# Contextualization Service

The Contextualization Service determines the context (topics) of the webpages by matching text content against a dictionary of significant words and weighting those matches, allowing ads or content to be targeted according to context.

# Overview

The Contextualization Service works in two parts:

1. **Determining and Saving a Contextual Profile:** The page scraper attempts to discern the context of a page (what it’s about). It hits URLs of pages within our publisher network, “scrapes” the content from each page, compares what’s found in that content against our database of health-related keywords, weights matches according to how often a keyword is found and where it is found (page text, vs meta tags, vs title tags, vs the URL), and then stores the results for that page as a “contextual profile” in the database.
2. **Retrieving a Contextual Profile for Targeting:** Before placing a contextually-sensitive ad or piece of content, we retrieve the “contextual profile” for the current page, and (in the case of ad targeting) inject the contextual terms for that page as DFP targeting key-values, allowing us to either positively or negatively target ads based on the context of the content.

# Architecture

# 

## Execution Workflow

## 

### Step-by-Step

1. **Receive request:**The Contextualization Service receives an HTTP request, with the url query string parameter set to the URL for which to retrieve a "context." A context is a JSON object consisting of keywords found for a page, and their scores. The service checks the Contextual database to see if a document already exists for the URL.
2. **Context is found:**If a context is found in the database for the URL, it returns it as JSON.
3. **Context is not found:**If a context is not found in the database for the URL:
   1. **Fetch URL content:**The service makes a server-to-server HTTP request for the URL.
   2. **Parse data sources:** The response is parsed using the jsDom Node.js module, saving the values of the following data sources:
      1. URL: the original page URL specified.
      2. meta keywords: the of the content attribute of the <meta name="keywords"> tag.
      3. meta description: the of the content attribute of the <meta name="description"> tag.
      4. title: the value <title> tag
      5. body: the value of the <body> tag, with script, style, and link tags stripped out.
   3. **Determine context:** Compare data source values to the list of keywords in the dictionary.
      1. **Score matches:** Exact keyword matches are scored according by multiplying the number of matches in a given data source times the weight for the data source as follows:
         1. URL matches: weight = 5. Note that this is the only data source for which an exact match is not necessary. For instance, the word "tween" would be found within the word "between," whereas in any of the other data sources, that would not be a match (regular expression word boundaries are observed).
         2. meta keyword matches: weight = 5
         3. meta description matches: weight = 5
         4. title matches: weight = 4
         5. body matches: weight = 1
   4. **Format and return JSON:**The JSON response payload is defined, populated, and returned as follows (see the "Parameters" heading within the next section, "Sample Request URL"):
      1. **Via a variable:** If the variable query string parameter was passed, the JSON is output as the value of a JavaScript variable declaration.
      2. **Via a callback:**If the callback query string parameter was passed, the JSON is output wrapped in a call to a Javascript function with the same name.

## Sample Request URL

http:// contextservice.elasticbeanstalk.com/xyzcompany/?url=http://www.mayoclinic.org/&variable=ap\_context.

### Parameters

* ***url***: Required. The address of the page from which to gather context data.
* One of the following is required:
  + **variable**: Name of a variable with to which to set the JSON response (the variable declaration is included in the output).

**Sample response:**

//For URL

//http://contextservice.elasticbeanstalk.com/xyzcompany/?url=http://www.cdc.gov/&variable=context

var context = {JSON Payload...};

* + **callback**: name of a callback in which to wrap the JSON response, resulting in a standard JSONP response.

**Sample response:**

//For URL

//http://contextservice.elasticbeanstalk.com/xyzcompany/?url=http://www.cdc.gov/&callback=doCallback

doCallback({JSON Payload...});

## Sample JSON Payload in Response

{

"id": "5580dccdd95356c373000003",

"url": "http://www.cdc.gov/",

"canonical": "http://www.cdc.gov/",

"matches": {

"healthy living": {

"count": 1,

"score": 1

},

"tb": {

"count": 3,

"score": 7

}

},

"sources": {

"url": [],

"link|canonical": [],

"title": [],

"meta|og:title": [],

"meta|description": [],

"meta|og:description": [],

"meta|keywords": ["tb"],

"meta|og:keywords": [],

"body": [

"healthy living",

"tb"

]

},

"dateCreate": "2015-06-17T02:31:37.156Z",

"timeForRequest": "1210ms"

}