The paths to the files in our directory are identical to our URL. For example, if we want to retrieve a Gazette document with issue name A and year or release 2010, the directory in our library would be Library/A/2010/. Similarly, when visiting the URL: domain/manager/A/2010/ will show us a list of all the issue A documents released in 2010. We can take this thought another step further.

Eg:If the requested file is ΦΕΚ Α 151 - 12.10.2017.pdf, then the direct link would be: domain/manager/A/2017/ΦΕΚ Α 151 - 12.10.2017.pdf.

So, to sum up, our pattern for directly viewing a Gazette issue, without browsing or searching is: domain/manager /{issue}/{year}/{filename}.pdf

On the other hand, visiting a link that does not exist as a directory in our library, for example, if no issues A were released in 2017 and we tried to access domain/A/2017, it would show us a blank page. Visiting though a page that does not match our regular expression raises a 404 exception. The regular expression used is:

r'^(?i)manager/(?P<typ>(A|B|Α|Β|Γ|Δ))\/(?P<year>[0-9]{4})\/(?P<fnam>[\w\.-
]+?\.pdf)$

This allows the following: Either Α/Β/Γ/Δ types, followed by a 4digit number, followed by a string ending in .pdf. The 4-digit number resembles the year and the string followed by ".pdf", the filename. The regular expressions for the sub-categories, are the same as the one above, subtracting the part that is not required. To be specific:

- Regular expression of issue subdirectory:

r'^(?i)manager/(?P<typ>(A|B|Α|Β|Γ|Δ))/$'

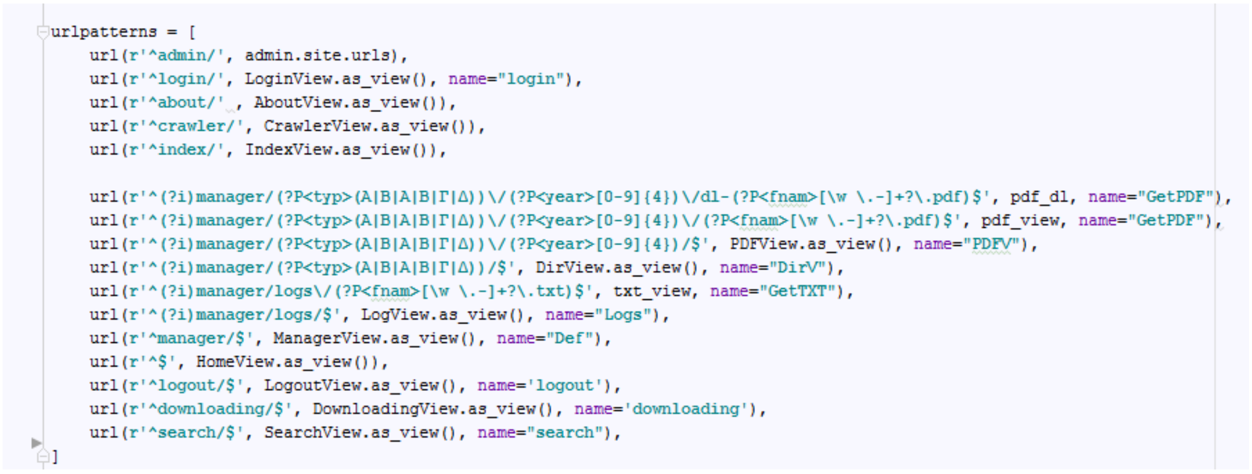
- Regular expression of issue/year subdirectory:

r'^(?i)manager/(?P<typ>(A|B|Α|Β|Γ|Δ))\/(?P<year>[0-9]{4})/$'

Last but not least, we also need a URL pattern to download the files. Simply adding "dl-"before the filename into our URL will save the file, instead of opening it into the browser.

Eg: If we wanted to directly download ΦΕΚ Α 151 - 12.10.2017.pdf, the URL would be: domain/manager /A/2017/dl-ΦΕΚ Α 151 - 12.10.2017.pdf.

A complete look on our URL patterns is shown below, in Figure 3.12.



```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^login/', LoginView.as_view(), name="login"),
    url(r'^about/' , AboutView.as_view()),
    url(r'^crawler/', CrawlerView.as_view()),
    url(r'^index/', IndexView.as_view()),

    url(r'^(?i)manager/(?P<typ>(A|B|A|B|Γ|Δ))\/(?P<year>[0-9]{4})\/dl-(?P<fnam>[\w \.-]+?\.pdf)$', pdf_dl, name="GetPDF"),
    url(r'^(?i)manager/(?P<typ>(A|B|A|B|Γ|Δ))\/(?P<year>[0-9]{4})\/(?P<fnam>[\w \.-]+?\.pdf)$', pdf_view, name="GetPDF"),
    url(r'^(?i)manager/(?P<typ>(A|B|A|B|Γ|Δ))\/(?P<year>[0-9]{4})/$', PDFView.as_view(), name="PDFV"),
    url(r'^(?i)manager/(?P<typ>(A|B|A|B|Γ|Δ))/$', DirView.as_view(), name="DirV"),
    url(r'^(?i)manager/logs\/(?P<fnam>[\w \.-]+?\.txt)$', txt_view, name="GetTXT"),
    url(r'^(?i)manager/logs/$', LogView.as_view(), name="Logs"),
    url(r'^manager/$', ManagerView.as_view(), name="Def"),
    url(r'^$', HomeView.as_view()),
    url(r'^logout/$', LogoutView.as_view(), name='logout'),
    url(r'^downloading/$', DownloadingView.as_view(), name='downloading'),
    url(r'^search/$', SearchView.as_view(), name="search"),
]
```

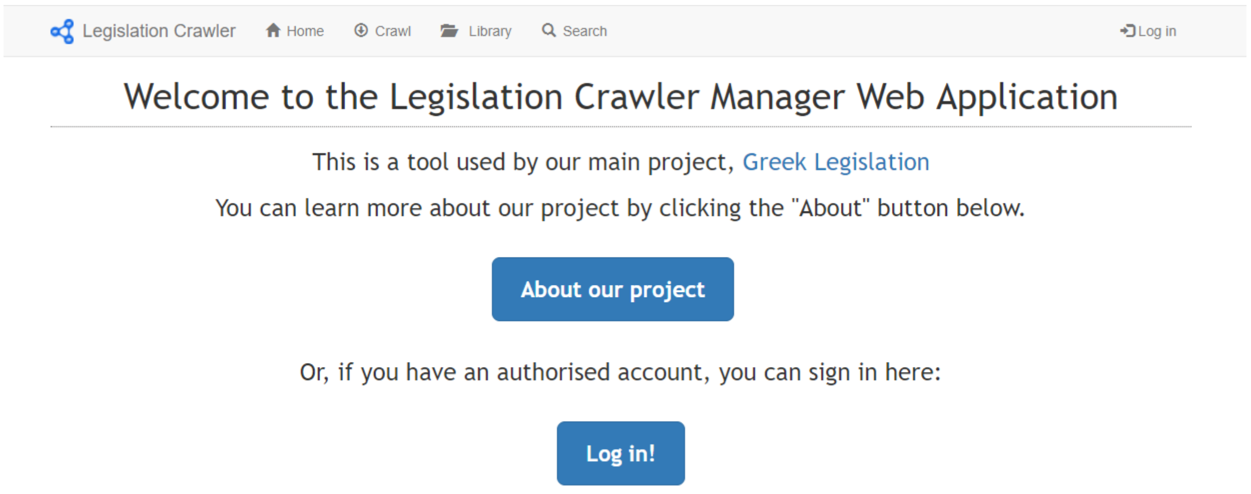
Figure 4. 12: URL patterns

Django Projects can be connected with ease to a database, the most popular being MySQL or MongoDB [17]. Our website does not require the documents to be inserted into a database since it lists the local files of our server. On the contrary, we are using Django’s integrated database, SQLite3 backend, to store the users that will be accessing our website and implementing an account authentication system. The users will be added by the system administrator into the database, therefore there is no registration page needed. Every feature of our website, except the ones that offer information like the homepage or the about page, require the user to be signed on and authorized account to make use of them. To carry out this detail, we are taking advantage of Django’s authentication system, using a «login_required» decorator that redirects to the login page, if a user navigates to a webpage that is not authorized to access without being logged on an account.

