ABSTRACT

A SEMI-AUTOMATIC KNOWLEDGE ACQUISITION SYSTEM FOR SEMANTIC WEB APPLICATIONS

Uluç, Fuat Önder

M.S., Computer Engineering Department

Supervisor: Asst.Prof.Dr. Çiğdem Turhan

Co-Supervisor: Asst.Prof.Dr. Nergis E. Çağıltay

April 2008, 114 pages

As the information and documents accessible through the Internet grows without any boundaries, accessing to specific information and documents becomes more and more difficult and in near future it will almost be impossible. The standard means available, will not be capable to sort the needed information or document among the lists provided which may practicaly be classified as endless. Therefore, to facilitate the access to the required documents or information the Semantic Web concept appears to be an available solution. As a natural consequence of this fact, automatic or semi-automatic knowledge acquisition methods have to be developed.

In this thesis, a semi-automatic knowledge acquisition system for the Semantic Web is developed and determining the problems encountered in the creation of such a system and suggesting possible solutions to these problems is aimed. The domain of the developed knowledge acquisition system is not limited to a specific area; the user can guide the system to acquire the knowledge of a required topic. The system is tested on university web sites and the results are evaluted.

Keywords: Semantic Web, Knowledge Acquisition

ÖZ

ANLAMSAL AĞ UYGULAMALARI İÇİN YARI-OTOMATİK BİLGİ EDİNME SİSTEMİ

Uluç, Fuat Önder

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Çiğdem Turhan

Ortak Tez Yöneticisi: Yrd. Doç. Dr. Nergis E. Çağıltay

Nisan 2008, 114 sayfa

İnternet ile ulaşılabilen bilgi ve doküman miktarının sınır tanımaz şekilde büyümesi, belirli bir doküman veya bilgiye ulaşabilmeyi gittikçe zorlaştırmaktadır ve yakın gelecekte neredeyse imkansız hale getirecektir. Mevcut standart araçlar karşılaşılan ve pratik olarak sonsuz sayılabilecek listedeki bilgi ve dokümanları tasnif edebilecek yetenekte değildirler. Bu nedenle, gerekli bilgi ve dokümanlara ulaşabilmeyi kolaylaştıracak mevcut çözüm olarak Anlamsal Ağ kavramı ön plana çıkmaktadır. Bu durumun doğal sonucu olarak, otomatik veya yarı otomatik bilgi edinme yöntemleri geliştirilmektedir.

Bu tez çalışması kapsamında, Anlamsal Ağ için yarı otomatik bir bilgi edinme sistemi geliştirilmiş ve benzer bir sistem oluştururken karşılaşılan sorunların belirlenmesi ve bu sorunlar için muhtemel çözüm önerilerinin oluşturulması amaçlanmıştır. Geliştirilen bilgi edinme sisteminin alanı belirli bir kapsam ile sınırlı değildir; kullanıcı, sistemi istenen konuda bilgi edinebilmek için yönlendirebilecektir. Sistem, üniversitelerin web siteleri üzerinde test edilmiş ve sonuçlar değerlendirilmiştir.

Anahtar kelimeler: Anlamsal Ağ, Bilgi Edinme

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Asst.Prof.Dr. Çiğdem Turhan for her guidance and sharing her knowledge. Thanks also go to my parents and my fiance for their patience and support during this study.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Today the most important communication and research tool used is the Internet. The Internet provides countless documents and a limitless communication opportunity.

The unpreventable growth of the Internet also creates unpreventable problems. The drastic increases in numbers of the Internet users and published documents naturally results in day by day increase of difficulty for accessing the sought information.

The Semantic Web technology is one of the solutions offering an easier access to the desired data over the Internet. The Semantic Web is a new presentation method for the data which can be recognised and followed by software. For creating a Semantic Web system, knowledge should be acquired manually or automatically and this knowledge should be represented by a Semantic Web tool such as Ontology Web Language (OWL). Knowledge acquisition means eliciting the expertise from domain experts and classifying it for expert systems [1].

The target of this thesis is developing a semi-automatic knowledge acquisition system for the Semantic Web, determining the problems encountered in the creation of such a system and suggesting possible solutions to these problems. To achieve the above mentioned target, the development of a Semantic Web frame is also aimed at to represent the acquired knowledge.

A network is a system with more than one computer or other network devices connected to each other. Internet can be defined as the network of networks. Through the Internet, millions of computers, computer systems and other networks or communication devices are connected to each other [2]. This connection can be made

by cables, satellites, radio waves and similar technologies used in telecommunication. The name given to this global network is World Wide Web (WWW) [3].

Through the Internet, transfer of mails or files between users, broadcasting video or radio streams, sharing information, reading papers and books, shopping, making phone calls, performing research and even establishing diaries are possible.

Briefly, the Internet is an open information source with unlimited documents and the most important communication tool with its millions of users [3][4].

One of the most important elements of the Internet is the web page. Web pages are coded with a special mark-up language known as **H**yper**T**ext **M**ark-up **L**anguage (HTML), a special form of e**X**tended **M**ark-up **L**anguage (XML) [3][6]. XML is a tag based language, where the information is enveloped and carried in the tags. In XML, a tag can have any name that the author appoints. Whereas, in HTML: a special form of XML used to create hypertexts, the tag names are specific and every tag has a meaning [5][6]. The web browsers know the meanings of HTML tags and convert the source code into an interface for the viewers of the web page in accordance with these meanings. The important aspect here is the format for carrying the information being covered by the web standards about the web pages but having no explanations about the information itself [7].

In the contemporary world, computers no longer function only as single isolated information processing machines. They are also entry ports to the global network of information where users can exchange data or make business transactions [8]. The World Wide Web is widely used for communication, research and commerce. Although it is being one of the basic communication tools in modern society, the capacity of the web is limited with the ability of the users. Thus, data for

support, information technologies and knowledge exchange are key concepts for the current computer technologies governing the efficiency of the user's ability [9].

The World Wide Web contains millions of documents and web pages which are used by millions of internet users around the world. Both the number of users as well as the amount of information are growing drastically every day [4][10][11]. When World Wide Web was started by limited information in the 90s, there were not that many of users accessing information. In time, the number of the users has increased and the available information on the web increased in parallel. However, the problem has come up with selecting, accessing and reaching the available data which is specific to the user's needs [10][11]. In the web, the presentation of the information is in natural language which results in increasing difficulty of finding the required information. For that reason, there is a gap between the information available for software agents and the information for the human users [12][13].

When users search on the web they access thousands of pages where most of them are not related with their research. Moreover, the researchers have to read this information in order to reach the "most desired" one. With World Wide Web, the scope of electronic information has changed widely. However, this wide range of information and growth of the web made it very difficult to find, access, present and maintain the information required by most of the users [10][11].

The list of information that web pages can include is an endless list. It is only limited with the imagination of mankind. Since there is no limitation for the contents of the web pages, an individual can find any information about anything in the Internet. This publishing freedom causes information pollution with millions of out-dated documents, repeating documents and corrupted documents.

People usually use search engines such as Google to find the information they are seeking. Assume that a researcher is trying to make a search about an insurance company called “Ankara Insurance”. When the researcher types the words “Ankara” and “insurance” to a search engine, search engine will return the web pages about the city Ankara, the insurance companies that are in Ankara, maybe a comment on an insurance company in a forum by a user whose nickname is Ankara, or a comment about Ankara by a user whose nickname is insurance, an article which contains both of the words, lots of broken or corrupted links and the thousands of documents about the “Ankara Insurance” company. But the researcher needs only one of those links or documents. However, the endless list with Ankara, insurance or the combination makes it impossible to reach that only one needed. This is a major problem of the Internet. Everyday thousands of documents are being uploaded and most of them are useless for the majority of the users and again most of them are repeating documents which already exist in the World Wide Web.

The reason for this problem is that even if the information carried by the hypertext documents could be meaningful for the viewers of the web pages, software-wise, the information has no meaning, therefore, when a researcher asks a software to bring back a specific information, it is not possible for the software to find the exact information due to the fact that the information does not carry any meaning that the software can understand [7]. If there is going to be a future for the World Wide Web, users should be able to find what they are seeking in this endless garbage which is getting more difficult by the day [10][11].

This fact led many new searching tools and commercial enterprises to classify available information by making it processable for the machines. These tools are essential for making web more beneficial for the users. A possible solution for the

problem of the Internet is giving a meaning to the information that software can recognise and understand. In other words, the solution could be giving semantics to the information carried by the hypertexts. The Semantic Web technology is the product of this thought and the Semantic Web tools have been recommended by the World Wide Web Consortium since 2004 [3][4][14].

