National Technical Univercity of Athens

School of Electrical and Computer Engineering
Software Engineering Project Proposal
Project Title: Spyglass
Academic Year 2012-2013

Alex Maurogiannis (03109677)
Gregory Lyras (03109687)

January 17, 2013

1 Overview

Instead of the default Software Engineering Course project we present this following project proposal. With the advances in computer science and the development of the World Wide Web it has been clear that the amount of information one can find is more than one can process. What's more, it's hard to monitor websites for topics of interest manually, especially due to the fact that their content may have a much more broad spectrum than one's interests. In addition, the number of sites that might contain useful information is vast. As a result we rely on search engines to provide us with a limited number of web pages based on our own search criteria. This approach serves us well when one actually wants to find some specific pieces of information. However, it is seriously lacking when one simply wants to get new pieces of information on his subjects of interest as they appear.

This project aims to address the aforementioned issue in a different manner. Instead of searching on one site, we can monitor sites of interest and filter their updates based on keywords provided. If the updates match the search criteria then the end user is notified for the latest developments. The proposed project is a meta-search engine used to monitor sites for required information and provide notifications to its users accordingly.

2 Components and Interactions

The project consists of the following three main components.

- A web application framework providing the user interface and the database handling.
- A distributed data collection system (crawler network).
- A notification system.

2.1 Framework

The framework allows a developer to produce a configuration of the web application in order to provide an interface to the end user, declare the desired sites to be monitored and define the methods of the searching process. It communicates directly with the user through a developer-created interface in order to receive monitoring requests. It also guides and monitors the data collection system, providing the system administrator with control options over the collection process.

2.2 Data Collection System

The data collection system is a distributed network of crawlers which poll the framework's server on predefined intervals in order to receive a workload. They then proceed to process the workload by searching on the sites for the required data and send the results back to the server.

2.3 Notification System

The notification system provides email services. When a notification is triggered after a request is fulfilled, the notification system is directed by the framework to send an email to the end user with the information he requested.

3 Generic Use Cases

3.0.1 Developer Case

The developer configures the framework to provide a front-end to the end user and a collection of websites which will be crawled for the requested data. For each website, he will need to provide the corresponding instructions so the crawlers will be able to locate the data of interest. Additionally, he has full control over which subsets of the website collection will be available to the end user.

3.0.2 End User Case

The end user interacts with the web application and the notification system. He specifies the search terms on the web user interface and selects the sites to be monitored, as configured appropriately by the developer. The framework then handles the user request and forwards the necessary information to the crawler network. When an appropriate result is found, the user is contacted by the notification system.

4 Typical End User Scenario

In a system configured to track news sites, the end user visits the service's home page which consists mainly of an input form. There, he inserts the keywords he wishes to monitor, and selects the news sites that will be crawled for those keywords from a predefined list. After submitting the form he may choose to create an account by verifying his email address and track the progress of his query, or modify it. When a result is found, he receives an email and his query optionally may be removed from the system. Finally, the notification email includes a link to allow him to provide feedback on the quality of the overall service.

5 High-Level Description

The framework is structured in a Model-View-Controller architecture, while the other components are each structured in one class.

The Model is composed by three classes of objects: Site, Query and Crawler. The Site class is populated on system configuration, and defines the total of sites and data fields that can be monitored. The Query class consists of objects that are created when the user makes a request, and are completed by a crawler when a result is found. The Crawler class holds all crawler-related data for control over the network.

Controllers are responsible for creating Query objects on the end user's requests, and for the safe interactions of the crawlers with all of above classes.

The View is a generic web interface, meant to be extended by the developer to provide the final user interface.

6 Project Specifications

The final system to be delivered as part of the course should consist of the following:

- The generic web framework in the form of a Django application
- The crawler implementation.
- The notification system.
- All the above configured in a usage example along with a user front end running on a production server.