4.3.2 Greek Government Gazette Library Demonstration

After starting the developmental server that Django offers, to debug and execute our project in our localhost, without deploying it into an actual server, we navigate to the homepage of our project, which is localhost:8000. This demonstration is consisted by 3 parts: Crawling, Library browser and the search function.

For all these functions to work as we talked about previously, we need to log into an authorized account, by clicking the “Log in” button on the top right of our navigation bar, by following the link localhost:8000/login/ or by clicking the button in the center of our homepage. After doing so, we can access all the features of our website. Our demonstration begins with the usage of our integrated crawler.



Welcome to the Legislation Crawler Manager Web Application

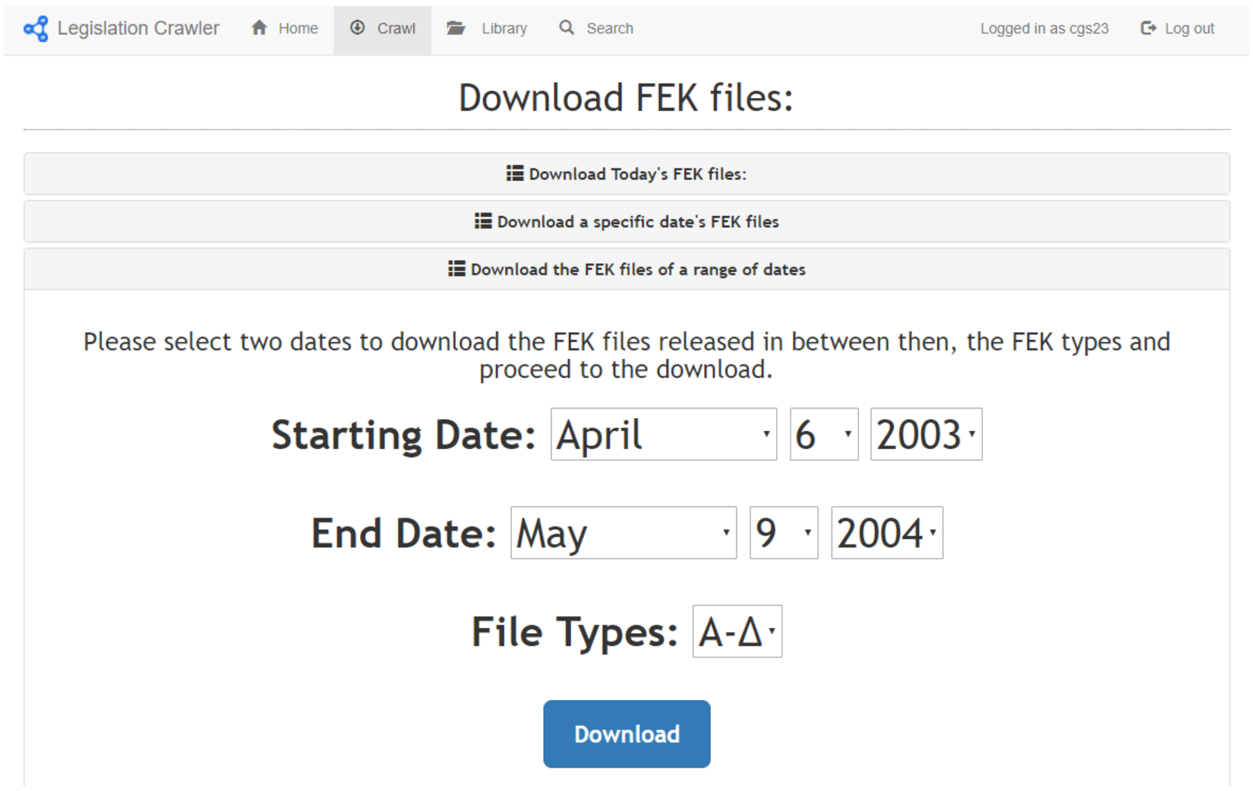
This is a tool used by our main project, Greek Legislation

You can learn more about our project by clicking the "About" button below.

Or, if you have an authorised account, you can sign in here:

Figure 4. 13: Homepage

Clicking the “Crawl” button on the navigation bar redirects us to localhost:8000/crawler, which brings us to the crawling interface. We open one of the three panels which represent the three crawling options. For the sake of this demonstration, we will select the bottom panel, and input the dates 6/4/2003 and 9/5/2004, as well as the Gazette Issues A/B/Γ/Δ to be downloaded. After filling the form, we can submit it using the “Download” button. This will cause cURL command to be called and schedule a crawl for the range of inputted dates via Scrapyd. The downloading of the files has begun and we can see the progress of the retrieval via Scrapyd’s web interface (Figure 4.9)



Download FEK files:

Download Today's FEK files:

Download a specific date's FEK files

Download the FEK files of a range of dates

Please select two dates to download the FEK files released in between then, the FEK types and proceed to the download.

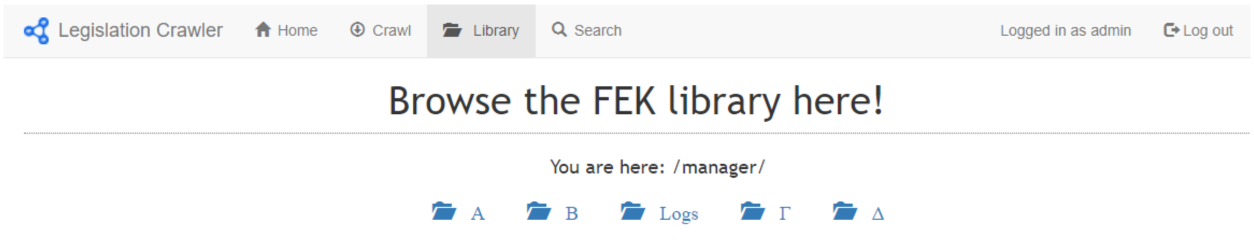
Starting Date: April 6 2003

End Date: May 9 2004

File Types: A-Δ

Figure 4. 14: Crawling page

Navigating to the Library tab takes us to the Gazette document and log file browser. We are using a small sample of our library so the results shown in the demonstrative figures do not include the total files. The Library tab shows us the list of folders that exist into our library's directory. Clicking any of the folders takes us a level deeper. The path that we currently occupy is shown above the folders/files.