In the following two chapters, the literature background of this thesis is detailed. Semantic Web overview and the problems that arise in its development and ontology concept are presented in Chapter 2 and knowledge acquisition overview is presented in Chapter 3. Chapter 4 presents the design of the system and Chapter 5 presents its implementation. Finally, the conclusion and future work proposed for this thesis is given in Chapter 6.

2. SEMANTIC WEB

2.1 Overview

Two main functions which the original web lacks are targeted by the Semantic Web. Integration and combination of data drawn from various sources by using common formats are primarily targeted by Semantic Web, whereas, the original Web mainly concentrates only on the transfer of documents. The second target is the methodology for recording how the data and real world objects relate to each other. These two targets when achieved together, allows a person or a software agent to navigate between databases which are connected to each other [3] [14] [15].

Consequently, the Semantic Web provides to reach the huge amount of information with wide variety, by enabling certain kind of software agents to interact between researchers' needs and information sources [13].

With the Semantic Web, the World Wide Web becomes readable by the machines and it allows computers to use this information to differentiate between information from the other resources to provide only the necessary ones to the researcher [14][16][17].

By increasing the web page content it becomes possible to understand the basis for action about that web page. The impact of the Semantic Web is so large that it is almost considered as the reinvention of the World Wide Web's infrastructure on at least the scale of the original web. For the time being, a limited amount of technology is used to make exploratory and demonstrative applications. When the technologies for the Semantic Web develop more, it is certain that it will grow with an enormous speed [14] [18].

2.2 Ontology

Ontologies are the essentials of the Semantic Web technology as they provide an important base [17][18]. Ontology makes semantics understandable by software. Ontology is developed with artificial intelligence techniques to make knowledge acquisition and reusing of information easier. An ontology is an expert system where the concepts of domain and their relations with each other are represented. Since the beginning of World Wide Web in 90s, ontologies have become a popular research topic investigated by artificial intelligence researchers such as Knowledge Engineers [4][19][20].

The ontology concept is getting widely used in computer science and information system research areas[15][17][21][22]. Offering a shared and common understanding which can be communicated between people and application systems, the ontologies have become very popular. Ontologies aim to present a consensual knowledge of domain. Their development is often a cooperative process which involves various users at various locations [22][23][24][25].

In an ontology there are classes where the information can be named and there are properties where the classes can be privatized with attributes. Another important feature of an ontology is the relation between classes. If giving semantics to data is desired, these relations are a must [24][26].

To achieve the full capacity of the Semantic Web, meta-data and knowledge from different sources should be combined for data reasoning [23][27]. Ontologies may be different from each other in many ways: from vocabulary differences to concept differences. To combine these ontologies, many of those differences must be solved on some basis. Ontologies can be combined to create a new ontology from two or more existing ones [4].

There are some drawbacks in the ontology usage [4];

1. **Evolving ontologies**: Since ontologies change and have to be updated regularly, they are unstable. Because of this change, the reuse of ontologies may create problems.
2. **Combining ontologies**: Integrating different ontologies to create a new ontology causes various problems.

In an ontology, the author can define the classes, their attributes, data types of these attributes (e.g. string, integer) and relations or inheritance between the classes. In other words in the ontology, the author can prepare the outline of the information that is going to be represented.

After preparing this outline, the author also installs the information to the same ontology by creating individuals of the classes and writing the values of the properties that the class has for that individual [24].

For example, if the author is trying to create an ontology which represents the information about students and their advisers, two classes: adviser and student are required. The adviser class may have the properties; title, name and surname and the student class may have the properties student ID, name and surname. Since both classes will have people as individuals and have some similar properties as name or surname, it would be efficient to create a person class with properties; name and surname, and both the adviser and the student classes may inherit to person class.

Class: Person

Property_1: name (string)

Property_2: surname (string)

Class: Adviser is a subclass of Person

Property: title (string)

Class: Student is a subclass of Person

Property: student ID (integer)

This way, because both adviser and student classes inherit from the person class, they both carry the properties of their mother class. After building this class hierarchy, the author should also define the data types of the properties and the relation between classes. The relation between student-person and adviser-person is that person class is the mother class of the other two. The relation between the student and the adviser class can be named like Adviser **advises** student. Also the data types of the attributes of the classes must be defined. Name, surname and title classes should be string data type and student ID property may be integer.

After preparing this information outline, the individuals will be created.

1st individual: is a **Student**

Name: John

Surname: Smith

Student_ID: 123456

2nd individual: is a **Student**

Name: Smith

Surname: John

Student_ID: 123457

3rd individual: is an **Adviser**

Name: Jane

Surname: Brown

Title: Dr.

Advises: 1st individual

4th individual: is a **Student**

Name: Mary

Surname: Smith

Title: Prof.Dr.

Advises: 2nd individual

This information can be easily represented with an ontology and this data will have a meaning also for the software. The software will know that name is the

property of Person, Student and Adviser classes and 3rd individual belongs to the Adviser class, so it has four properties. It should have a **"name",** a **"surname",** a **"title"** which are all string and it should "**advises"** another individual which belongs to the "**Student**" class.

Ontologies have some similarities with other computer technologies. To define what an ontology is, it is also necessary to discuss the differences of an ontology from similar systems like databases.

Firstly, an ontology does not store the data but gives a map for the stored data unlike a database. In databases the data is stored inside and when a user makes a query, database returns the exact information that matches the conditions of that query. An ontology does not return the exact information. Instead it has a reasoning process and it generates the data it will return according to the ontology representation [25][28].

Ontology technology has some similarities also with the object-oriented paradigm. Like object orientation, ontologies also have classes and an inheritance between classes where mother classes share their properties with their child classes. But unlike object-orientation multi-inheritance is possible in ontology technology [26] [29].

2.3 Tools Used in Semantic Web Technology

2.3.1. XML

XML refers to the extensible mark-up language. It is a meta-language that meets the requirement of defining the mark-up languages specific for certain applications. XML is used as a tool to represent other languages in a formal form. XML provides a data format to define structures in a document, without a specific

vocabulary. XML Schema is for specifying this vocabulary and combining tags [4][5][6][30].

Each XML document is made up of an ordered, labelled structure. This generality refers to XML's pros and cons. All kinds of data structures can be encoded in an explicit syntax, but on the other hand XML does not specify the usage and the meaning of the data [4][5].

2.3.2 RDF and RDF Schema

RDF is a World Wide Web Consortium recommended framework which targets to give descriptions about data available in the World Wide Web. While XML targets to structure data, RDF targets to give information about the data. These data are called as resources in RDF. RDF presents metadata: data about data. Metadata allows search engines to filter out the unnecessary information and return better results [4][31][32].

RDF is designed to have a simple data model and standard semantics using an XML-based syntax to allow anyone to make descriptions about any data on the web [3].

RDF Schema is a vocabulary extension for RDF. While RDF only describes the properties of the resources, RDF Schema language allows defining the types of resources also. By the help of RDF and RDF Schema concepts of a domain can be built [3][4][31].

2.3.3. OWL

OWL refers to Ontology Web Language. OWL has a better machine interpretability than RDF, and RDF Schema (RDF-S) by an additional vocabulary like cardinality or symmetric properties, and it is possible to say that OWL is an extension of RDF [3][33].

The representation of the example ontology given in section 2.2 with OWL is shown in Appendix A.

In the ontology with the following lines;

“<owl:Class rdf:ID="Person"/>

<owl:Class rdf:ID="Adviser">

<rdfs:subClassOf rdf:resource="#Person"/>

</owl:Class>”

The classes are created and their hierarchy is given. In the first line the Person class is created. In the second line the Adviser class is created and on the third line Adviser class is defined as a sub class of class Person. On the last line, the tag for creating the class is closed.

“<owl:DatatypeProperty rdf:ID="Name">

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>

<rdfs:domain rdf:resource="#Person"/>

</owl:DatatypeProperty>”

The above lines show how to define the name attribute of the person class. There are two types of properties in OWL. One is the data type property which is an attribute of a class as in Person-Name [3]. In the above lines, in the first line the name of the attribute is given. In the second line its data type is given. On the third line its class is given and on the last line the tag for that data type property is closed.

The other type of property is the object property which defines the relation between two classes as seen in the following example [3].

“ <owl:ObjectProperty rdf:ID="advises">

<rdfs:range rdf:resource="#Student"/>

<rdfs:domain rdf:resource="#Adviser"/>

</owl:ObjectProperty>”

Name of the property is given in the first line as “advises”. Domain of the property is Adviser class as given in the 3rd line and the range of the property is Student class as given in the 2nd line.

In this thesis, OWL is selected as the ontology tool because it is the most commonly used tool and has the newest vocabulary [3].