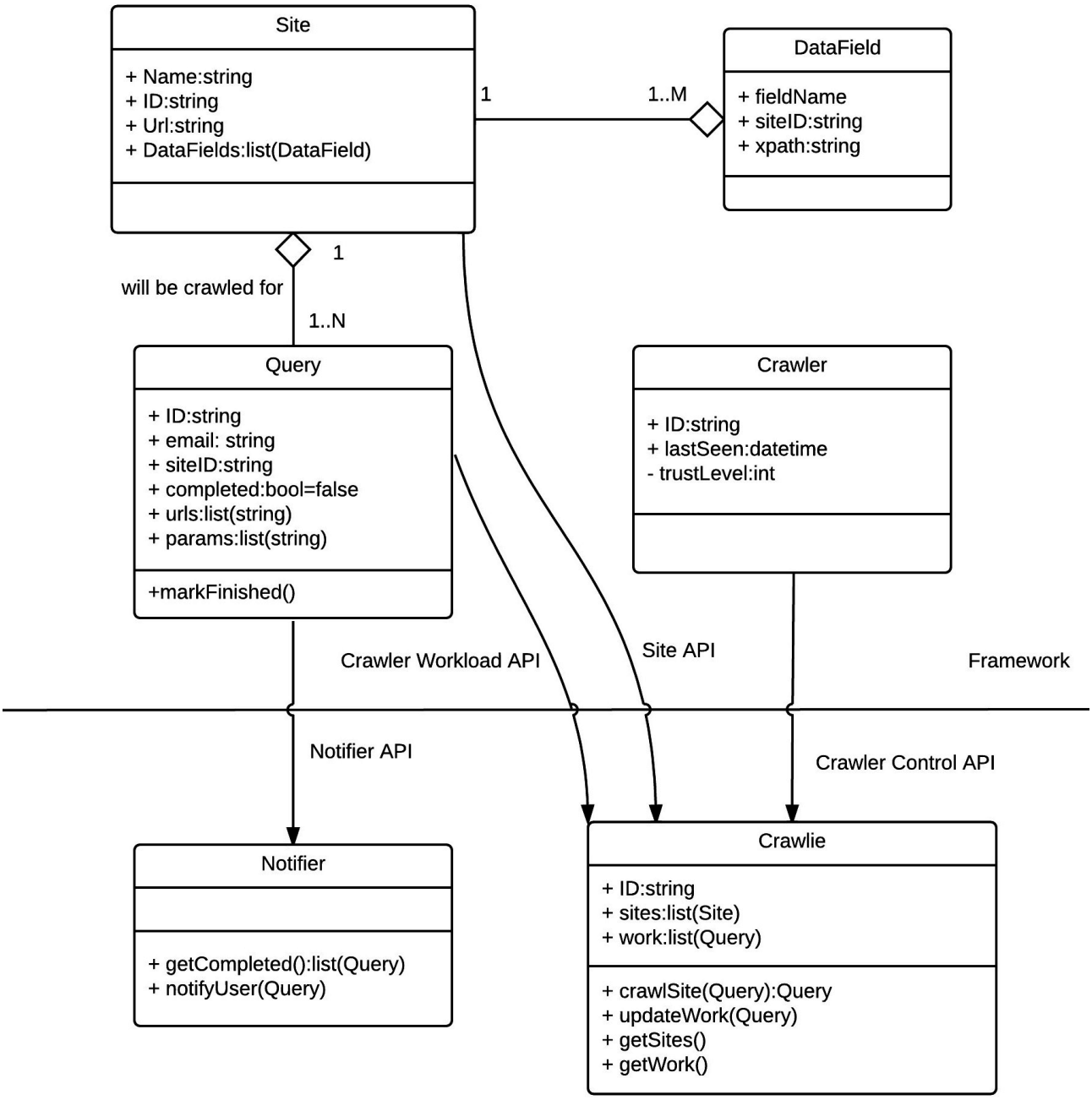

Figure 1: High Level Diagram