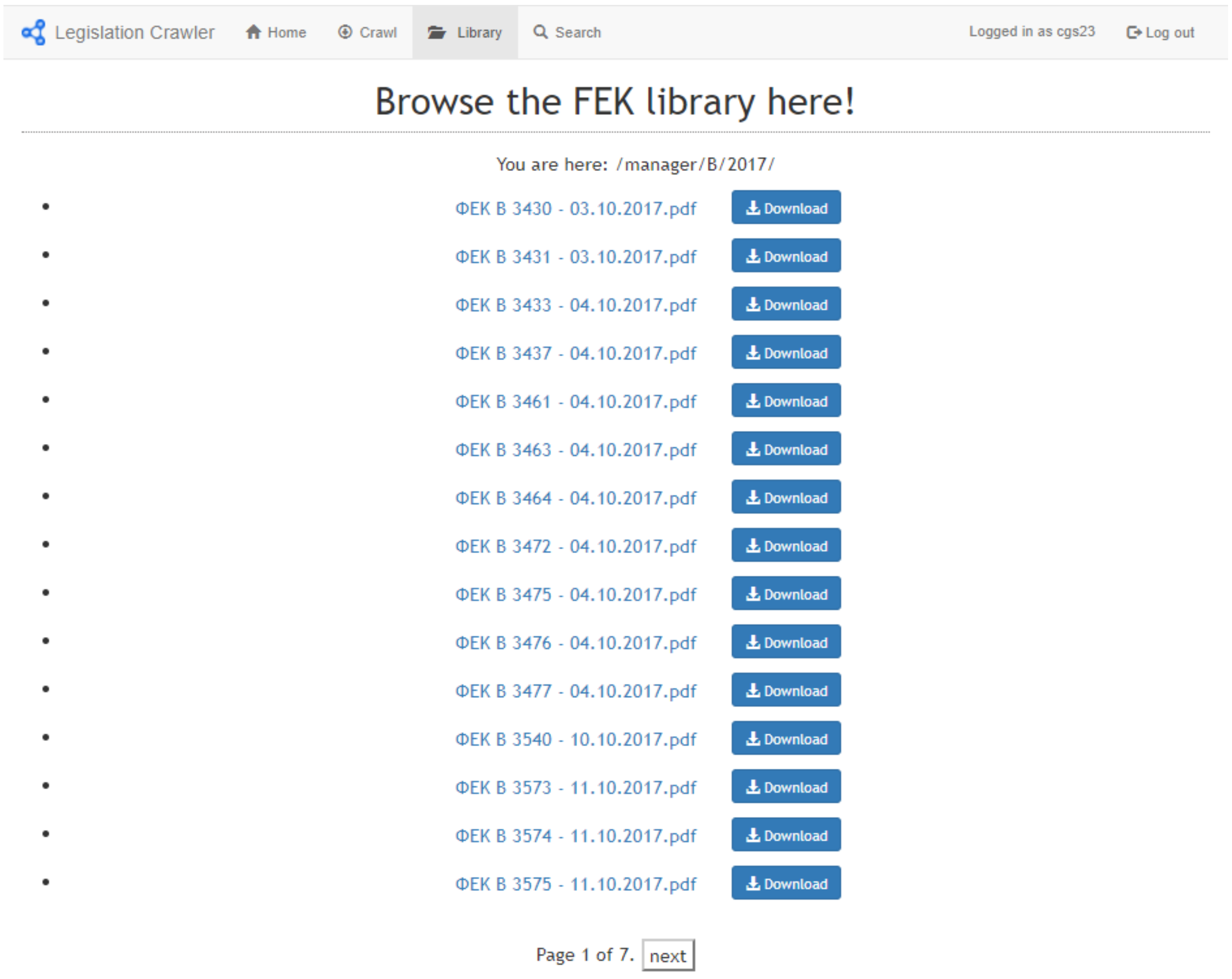
Browse the FEK library here!

You are here: /manager/

A B Logs Γ Δ

Figure 4. 15:Library page (directory view)

After clicking on the Issue name folder, and then the year of release, we are shown the list of documents that subside in our system. We can either click them to view them on our browser or click the download button to retrieve them into our computer.



Browse the FEK library here!

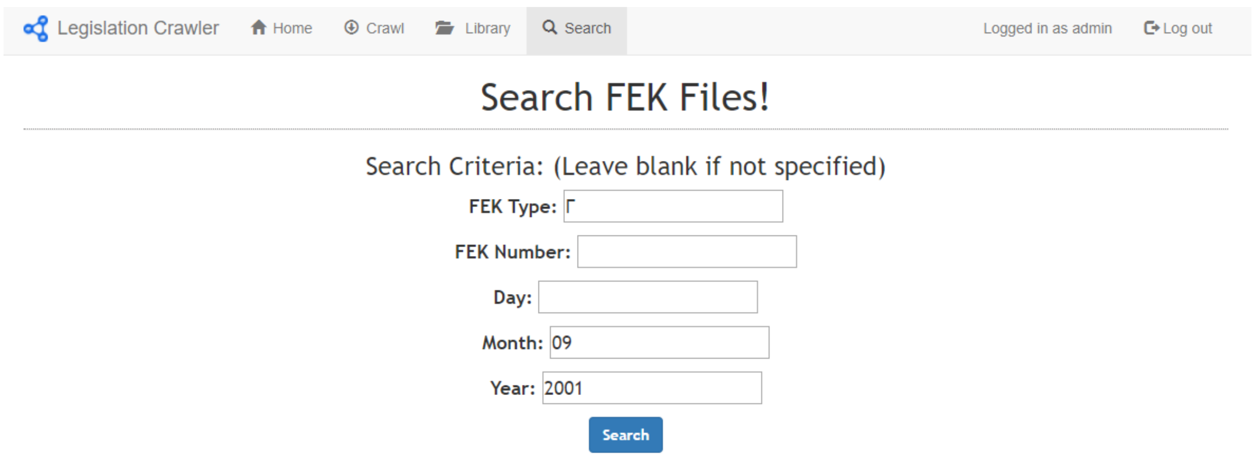
You are here: /manager/B/2017/

- ΦΕΚ Β 3430 - 03.10.2017.pdf Download
- ΦΕΚ Β 3431 - 03.10.2017.pdf Download
- ΦΕΚ Β 3433 - 04.10.2017.pdf Download
- ΦΕΚ Β 3437 - 04.10.2017.pdf Download
- ΦΕΚ Β 3461 - 04.10.2017.pdf Download
- ΦΕΚ Β 3463 - 04.10.2017.pdf Download
- ΦΕΚ Β 3464 - 04.10.2017.pdf Download
- ΦΕΚ Β 3472 - 04.10.2017.pdf Download
- ΦΕΚ Β 3475 - 04.10.2017.pdf Download
- ΦΕΚ Β 3476 - 04.10.2017.pdf Download
- ΦΕΚ Β 3477 - 04.10.2017.pdf Download
- ΦΕΚ Β 3540 - 10.10.2017.pdf Download
- ΦΕΚ Β 3573 - 11.10.2017.pdf Download
- ΦΕΚ Β 3574 - 11.10.2017.pdf Download
- ΦΕΚ Β 3575 - 11.10.2017.pdf Download

Page 1 of 7. next

Figure 4. 16: Library page (document view)

Last but not least, we will demonstrate the search function. Visiting the search page shows us a form so we can fill our criteria and look for the documents we need in our library. We will perform a search to find documents with the following specifications: Issue Γ, release month: September, release year: 2001 as shown in Figure 4.17 bellow.