2.3.4. Other Languages

RDF Schema is quite simple compared to other languages such as knowledge representation languages. To be able to specify the meaning of data more precisely, richer languages are necessary, such as OIL13. OIL has been developed with the aim of combining “intuitive modelling primitives, Web-languages, and formal semantics into one language” [4]. OIL’s “onion model” is made up of a complex structure which allows applications to select the degree of complexity they need. Based on OIL“DAML+OIL14” has been developed which is mainly used by European and North American modelling primitives for the Semantic Web [4][23][34][35].

In addition to the languages described in the above sections, there have been other languages used in the development of the Semantic Web.

A selection of other Semantic Web languages:

XOL: XML-based ontology-exchange language [36]

Topic Maps: “ISO standard for describing knowledge structure” [37]

SHOE: Simple HTML Ontology Extensions [38]

2.4. Previously Developed Semantic Web Applications

Neurocommons: The Neurocommons is a RDF database released by Science Commons. It was created from life sciences databases with a focus on neuroscience [39].

Foaf: A Semantic Web application which describes relationships among people and other agents using RDF [40].

Sioc: Semantically-Interlinked Online Communities provides a vocabulary of terms and relationships that model web data spaces. Examples of such data spaces include, among others: discussion forums, weblogs, mailing lists, shared bookmarks, image galleries [41].

Simile: Semantic **I**nteroperability of **M**etadata and **I**nformation in un**L**ike **E**nvironments. SIMILE is a joint project to improve interoperability among digital entities, vocabularies, ontologies and meta data, directed by the MIT Libraries [42].

Biomed: Created by the National Centre for Biomedical Ontology to develop methods that allow scientists to create, publish, and manage biomedical information and knowledge in a form that software understands [43].

3. KNOWLEDGE ACQUISITION

The process of;

- eliciting data
- analyzing data (by a rule based system)
- transforming data into knowledge
- classifying knowledge
- organizing knowledge
- integrating knowledge
- representing that knowledge in a form that can be used in a computer system

is called knowledge acquisition [1][44].

Many information extraction systems can capture the related content in documents such as:

“Atatürk” is a person or “19 May 1881” is a date. However, such information is not very useful without the information that Atatürk was born on 19 May 1881. Extracting such relations automatically lets us acquire more complete knowledge to populate the ontology.

The first phase of knowledge acquisition is eliciting the data. While eliciting the data about a specific domain, supervision of domain experts is required. Interacting with the domain experts and understanding them is not a simple task. First of all the domain experts are usually important and busy people and in most cases, because one person does not know everything about a domain, more than one domain expert is required to describe a domain. Secondly, because experts have a vast amount of knowledge and most of this knowledge is tacit knowledge, experts would have

some difficulties to describe their knowledge and sometimes they might not know what they really know. Some knowledge acquisition forms or tools would be necessary to guide both the knowledge engineer and the domain expert in the eliciting data process. Most of the knowledge acquisition works start with wrong beginnings and incomplete information [1][44][45].

The work of the knowledge engineer is not only getting the information from the expert. Since the engineer has less knowledge about the domain than the expert and it is most probable that the expert is using a complicated vocabulary, it is necessary to convert the knowledge gathered from the expert into knowledge understandable by non-experts. The engineer and the expert should work cooperatively to refine the vocabulary [45].

The knowledge gathered from different experts should be validated and combined together by the knowledge engineer.

Different methodologies have been developed for eliciting the knowledge from an expert. For describing different types of knowledge with text based information, such as interrelations and characteristics, protocol generation techniques including various types of interviews and protocol analysis techniques are used. For building taxonomies or similar hierarchical structures such as goal trees and decision networks, laddering like hierarchy generation techniques are used. Facing problems against possible solutions, matrix-based techniques by constructing grids are required indicators. In main matrix – based techniques frames are used for representing the characteristics of concepts and for eliciting, rating, analyzing and classifying the properties of concepts repertory grid techniques are used. For complying with the user's way of comparing and ordering concepts, sorting techniques are developed

which lead to the eliciting of knowledge about classes, properties and priorities [1][44][46].

By populating the ontology through automatic-knowledge-acquisition software, the users bring the necessary information together from the Web's wide range of information. For most of the domains, the interviews with experts will help to build the concepts of the domain. After that an automatic knowledge acquisition system is required. Without an automatic knowledge acquisition system, it is not an easy task to create an ontology and keep it up to date. Unlike databases, ontologies do not need to store information of only one web site. Because the goal of creating an ontology is providing easy access to information, it is more logical for an ontology to hold the information of as many web sites as it can. Even if it is possible to build an ontology manually, keeping it up-to date by performing knowledge acquisition regularly would be a very difficult task [15][47].

Since the work area of an automatic knowledge acquisition system for Semantic Web is the Internet and the information sought is represented in HTML source codes, the raw data should be taken from the source codes of the web pages [48][49].

After building the concepts with the help of experts and taking the raw data from the web, the data should be analyzed to comb out unnecessary information by the help of a rule-based system. This should include keyword search with the help of a rule-based logic.

For a data to be knowledge, the data should be understandable by software. To convert the data into a machine-understandable format, the data should be classified, organized and integrated in a specific format. This conversion should also be done

with the help of a rule-based system. After the data is represented in a way that the computer can understand it, knowledge acquisition is completed [44].

4. DESIGN OF THE SYSTEM

4.1 Domain

The target of this thesis is to create a semi-automatic knowledge acquisition system for the development of an ontology that includes the acquired knowledge. The domain that the system is tested on is the university web pages and the desired knowledge is information about the staff, a university department and information about the department itself. The same system can be applied as a general knowledge acquisition system that can work in any domain.

Because, creating a detailed ontology about the universities is not the main concern of this thesis, the ontology structure designed is relatively simple. The information ontology holds about the staff is limited with the title, name and the surname of the staff and information about the department is limited with the department name and the relation of the staff individuals with the department.

As stated above, the sample domain to be tested is chosen as the web pages of a university, and Atılım University's and Ankara University's web sites are chosen for the first sample runs.

4.2 General Structure of The System

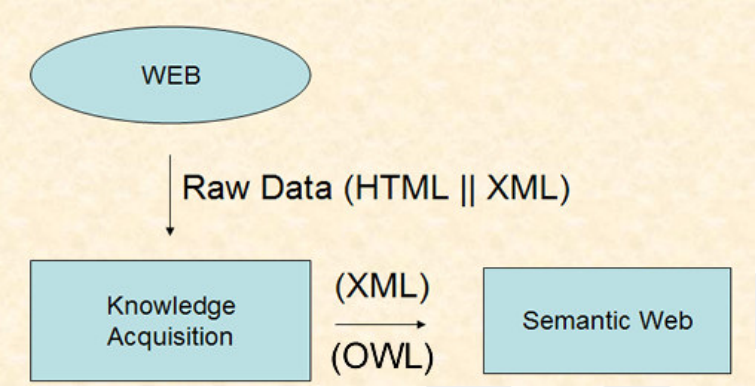

Figure 4.1 Top Level Design

As seen in Figure 4.1 the knowledge for the ontology is acquired from web pages such as www.atilim.edu.tr. These web pages are marked up by HTML into hypertexts, so the first module of the system is parsing the data (source code of the web page) by the help of a parser.

After the data is parsed from the web, the data is split into words. This is required because the only way to find the knowledge inside the raw data is keyword search.

After deciding on a split method such as “Split according to spaces between words”, keyword search is done. For keyword search, a keyword database is required but if a general system is what is talked about there cannot be specific keywords. Keywords should be selected according to the work domain and as mentioned before there cannot be a specific domain for a general system. Instead of creating a keyword database, creating an interface that a user can input keywords is decided.

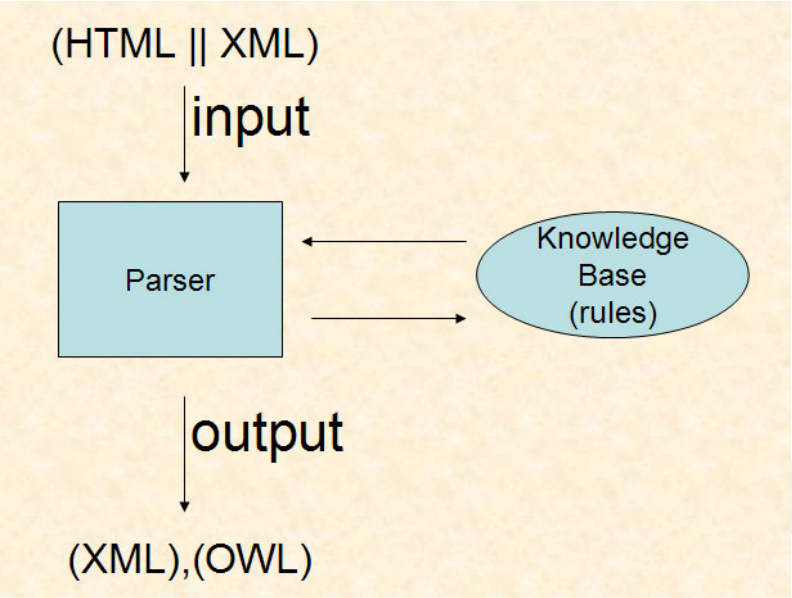

Figure 4.2 Knowledge Acquisition Design

After finding the information that is sought, inside the parsed data by the help of keyword search, the found information is split and the data properties of the information are combed out to fit the ontology's required format. Then the formatted information are both represented in OWL and XML formats as given in Figure 4.2. OWL format is for the ontology. XML format is for creating database if desired, but ontologies can also be represented in XML format.

4.3 Modules of The System

There are seven modules in the system as seen in Figure 4.3. First module is an interface module which is designed to reach the web page that includes the desired information. For this purpose, there is a textbox for the user where he can type the root page of a web site, and another textbox where the user can type keywords to search the hyperlinks to follow, included in the present web page. For example to the first textbox user will enter www.atilim.edu.tr and to the second textbox user will

enter the keyword “Bölüm”. Then software will list the departments that are linked from Atılım University’s home page. Similarly user can select the Computer Engineering’s web page and perform another keyword search for example with the word “Staff”. When the required page is reached the second module will start.

The second module is where the desired web page’s source code is parsed. After deciding on the web page that can hold the sought information is found in the first module, the parser parses the source code of the web page and splits it into single words.

The third module is in the user interface. This is the module where user enters keywords to perform keyword search to find the desired information. This keyword input is retrieved by a textbox. For example the user enters “Prof” to the textbox, then the software will return phrases that contains the keyword.

In the fourth module, after finding the desired information, the user enters names of the ontology’s classes and with a form, defines some rules about those classes, such as “this class can hold only a single word” or “this class holds an integer”. The name of the ontology is also decided in this module.

In the fifth module a rule-based algorithm is executed on the found information to split the information according to the ontology classes. These rules are selected by the user with a form.

In the sixth module, the split data is transferred and written into XML and OWL formats.

In the last module more than one OWL files, each containing an ontology class are merged and a relation between them is defined.

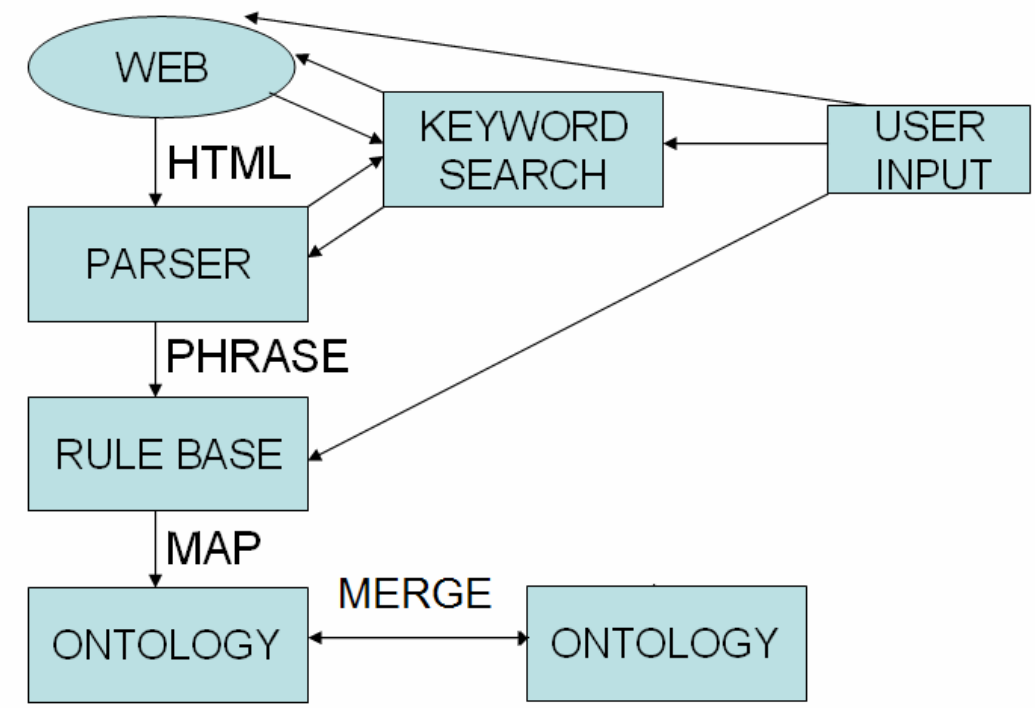

Figure 4.3 Flowchart of the Design

5. IMPLEMENTATION

5.1 Development of the System

The system was implemented with seven modules. These modules are described in the design chapter. For the implementation, a software that can work with the web pages dynamically was required. .NET Frame and ASP was selected for the implementation and outputs are given in OWL and XML formats. OWL stands for Ontology Web Language and has the most recent vocabulary for Semantic Web and XML is the root mark-up language. The implemented code with ASP is given in Appendix B.

In the very near future, software agents will probably be able to make searches in the web instead of users and will return the web pages found to their owners. Until date of completion of this study such an agent was not yet developed. This subject could be the topic of another thesis. Therefore it was decided to input the web sites' URL where the knowledge acquisition would be done, manually by the user.

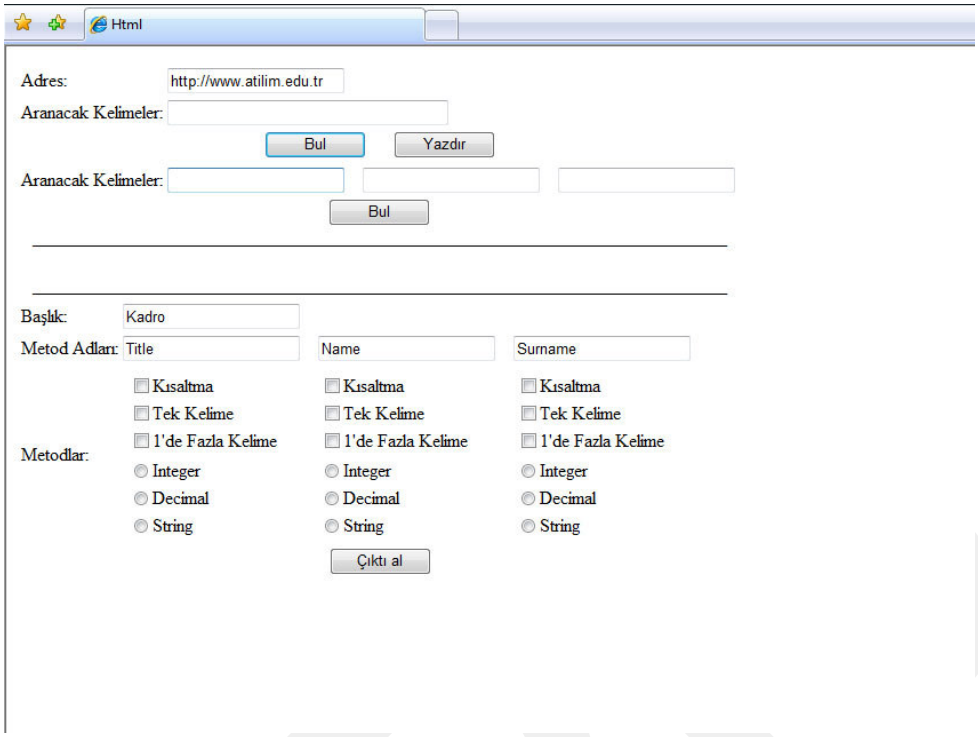

Figure 5.1 Web Site Input

The user inputs the URL of the root web page of the web site, from which the knowledge will be acquired as seen in Figure 5.1. ASP method webRequest calls the URL inputted to the text box by the user and webRequest.GetResponse method brings the source code of the web page. The code is converted to a huge string with the method StreamReader.

User interference is needed to find the data required which will be parsed and then eliminating the unnecessary data from the parsed data. This parsing and elimination phase is performed with the help of keyword searches. Those keywords should be inputted by the user as given in Figures 5.2, 5.3 and 5.4. The system works like user inputs a keyword and software returns web pages linked from the root page according to the keyword search, another keyword input by the user and software returns other web pages linked from previous web page. This mechanism continues until the user decides that required web page is reached. Because most of the knowledge that is going to be mapped in an ontology exist in the databases of web

sites, and when a query has been made against those databases, databases return the information in tables, lists and headings. The related information exists inside single tags in the parsed source code. While parsing the data the first idea was parsing the data according to spaces between words, but after realizing above mentioned fact instead of spaces, the data was parsed according to tag separators '>' and '<'.