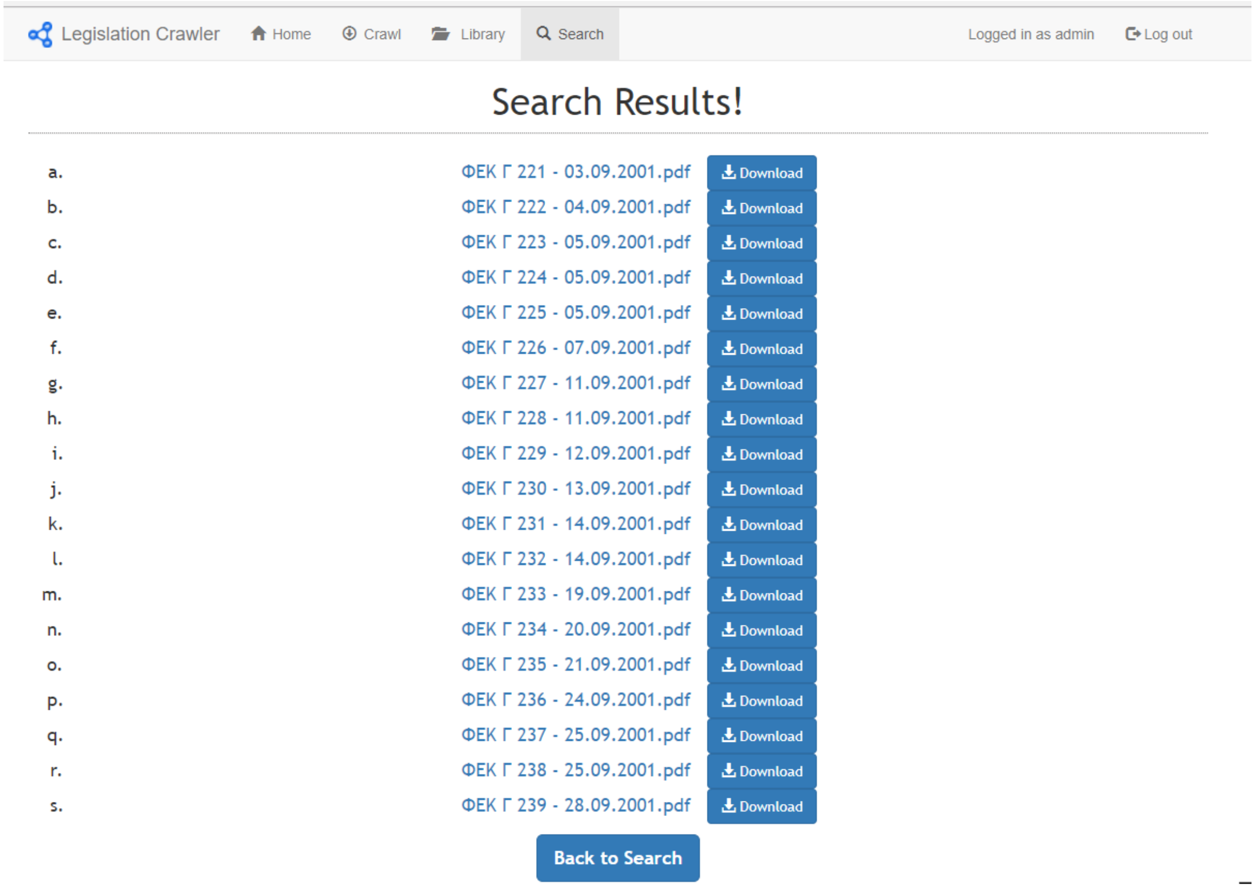
Search FEK Files!

Search Criteria: (Leave blank if not specified)

FEK Type:	Γ
FEK Number:	
Day:	
Month:	09
Year:	2001

Figure 4. 17: Search page

The blank, non-specified criteria are like variables, they can be anything. Searching for the above specifications returns the results showed in Figure 4.18:

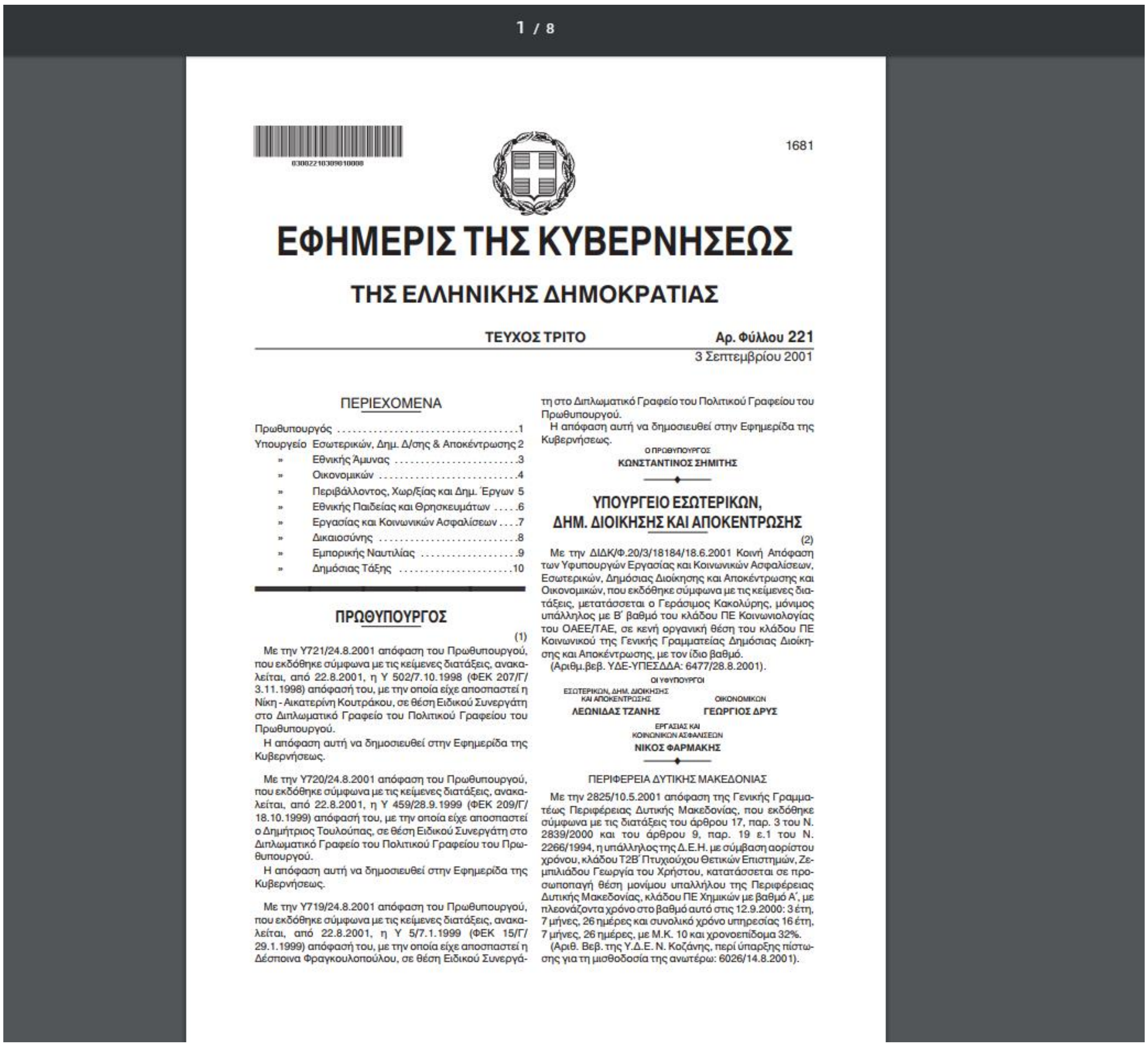


Search Results!