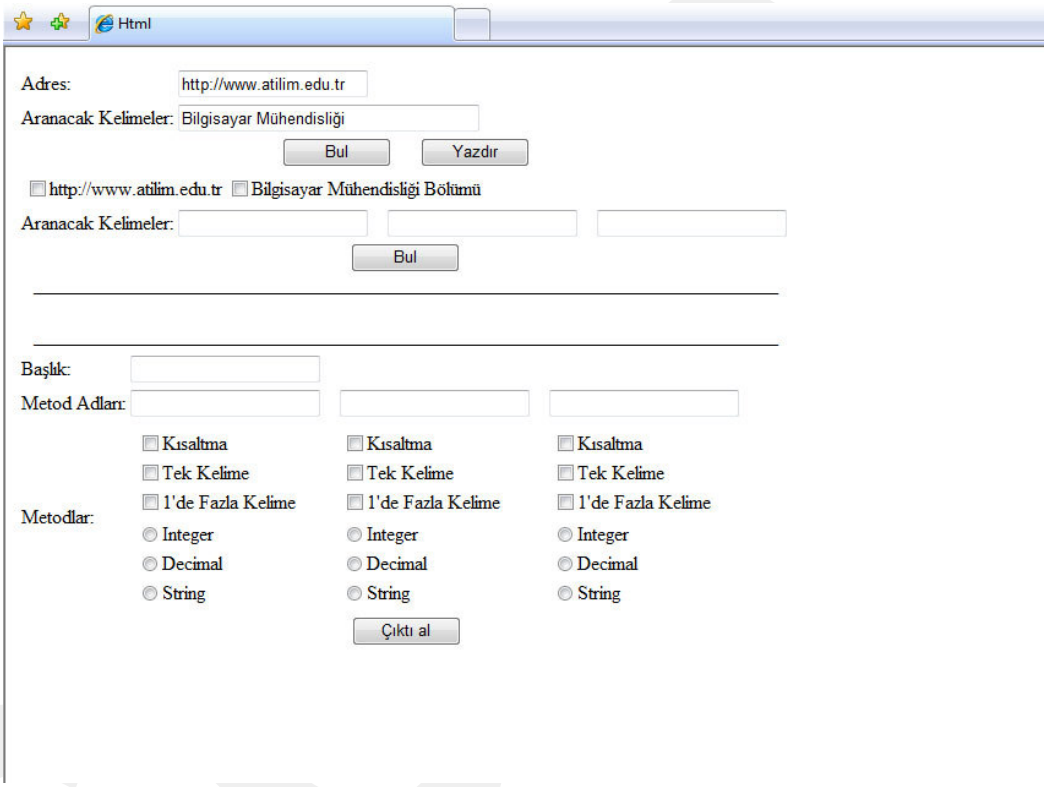

Figure 5.2 Keyword Input

The software separates the links in the split code according to "a href" which refers to hyperlinks in HTML codes. Then it eliminates the unnecessary links according to the keyword input. For these processes a rule-based system and string comparison algorithms are used.

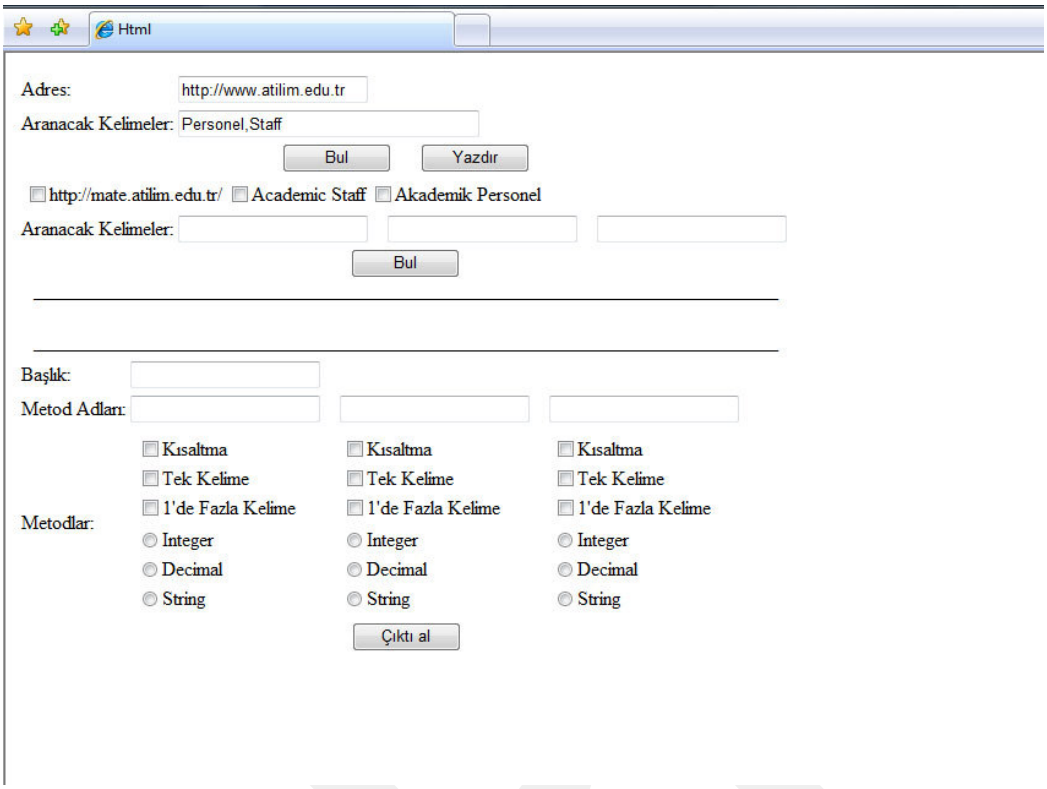

Figure 5.3 Keyword Input

The method toString().split('<') splits the data tag by tag.

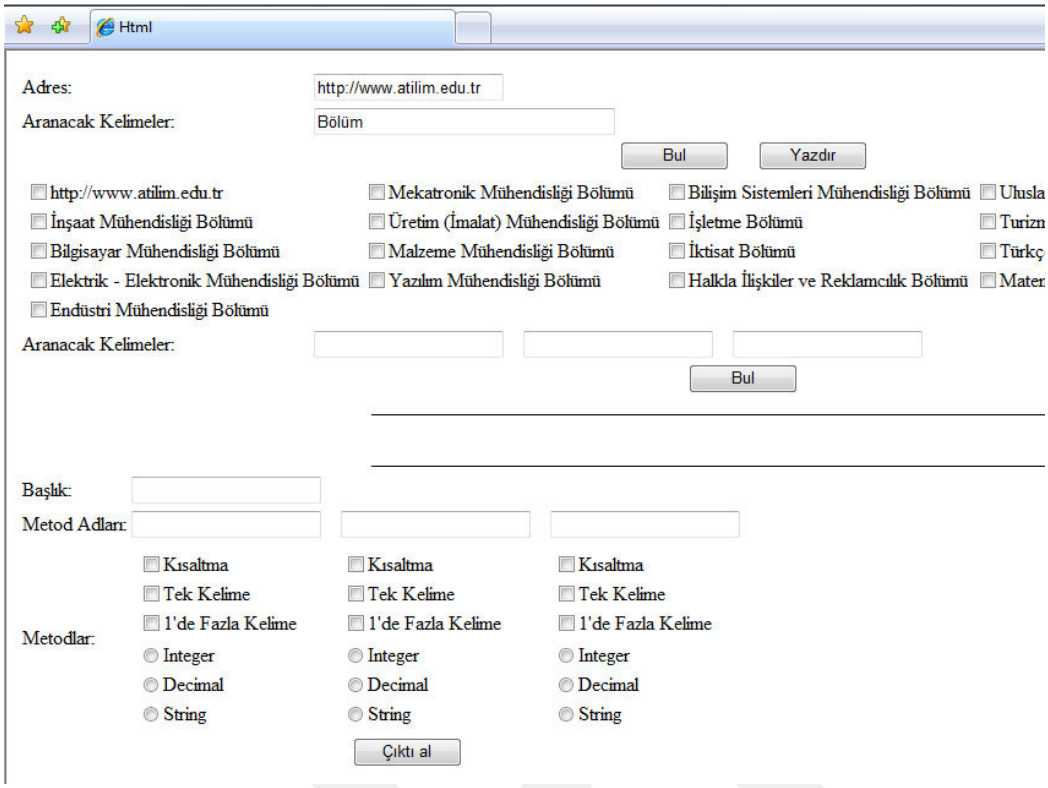

Figure 5.4 Listing of the Results

The software lists the links that suits the keyword search with checkboxes near them so the user could select the next link to follow as seen in Figure 5.5.

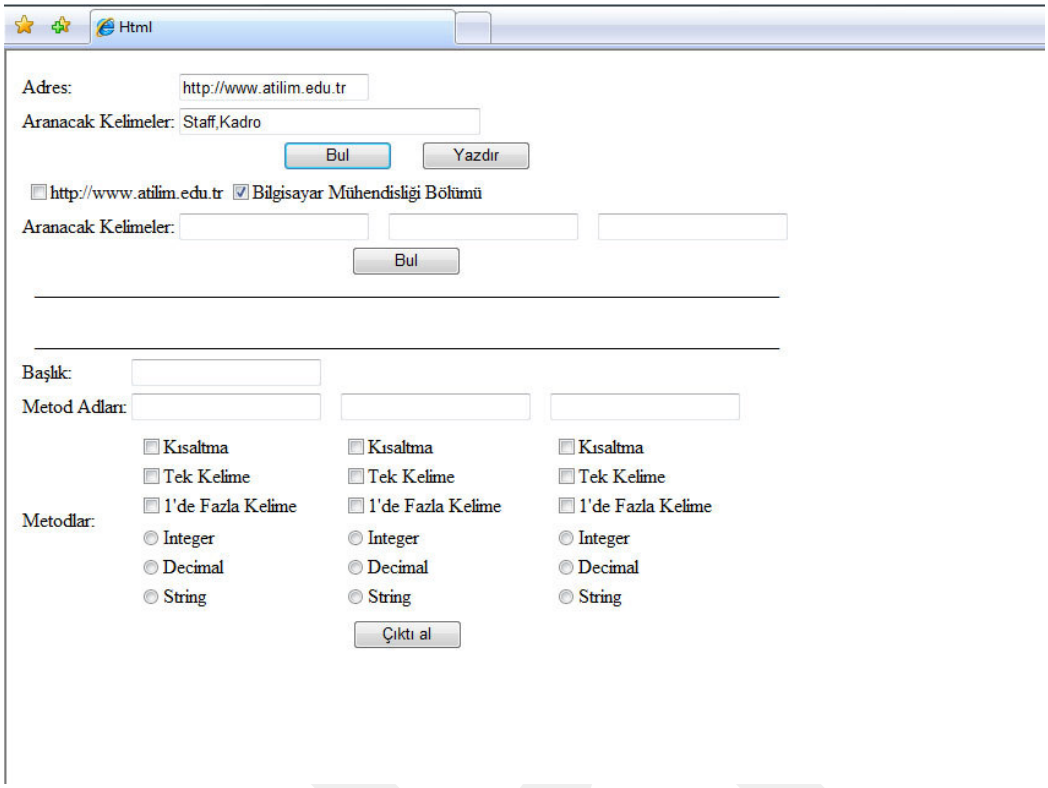

Figure 5.5 Link Selection

This keyword input, keyword search, link listing and link searching process continue until the user decides that the target page is reached. After finding the target page, the system parses its source code and splits it again tag by tag according to the tag separators with the method toString().Split('<').

After the data is parsed according to tag separators, a keyword search is required and the keywords for this search should also be inputted by the user. The system will find the matches and list them with their related data inside their tag.

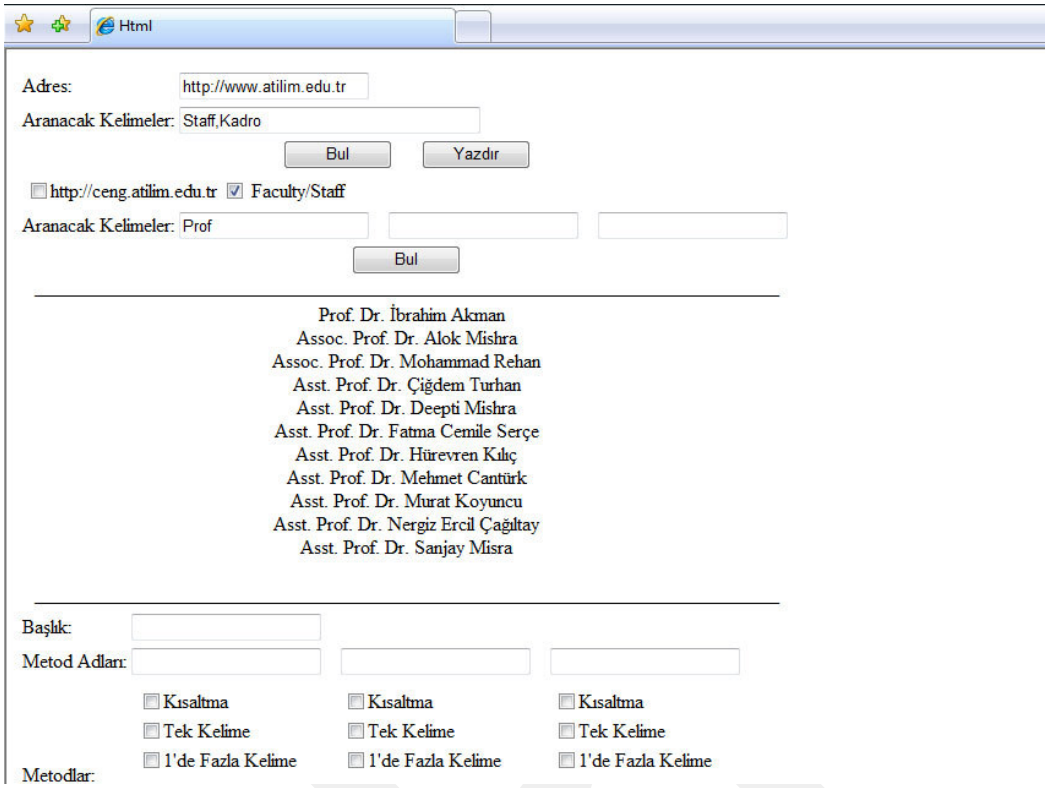

Figure 5.6 Staff is Listed

With string comparison algorithms this information and its related data within their tag are selected by keyword searches and they are listed to the user (Figure 5.6). Here with the help of and, or clauses and for statements user can input more than one keyword (Figure 5.7).

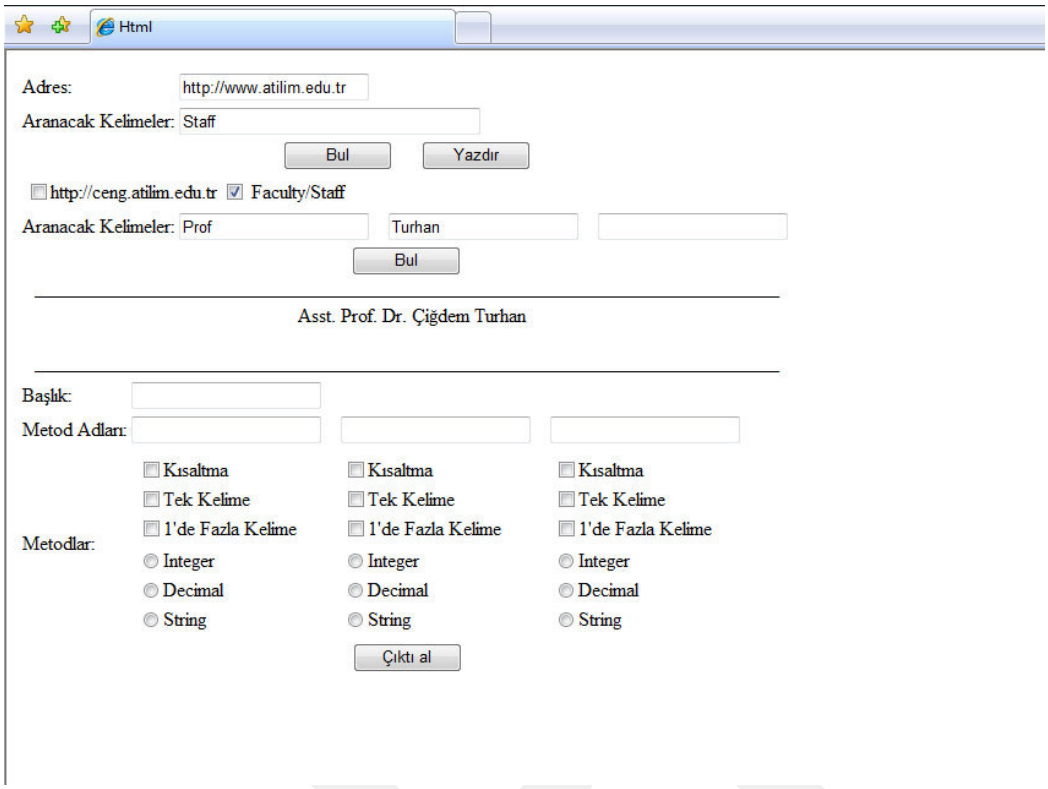

Figure 5.7 Searches with Two Keywords

Another user interference is required for mapping the knowledge into the ontology. For this task creating the ontology was also required. Because the software could not know the domain concepts it is not possible for it to create the ontology schema. For both ontology creation and knowledge mapping tasks, an HTML form was offered to the user. By the help of some textboxes, radio button and checkbox lists user help to the system to create the rule-base (Figure 5.9). Form allows three entities and each entity will be the data-type properties of that ontology class. By the radio buttons user can select the type of the entity. String, integer or decimal are the options there. Then the user selects one word and/or more than one word, abbreviation options for that entity by the checkbox list. According to those selections the software creates a dynamic rule-base with a huge nested if statement. In this if statement every possible combination of every entity is covered to classify and map

the related data into the ontology, and every selected information is tested in that rule-base. In the text boxes user inputs the name of the ontology class and the names of data-type properties of the class to create the ontology schema.