a.	ΦΕΚ Γ 221 - 03.09.2001.pdf	Download
b.	ΦΕΚ Γ 222 - 04.09.2001.pdf	Download
c.	ΦΕΚ Γ 223 - 05.09.2001.pdf	Download
d.	ΦΕΚ Γ 224 - 05.09.2001.pdf	Download
e.	ΦΕΚ Γ 225 - 05.09.2001.pdf	Download
f.	ΦΕΚ Γ 226 - 07.09.2001.pdf	Download
g.	ΦΕΚ Γ 227 - 11.09.2001.pdf	Download
h.	ΦΕΚ Γ 228 - 11.09.2001.pdf	Download
i.	ΦΕΚ Γ 229 - 12.09.2001.pdf	Download
j.	ΦΕΚ Γ 230 - 13.09.2001.pdf	Download
k.	ΦΕΚ Γ 231 - 14.09.2001.pdf	Download
l.	ΦΕΚ Γ 232 - 14.09.2001.pdf	Download
m.	ΦΕΚ Γ 233 - 19.09.2001.pdf	Download
n.	ΦΕΚ Γ 234 - 20.09.2001.pdf	Download
o.	ΦΕΚ Γ 235 - 21.09.2001.pdf	Download
p.	ΦΕΚ Γ 236 - 24.09.2001.pdf	Download
q.	ΦΕΚ Γ 237 - 25.09.2001.pdf	Download
r.	ΦΕΚ Γ 238 - 25.09.2001.pdf	Download
s.	ΦΕΚ Γ 239 - 28.09.2001.pdf	Download

Figure 4. 18: Search results page

To conclude our demonstration, we select to preview the first file returned from our search function, with filename ΦΕΚ Γ 221 – 03.09.2001.pdf. Clicking it redirects us to localhost:8000/manager/Γ/2001/ ΦΕΚ Γ 221 – 03.09.2001.pdf and we can see the file, as presented in Figure 4.19:


1 / 8

1681

ΕΦΗΜΕΡΙΣ ΤΗΣ ΚΥΒΕΡΝΗΣΕΩΣ

ΤΗΣ ΕΛΛΗΝΙΚΗΣ ΔΗΜΟΚΡΑΤΙΑΣ

ΤΕΥΧΟΣ ΤΡΙΤΟ — Αρ. Φύλλου 221

3 Σεπτεμβρίου 2001

ΠΕΡΙΕΧΟΜΕΝΑ

Πρωθυπουργός 1
Υπουργείο Εσωτερικών, Δημ. Δ/σης & Αποκέντρωσης 2
» Εθνικής Άμυνας 3
» Οικονομικών 4
» Περιβάλλοντος, Χωρ/ξίας και Δημ. Έργων 5
» Εθνικής Παιδείας και Θρησκευμάτων 6
» Εργασίας και Κοινωνικών Ασφαλίσεων 7
» Δικαιοσύνης 8
» Εμπορικής Ναυτιλίας 9
» Δημόσιας Τάξης 10

ΠΡΩΘΥΠΟΥΡΓΟΣ

(1)

Με την Υ721/24.8.2001 απόφαση του Πρωθυπουργού, που εκδόθηκε σύμφωνα με τις κείμενες διατάξεις, ανακαλείται, από 22.8.2001, η Υ 502/7.10.1998 (ΦΕΚ 207/Γ/3.11.1998) απόφασή του, με την οποία είχε αποσπαστεί η Νίκη - Αικατερίνη Κουτράκου, σε θέση Ειδικού Συνεργάτη στο Διπλωματικό Γραφείο του Πολιτικού Γραφείου του Πρωθυπουργού.

Η απόφαση αυτή να δημοσιευθεί στην Εφημερίδα της Κυβερνήσεως.

Με την Υ720/24.8.2001 απόφαση του Πρωθυπουργού, που εκδόθηκε σύμφωνα με τις κείμενες διατάξεις, ανακαλείται, από 22.8.2001, η Υ 459/28.9.1999 (ΦΕΚ 209/Γ/18.10.1999) απόφασή του, με την οποία είχε αποσπαστεί ο Δημήτριος Τουλούπας, σε θέση Ειδικού Συνεργάτη στο Διπλωματικό Γραφείο του Πολιτικού Γραφείου του Πρωθυπουργού.

Η απόφαση αυτή να δημοσιευθεί στην Εφημερίδα της Κυβερνήσεως.

Με την Υ719/24.8.2001 απόφαση του Πρωθυπουργού, που εκδόθηκε σύμφωνα με τις κείμενες διατάξεις, ανακαλείται, από 22.8.2001, η Υ 5/7.1.1999 (ΦΕΚ 15/Γ/29.1.1999) απόφασή του, με την οποία είχε αποσπαστεί η Δέσποινα Φραγκουλοπούλου, σε θέση Ειδικού Συνεργάτη στο Διπλωματικό Γραφείο του Πολιτικού Γραφείου του Πρωθυπουργού.

Η απόφαση αυτή να δημοσιευθεί στην Εφημερίδα της Κυβερνήσεως.

Ο ΠΡΩΘΥΠΟΥΡΓΟΣ
ΚΩΝΣΤΑΝΤΙΝΟΣ ΣΗΜΙΤΗΣ

ΥΠΟΥΡΓΕΙΟ ΕΣΩΤΕΡΙΚΩΝ, ΔΗΜ. ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΑΠΟΚΕΝΤΡΩΣΗΣ

(2)

Με την ΔΙΔΚ/Φ.20/3/18184/18.6.2001 Κοινή Απόφαση των Υφυπουργών Εργασίας και Κοινωνικών Ασφαλίσεων, Εσωτερικών, Δημόσιας Διοίκησης και Αποκέντρωσης και Οικονομικών, που εκδόθηκε σύμφωνα με τις κείμενες διατάξεις, μετατάσσεται ο Γεράσιμος Κακολύρης, μόνιμος υπάλληλος με Β΄ βαθμό του κλάδου ΠΕ Κοινωνιολογίας του ΟΑΕΕ/ΤΑΕ, σε κενή οργανική θέση του κλάδου ΠΕ Κοινωνικού της Γενικής Γραμματείας Δημόσιας Διοίκησης και Αποκέντρωσης, με τον ίδιο βαθμό.

(Αριθμ.βεβ. ΥΔΕ-ΥΠΕΣΔΔΑ: 6477/28.8.2001).

ΟΙ ΥΦΥΠΟΥΡΓΟΙ

ΕΣΩΤΕΡΙΚΩΝ, ΔΗΜ. ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΑΠΟΚΕΝΤΡΩΣΗΣ
ΛΕΩΝΙΔΑΣ ΤΖΑΝΗΣ

ΟΙΚΟΝΟΜΙΚΩΝ
ΓΕΩΡΓΙΟΣ ΔΡΥΣ

ΕΡΓΑΣΙΑΣ ΚΑΙ ΚΟΙΝΩΝΙΚΩΝ ΑΣΦΑΛΙΣΕΩΝ
ΝΙΚΟΣ ΦΑΡΜΑΚΗΣ

ΠΕΡΙΦΕΡΕΙΑ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Με την 2825/10.5.2001 απόφαση της Γενικής Γραμματέως Περιφέρειας Δυτικής Μακεδονίας, που εκδόθηκε σύμφωνα με τις διατάξεις του άρθρου 17, παρ. 3 του Ν. 2839/2000 και του άρθρου 9, παρ. 19 ε.1 του Ν. 2266/1994, η υπάλληλος της Δ.Ε.Η. με σύμβαση αορίστου χρόνου, κλάδου Τ2Β΄ Πτυχιούχου Θετικών Επιστημών, Ζεμπιλιάδου Γεωργία του Χρήστου, κατατάσσεται σε προσωποπαγή θέση μονίμου υπαλλήλου της Περιφέρειας Δυτικής Μακεδονίας, κλάδου ΠΕ Χημικών με βαθμό Α΄, με πλεονάζοντα χρόνο στο βαθμό αυτό στις 12.9.2000: 3 έτη, 7 μήνες, 26 ημέρες και συνολικό χρόνο υπηρεσίας 16 έτη, 7 μήνες, 26 ημέρες, με Μ.Κ. 10 και χρονοεπίδομα 32%.

(Αριθ. Βεβ. της Υ.Δ.Ε. Ν. Κοζάνης, περί ύπαρξης πίστωσης για τη μισθοδοσία της ανωτέρω: 6026/14.8.2001).

Figure 4. 19: Browser File Preview

4.4 Conclusion

In Chapter 4 we went into specifics regarding the technologies used for the development of our web crawler, scheduler and CMS. We explained why we picked these frameworks and tools and we demonstrated how our CMS and Crawler function, giving in depth figures and examples of every function that they are capable of executing. This way the reader can follow our thought process and figure out how everything operates, alone as a unit and all together as a whole.

5. GREEK GOVERNMENT GAZETTE ONTOLOGY

As we mentioned in the background section of this thesis, specifically Chapter 3.4, the Semantic representation of the Government Gazette via ELI and Akoma Ntoso is limited. To provide more depth to the illustration of the Gazette, we propose an extended ontology to assist us in displaying some basic attributes and characteristics of Gazette documents.

5.1 Ontology

As we mentioned, the lackluster representation of Gazette information by the big Legislative Semantic Standards creates the need for an updated Greek Government Gazette Ontology. Using classes and properties from ELI, Akoma Ntoso, the Core Public Organization Vocabulary and the Simple Knowledge Organization System data model and combining them helped us generate the Ontology presented in Figure 5.1. We will proceed to explain and analyze the Ontology.

Our aim is to present the Gazette's information. Expanding from Akoma Ntoso, we get an abstract "OfficialGazette" class. This class has subclasses for each of the Gazette issue, Issue A, B, C, etc. According to ELI, there has to be a direct property connecting the Gazette and the Format it gets published in. So, with the existent ELI properties, the only way to express the legal content of a Gazette is through a very complex relationship between entities. That a LegalResource is realized by a LegalExpression, which is embodied in a Format type, in which the document is published into the Gazette. The solution we propose, shown in the Figure below, is a new property, the nomothesia:published_in property. This relation connects the Gazette directly to the LegalResource, showing us what kind of legal information is published into it.

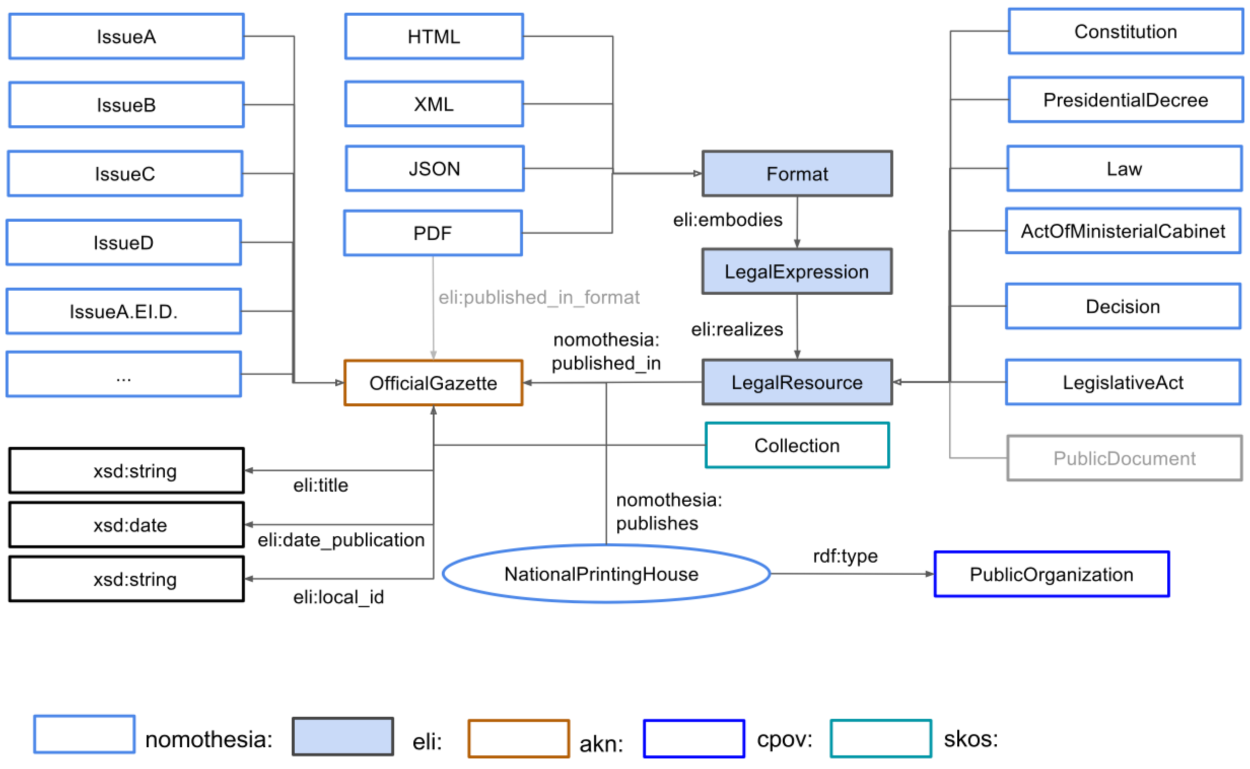


Figure 5. 1:Greek Government Gazette Ontology

Furthermore, via ELI, we can add other information to OfficialGazette via data properties, for instance title, local id and date of publication.

It is also visible that the OfficialGazette entity is a collection of LegalResources., via the Simple Knowledge Organization System data model. Moreover, the OfficialGazette is getting published by the National Printing House, which, according to the Core Public Organization Vocabulary, is a PublicOrganization.

5.2 Greek Government Gazette URI Pattern

Uniform Resource Identifiers (URIs) are strings of characters used to identify a name or resource on the internet, like documents, images, downloadable files. Every Greek Government Gazette document has its own, exclusive, unique URI pattern. URIs need to be highly guessable so we propose a schema of dedicated HTTP URIs. The URI pattern proposed is based on the Gazette ontology presented beforehand. The pattern is:

http://legislation.di.uoa.gr/eli/gazette/{Issue_ID}/{Year}/{Serial_Number}

Any field between curly brackets needs to take specific value. Issue ID should be replaced with the issue of the Government Gazette document we want to retrieve. We are using the encoding presented in Table 5.1 below. Year is actually the year of publication (e.g., 2012) and Serial_Number is the number of the specific Gazette document. So, for example if we want to address the document 54 of 2012, with the Issue ID of A, the corresponding URI is:

http://legislation.di.uoa.gr/eli/gazette/A/2012/54/

Table 5. 1: URI Issue IDs

Issue Name	Issue ID
A, B, C, D	A, B, C, D
Α.Α.Π	AAP
Α.ΕΙ.Δ	AEID
Α.Σ.Ε.Π.	ASEP
ΠΡΑ.Δ.Ι.Τ.	PRADIT
Δ.Δ.Σ	DDS
Ο.Π.Κ.	OPK
Υ.Ο.Δ.Δ.	YODD
ΑΕ - ΕΠΕ / ΓΕΜΗ	AEEPE

Another example using a more complex type of Gazette issue would be the document 100, published in 2010, with the Issue ID of ΠΡΑ.Δ.Ι.Τ. The analogous URI is:

http://legislation.di.uoa.gr/eli/gazette/PRADIT/2010/100/

5.3 Greek Government Gazette Dataset

Using the ontology we explained above, we developed a script using Python that generates a dataset based on the titles of the crawled Gazette documents. It creates RDF triples going through the whole Nomothesi@ library. Given that as of 01 September 2017 the Greek Government Gazette documents published after 01 January 2000 are 84,238, the dataset consists of 421,190 RDF triples, inside a text file. We are only taking into account the Issues A/B/C/D. Shown below in Figure 5.2 is a graphic example of our dataset.