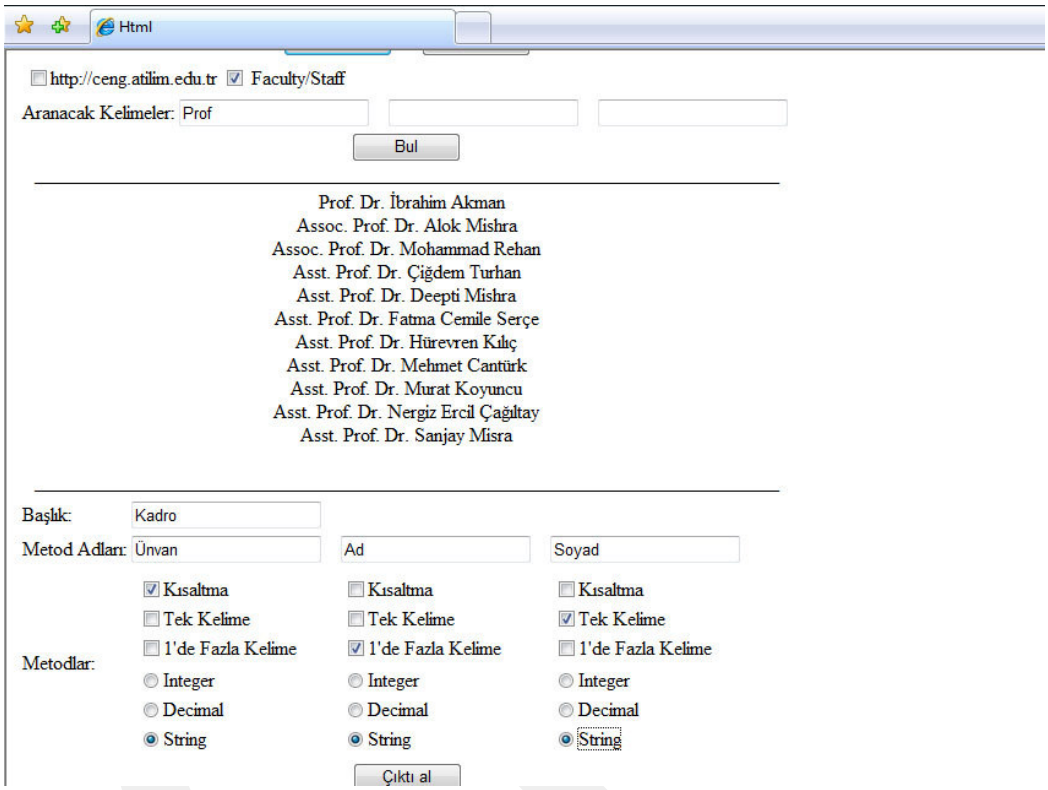

Figure 5.9 Building of the Rule Base

When the information is classified and mapped, and the ontology schema is created, user clicks to take output button and the class is represented both in XML and OWL formats. An OWL ouput is given in Figure 5.10

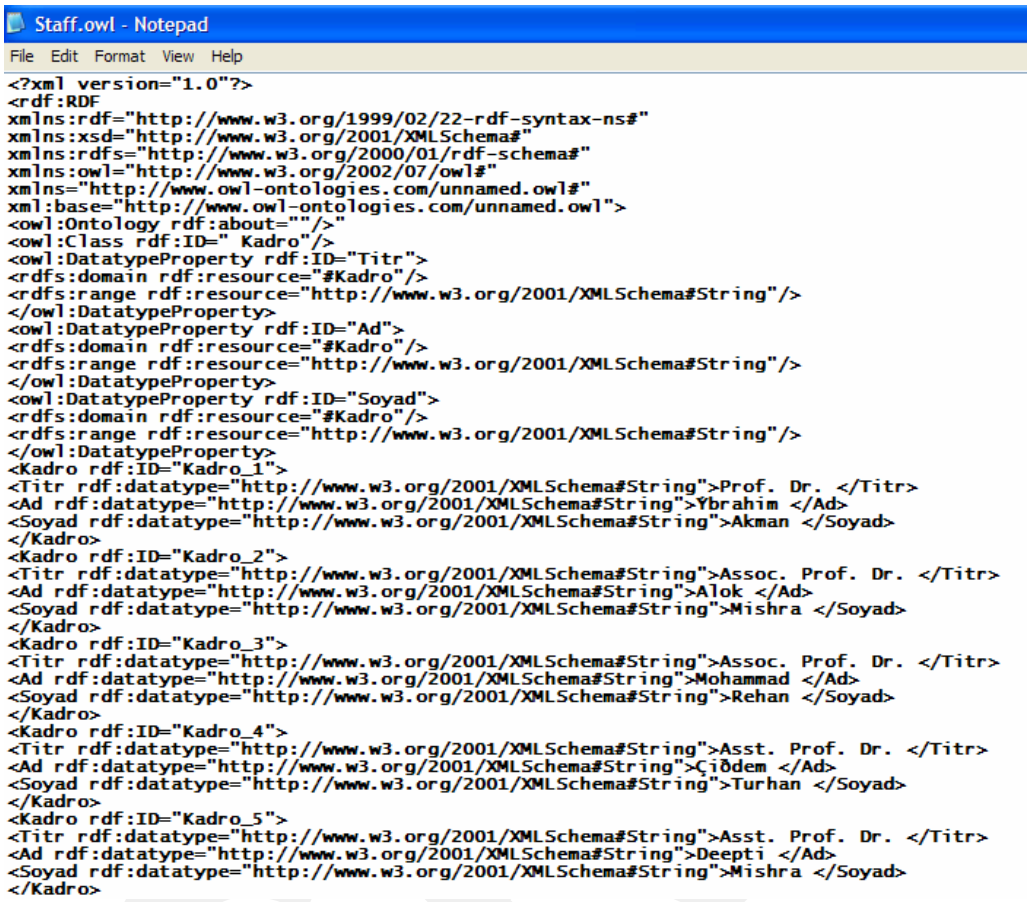

Figure 5.10 OWL Output

To merge more than one ontology classes and form an ontology the owl files that contain the classes are selected and a relation between them is defined by a textbox which will be the object property between those classes as seen in Figure 5.11. After merging the necessary classes the ontology building is finalized.

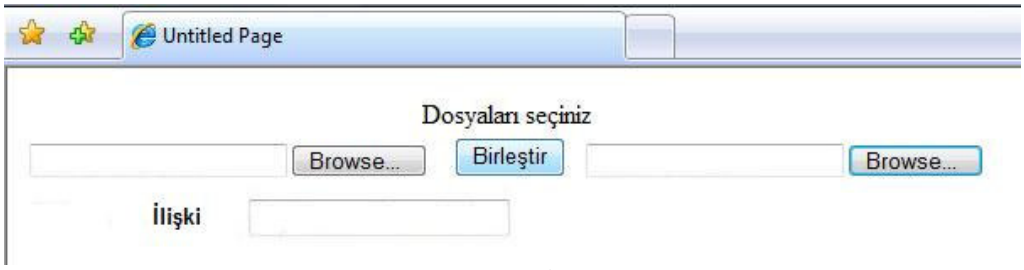

Figure 5.11 Merging the Ontologies

5.2 Evaluation of the System

The system is tested in the university web sites. Two web sites were used in these tests. Atılım University's web Site and Ankara University's web site. In Atılım University an ontology table about the staff of the computer engineering department; name, surname and the titles of the staff, and in Ankara University an ontology table about the staff of the geophysical engineering department; name, surname and titles of the staff were built by the system with semi-automatic knowledge acquisition. In these tests first the web page of the departments reached by the keyword "department". Then the web pages of computer engineering in one run and geophysical engineering in the other run were selected and the staff web pages of these departments were reached by another keyword search with the keyword "staff". While reaching the web pages that the knowledge will be acquired from, the system worked without any errors. Although it was known that "department" and "staff" are the correct keywords for these cases, they are also the first keywords that could come to a user's head for such cases. Therefore if the names given to the links make sense, in other words if the user can realize the correct keywords then the system is accessing the target web page without any problems.