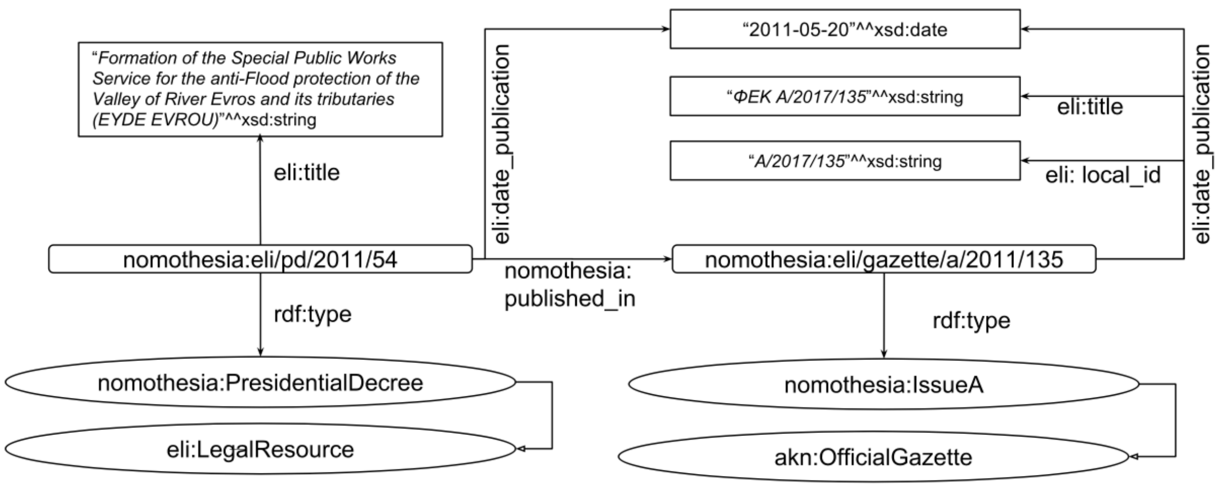

"Formation of the Special Public Works Service for the anti-Flood protection of the Valley of River Evros and its tributaries (EYDE EVROU)"^^xsd:string
eli:title
nomothesia:eli/pd/2011/54
rdf:type
nomothesia:PresidentialDecree
eli:LegalResource
eli:date_publication
nomothesia: published_in
"2011-05-20"^^xsd:date
"ΦΕΚ A/2017/135"^^xsd:string
"A/2017/135"^^xsd:string
eli:title
eli: local_id
eli:date_publication
nomothesia:eli/gazette/a/2011/135
rdf:type
nomothesia:IssueA
akn:OfficialGazette

Figure 5. 2: Dataset RDF Example Graph

5.4 Conclusion

Using ELI, Akoma Ntoso and other data sets and vocabularies we are able to create an ontology to display the Government Gazette information with RDF triplets. We proposed a new property that helps us achieve simple expression regarding the legal content of a Gazette and explained how all the new and old properties and classes coexist harmoniously into our new ontology. We also introduced a script that produces a dataset based on the titles of our Gazette documents.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this work, we discussed the importance of job scheduling and content management as well as the benefits that occur from web crawling. Updating Nomothesi@'s file library daily and automatically with the new Government Gazette publishes via our Spider and Job Scheduler assisted the main project in having the latest document to parse and serve for queries to the users, without human intervention. Also, managing the documents via our web application can help the system administrators perform various actions on the library remotely, which can help while parsing or when looking for a specific, unparsed Gazette document in PDF form. While these technologies are useful to our project, they can also relate to multiple other fields of science and other computer science projects, where automation and content management can be impactful. Also, we hope that our new, updated Ontology of the Greek Government Gazette will assist developers that manage legal data with their activities.

Despite the research presented in this work and the fact that we have demonstrated the effectiveness of such a platform for Content Management and Web Crawling, we believe that it could be further developed in a number of ways, that are being described below.

6.2 Future Work

6.2.1 Expanding the Content Management System's features

Our developed Content Management System web application, as it stands, offers simple features. Features that include file browsing and downloading and an advanced search, based on filename. Having new ways to manage the Government Gazette's documents is included in future plans. For example, moving all the files into a database that migrates automatically after every download, is a feature to be implemented. Moreover, a function that deletes specific files is an element that we want to look into developing for our system. Lastly, regarding our Content Management System, albeit our search function is working as it should, another one could be developed that, in this case, searches keywords inside the files and not just file names. For this feature to work, we have to make use of a tool that transforms PDF files into plain text ones, removing photographs and graphs, so we can search correctly. Though, as much as this sounds simple, it can be troubling for older Gazette documents that are scanned photographs and do not have a text area like the modern ones.

6.2.2 Expanding the Web Crawler's features

As shown in Figure 4.1, our spider functions based on the webpage that shows the Daily Releases of Government Gazette documents. It helps us keep track of the new releases and gather them quickly and efficiently. While this webpage is suitable for our work, it only shows documents published after 01.01.2000. All the files before this date are not reachable via the daily release page, but do exist on the server. We can retrieve those files via the ET.gr's search page, shown in Figure 3.4. To achieve this, we need to expand the features of our spider to work on this webpage too if a requested crawl includes a date that is before the specified date above. Also, for documents published after 1990, the website offers a text version of the file besides the PDF version. Implementing a feature that lets the user select which version of the file he wants is in our plans, since it would also benefit the keyword search we mentioned in our plans for the Content Management System application in Chapter 6.1.1.

Table 6. 1: List of Abbreviations

API	Application Programming Interface
CS	Computer Science
CM	Content Management
CMS	Content Management System
ELI	European Legislative Identifier
EU	European Union
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
OS	Operating System
OWL	Web Ontology Language
PDF	Portable Document Format
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
URI	Universal Resource Identifier
XML	Extensible Markup Language

Table 6. 2: Table of Translations

Content Management Systems	Συστήματα Διαχείρισης Περιεχομένου
E-Government	Ηλεκτρονική Διακυβέρνηση
Greek Government Gazette	Φύλλο Εφημερίδας Ελληνικής Κυβερνήσεως
Job Scheduling	Χρονοπρογραμματισμός Εργασιών
Open Data	Ανοιχτά Δεδομένα
Semantic Web	Σημασιολογικός Ιστός
Web Crawling	Ανίχνευση Ιστού

APPENDIX

A: Web Application

Software Dependencies

Python: 3.6
Django: 1.11.0
Scrapy: 1.4
Scrapyd: 1.2.0
Scrapyd-client: 1.1.0
Schedule: 0.4.3

Installation

1. Python (https://www.python.org/):

To install the Python programming language in a Windows environment you have to download the installer from the official website (https://www.python.org/downloads/). After that, following the instructions you can install Python and use the Windows Command Prompt (or Powershell) to use the Python Shell or to run Python code (.py files) which we need for this project.

2. Django (https://www.djangoproject.com/):

To install Django Web Framework, you can follow the specific directions given at the official website (https://www.djangoproject.com/download/) or just use the "pip install Django==1.11.6" (which is the latest version at this point). After the installation completes, you can use all the Django related Libraries.

3. Scrapy (https://scrapy.org):

To install the Scrapy Web Crawling framework you can follow the directions given at the official documentation (https://doc.scrapy.org/en/latest/intro/install.html). You can of course use the simple "pip install Scrapy" and you are good to go. As stated in the Official installation guide, Scrapy is written in pure Python so depends on some other key Python packages like lxml, w3lib, twisted, parsel, cryptography and pyOpenSSL.
Scrapy and its dependencies are installed by the pip command given abode.
If by any chance there is a problem related to these dependencies, please refer to the respective installation guides.
lxml: http://lxml.de/installation.html
cryptography: https://cryptography.io/en/latest/installation/
The minimal versions which Scrapy is tested against are:
Twisted 14.0
lxml 3.4
pyOpenSSL 0.14

4. Scrapyd (https://github.com/scrapy/scrapyd):

Scrapyd is an application for deploying and running Scrapy spiders. It enables you to deploy (upload) your projects and control their spiders using a JSON API.
To install Scrapyd, you can follow the instructions given at http://scrapyd.readthedocs.io/en/stable/install.html, or, once again, use "pip install scrapyd".

5. Scrapyd-client (https://github.com/scrapy/scrapyd-client):

Scrapyd-client is a client for Scrapyd. It provides the general scrapyd-client and the scrapyd-deploy utility which allows you to deploy your project to a Scrapyd server.
To install, use pip install git+https://github.com/scrapy/scrapyd-client to get the latest version.

6. Schedule (https://github.com/dbader/schedule):

Schedule is a simple to use API to schedule jobs. Has no external dependencies. To install, use "pip install schedule".

Directories / File Information

I. The main directory is the Django Project directory. Everything in here except some folders and files is related to the website we built.
II. The FEKdownloader directory is the Scrapy Project, the Crawler. In there, the "FEK Library" directory is where the FEK files are crawled into and then shown in the website.
III. the Schedulescrape.py file is the Scheduler. It schedules crawls for specific hours of the day, or days of the week. (Functionality bellow)
IV. the ontology.py file is taking every FEK file title and creating triplets in a text file when executed.

Everything else is Django related.

I. the templates directory contains the .html files of our website.
II. the static directory contains the .css files and some other static files served into our website.
III. the crawl and manager directories are related to the "crawl" and "manager" apps of our website
IV. the FEKsite directory has some configuration files for our website.

Project Function and Setup

To run the project, after installing all the needed packages and having the right dependencies you have to follow the next steps.

a) You need to have Scrapyd running in the background, for all the incoming crawl requests to be fulfilled. So, after downloading the files and unzipping them to a folder, you need to open a terminal and change directory to "../FEKSite/FEKdownloader". This is the directory of the crawler. Here, run the command "scrapyd". By doing that, you have a Scrapyd instance running and you can visit http://localhost:6800 to see the deployed spiders, logs and running/pending scheduled crawls.

b) Minimize the terminal and open another one at the same directory (../FEKSite/FEKdownloader). Now, we need to deploy the spider to the Scrapyd instance. We can do that just by using the "scrapyd-deploy" command. After doing so, we will see a message like this:

```
========================================================================
Packing version 1507839617
Deploying to project "FEKdownloader" in http://localhost:6800/addversion.json
Server response (200):
{"node_name": "DESKTOP-XXXXXXXX", "status": "ok", "project": "FEKdownloader",
"version": "1507839617", "spiders": 1}
========================================================================
```

This means that the spider was deployed successfully. You can also check the http://localhost:6800, under "Available projects" for our project to be there.
You can use the same terminal to schedule a job to see if everything is working correctly by using "curl http://localhost:6800/schedule.json -d project=FEKdownloader -d spider=downloadFEK"
If you need to delete the project from the Scrapyd instance, you can use the "curl http://localhost:6800/delproject.json -d project=FEKdownloader" command.

c) Now that Scrapyd is up and running with the crawler deployed, it is time to fire up our website. We can do so by navigating to ../FEKSite and opening up a terminal there. Then, we use the "python manage.py runserver" command of Django to run the developmental server on our localhost and our website is fully functioning at http://localhost:8000. From there we can login using the following account: username: admin // password: password123 and start using the website services. By going into the "Crawler" tab, we can send crawl requests to the Scrapyd instance running in the background and download our files using the web interface.By going into the "Manager" tab, we can browse our downloaded files. The files displayed are from the "../FEKSite/FEKdownoader/FEK Library" directory.

This concludes the Setup and usage process of the website.

d) (Optional) If we want to Schedule jobs(crawls) for a specific time every day or a specific day every week, we can modify the Schedulescrape.py file accordingly.The file includes some commented code lines that show the way the scheduler works. The default schedule plan is for the crawler to run every day at 14:30. So, having a Scrapyd instance running the way we talked about in step a) and a deployed spider into the instance (step b), simply execute "python Schedulerscrape.py" and it will work until stopped by a kill signal from keyboard.

REFERENCES

[1] Ilias Chalkidis. Nomothesi@: Greek Legislation Platform. Technical report, National and Kapodistrian University of Athens, 2014.
[2] Pau Valles Fradera. "Personalizing web search and crawling from clickstream data", 2009. http://upcommons.upc.edu/bitstream/handle/2099.1/6258/Pau%20Valles.pdf?sequence=1
[3] "GoogleBot", https://support.google.com/webmasters/answer/182072?hl=en
[4] Joao Leonardo Vicente do Carmo. "Web Content Management Systems: Experiences and Evaluations with the WebComfort Framework", University Of Lisbon, 2007. http://isg.inesc-id.pt/alb/static/students/msc-thesis/2007-jleon-MScThesis.pdf
[5] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. "Operating Systems: Three Easy Pieces", 2014, Chapter 7 “Scheduling: Introduction” http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched.pdf
[6] ELI Task Force, Publications Office (European Commission). "ELI implementation methodology: Good practices and guidelines", 2016. https://publications.europa.eu/en/publication-detail/-/publication/a8367080-bdad-11e5-bfdd-01aa75ed71a1/language-en
[7] ELI Task Force, Publications Office (European Commission).” ELI: A technical implementation guide”, 2015. https://publications.europa.eu/documents/2050822/2138819/ELI+-+A+Technical+Implementation+Guide.pdf/
[8] Monica Palmiran, Roger Sperberg, Grant Vergottini, Fabio Vitali. "Akoma Ntoso Version 1.0. Part 1: XML Vocabulary", 2017. http://docs.oasis-open.org/legaldocml/akn-core/v1.0/cs01/part1-vocabulary/akn-core-v1.0-cs01-part1-vocabulary.pdf
[9] Official Greek Government Gazette, http://www.hellenicparliament.gr/en/Vouli-ton-Ellinon/I-Bibliothiki/Koinovouleftiki-Syllogi/Efimeris-Tis-Kyverniseos-FEK/
[10] J. Eder, I. Rozman, and T. Welzer. Advances in Databases and Information Systems: Third East European Conference, ADBIS’99, Maribor, Slovenia, September 13-16, 1999, Proceedings. Advances in Databases and Information Systems: Third East European Conference, ADBIS’99, Maribor, Slovenia, September 13-16, 1999 Proceedings. Springer, 1999. https://static.aminer.org/pdf/PDF/000/026/087/evaluation_of_data_modeling.pdf.
[11] Yugandhara Patil, Sonal Patil. "Review of Web Crawlers with Specification and Working", 2016. http://www.ijarcce.com/upload/2016/january-16/IJARCCE%2052.pdf
[12] Carlos Castillo. "Effective Web Crawling", University of Chile, 2004. http://chato.cl/papers/crawling_thesis/effective_web_crawling.pdf
[13] Dimitrios Kouzis-Loukas. "Learning Scrapy", 2016.
[14] "Scrapy Documentation", 2017, http://scrapy.readthedocs.io/en/stable/intro/overview.html
[15] "Scrapyd Documentation" http://scrapyd.readthedocs.io/en/stable/install.html
[16] Adrian Holovaty, Jacob Kaplan-Moss. "The Definitive Guide to Django: Web Development Done Right", 2009
[17] Jeff Forcier, Paul Bissex, Wesley J Chun. "Python Web Development with Django" 1st edition, 2008
[18] W3C. Semantic Web. https://www.w3.org/standards/semanticweb/