In next steps, the system is working according to the tags. If the whole information inside a tag will be all mapped into the ontology again the system works without any problems. The knowledge would be easily acquired from, lists, tables and headings, because in most of the cases a list item or a table row or a heading contains neither less nor much but exact information for an ontology entity. In the test domains the title, surname and name of a staff were always inside a tag and there were no other information enveloped in that tag. If the database of the web-site built logically and if the ontology schema for that domain is going to be built with a similar logic then there should be no problem acquiring knowledge from that site.

For some cases the information could be given inside paragraph tags without any separator between them although this is not good web programming. In this case it is not possible to comb out the required information and system will fail.

6. CONCLUSION

The target of this thesis was to develop a semi-automatic knowledge acquisition system for the Semantic Web. The target of the semantic web technology is mapping knowledge about a domain into an ontology where software agents can also track it.

While building an ontology, the author first decides on the domain concepts usually by the help of domain experts. Then the knowledge that will be mapped into the ontology should be acquired. As stated in the Design and the Implementation chapters, the system developed for this thesis does not deal with domain description phase, but only deals with the knowledge acquisition phase. By the help of the ontology form given in the interface, the user is able to create the ontology for which support can be provided from the experts of the domain. The target of this thesis is the knowledge acquisition and representation of the acquired knowledge in an ontology. To achieve this target the first aim was creating a rule-based algorithm, which will eliminate the unnecessary information from a parsed data and classify the remaining necessary data. In the next steps representing that knowledge in OWL and XML formats, in other words representing the knowledge in an ontology was aimed.

As explained in the previous chapters, for a knowledge acquisition system to offer an advantage to a semantic web application, it should be at least semi-automatic or for most of the cases automatic.

Creating an automatic knowledge acquisition system for some cases is not possible within today's web concepts. For a knowledge acquisition system to be automatic the domain which the knowledge is going to be acquired from, should be described perfectly and the ontology schema should be built exactly for that domain.

Because there will be keyword searches and a very specific rule-base is needed, it would not be possible to adapt an automatic knowledge acquisition algorithm written for one domain, to another domain. There will certainly be lots of similarities in the logic of the systems but the algorithm will be totally different. That is why, while creating a general knowledge acquisition system which can work in any domain, a semi-automatic approach should be made instead of an automatic approach which is one of the suggestions reached by this thesis. But again although it was not considered as a concern of this thesis, with some future web standards that are unavailable today, it might be possible to build the domain automatically with a rule-base that can work in any domain. For this purpose the tacit information in the domain experts should also be implemented in the web and should be represented according to some standards. After deciding to build an acquisition system that can work in any domain, creating this system within a semi-automatic working concept was a necessity.

A semi-automatic system means there will be interferences from the users to the system while it is working. The system developed for this thesis requires three different interferences as given in the Implementation Chapter.

During the implementation phase, the main problem was the unavailability of a web standard for information representation. While coding the web pages, web programmers almost are never concerned with how readable and understandable their codes are. In almost every web page, coders give id's and names to most of the tags to give them styles or work with them dynamically. Instead of giving meaningless id's as they usually do, if they had given an id to the tag related with the information inside the tag, keyword search mechanism would be more efficient. While parsing the data the methods that server side scripter use usually make the parsing line by line.

Most of the web site source codes are not even suitable for that kind of parsing, because the coders did not care about the readability of their code.

As a result with user interferences at several points a semi-automatic knowledge acquisition system was created. The system is creating ontology classes with at most three attributes. If this number is desired to be increased, the number of rules increases drastically. There are several reasons for that. One of them is, for making the data represented in an ontology easily understandable by software, a lot of properties need to be added to each entity. Another reason is while combing out words, increasing the number of words and entities will increase exponentially the number of required rules to classify the data.

Creating relatively simple ontologies with a general semi-automatic knowledge acquisition system is a troublesome work for today's web yet for the web to have future with ontologies, improving automated knowledge acquisition is a must. For this improvement some information representation standards are required, to improve the coding of the source codes of web pages. Some useful annotations could be added to each tag about the information carried inside that tag. If the information representation task could be standardized, the difficulties against knowledge acquisition would be surpassed. When efficient knowledge acquisition systems for creating also complex ontologies will be developed, the Semantic Web will reach its full potential.

The Semantic Web is an important footstep for the World Wide Web. It allows the information to be shared and reused with making it understandable by software. Therefore, it can be concluded that by facilitating the access to the required documents and information the Semantic Web concept appears to be the future of the World Wide Web.

7. REFERENCES

[1] Liou, Y I. 1990. Knowledge Acquisition: Issues, Techniques and Methodology. *Proceedings of the 1990 ACM SIGBDP Conference on Trends and Directions in Expert Systems*. pp. 212 – 236.

[2] Forouzan, B A. 2004. *Data Communications and Networking*. Boston: McGraw-Hill

[3] *World Wide Web Consortium*. [Accessed January 2008]. Available from World Wide Web: <http://www.w3.org>

[4] Ding, Y, Fensel, D, Klein, M & Omelayenko, B. 2002. The Semantic Web: Yet Another Hip?. *Data and Knowledge Engineering*. pp. 205 - 227.

[5] *W3Schools Online Web Tutorials*. [Accessed January 2008]. Available from World Wide Web: < http://www.w3schools.com>

[6] Ray, E T. 2003. *Learning XML*. California: O'Reilly

[7] Warren, P. 2006. Knowledge Management and the Semantic Web: From Scenario to Technology. *IEEE Intelligent Systems*. **21** (1). pp. 53–59.

[8] December, J. 1996. Units of Analysis for Internet Communication. *Journal of Computer-Mediated Communication.* **1** (4), pp.0-0

[9] Hongbing, W, Zhexue, H J, Yuzhong, Q & Junyuan, X. 2004. Web Services: Problems and Future Directions. *Journal of Web Semantics.* **1** (3). pp. 309 – 320.

[10] Fensel, D, Bussler, C, Ding, Y, Kartseva, V, Klein, M, Korotkiy M, Omelayenko B & Siebes R. 2002. Semantic Web Application Areas. *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems*.

[11] Fensel, D, Hendler, J, Lieberman, H & Wahlster, W. 2001. Creating of Semantic Web. *http://citeseer.ist.psu.edu/481673.html.*

[12] Uschold, M. 2003. Where Are the Semantics in the Semantic Web?. *AI Magazine.* **24** (3). pp. 25 – 36.

[13] Ding, L, Zhou, L & Finin, T. 2003. Trust Based Knowledge Outsourcing for Semantic Web Agents. *Proceedings of the IEEE/WIC International Conference on Web Intellegence*. pp. 379

[14] Berners-Lee, T, Hendler, J & Lassila, O. 2001. The Semantic Web. *Scientific American*. pp. 35 – 43.

[15] Man, L, Xiao-Yong, D & Shan, W. 2005 Learning Ontology From Relational Database. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics.* **6**. pp. 3410 - 3415.

[16] Decker, S, Fensel, D, Harmelen, F van, Horrocks, I, Sergey, M, Klein, M & Broekstra, J. 2000. Knowledge Representation on the Web. *Description Logics*. pp. 89 - 97.

[17] Silva, N & Rocha, J. 2003. Semantic Web Complex Ontology Mapping. *IEEE/WIC International Conference on Web Intelligence*. pp. 82 – 88

[18] Zhang, J. 2007. Ontology and the Semantic Web. *Proceedings of the North American Symposium on Knowledge Organization*. pp.9 - 20,

[19] Ying, D & Engels, R. 2001. IR and AI: Using Co-Occurrence Theory to Generate Lightweight Ontologies. *Proceedings of the 12th International Workshop on Database and Expert Systems.* pp. 961 - 965.

[20] Colace, F, Santo, M De & Vento, M. 2004. An Automatic Algorithm for Building Ontologies from Data. *Proceedings of the International Conference on Information and Communication Technologies: From Theory to Applications.* pp. 117 - 118.

[21] Asunción, G P, Corcho, O. 2002. Ontology Languages for the Semantic Web. *IEEE Intelligent Systems*. **17** (1). pp. 54 - 60

[22] Chen, E & Wu, G. 2005. An Ontology Learning Method Enhanced by Frame Semantics. *Proceedings of the Seventh IEEE International Symposium on Multimedia*. pp. 374 - 382

[23] Baader, F, Horrocks, I & Sattler, U. 2005. Description Logics as Ontology Languages for the Semantic Web. *Lecture Notes in Computer Science*. pp. 228 - 248

[24] Noy, N F & McGuiness, D L. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*.

[25] Jacob, E K. 2003. Ontologies and the Semantic Web. *Bulletin of the American Society for Information Science and Technology*. **29** (4). pp. 19 - 22

[26] Natalya, F N, Sintek, M, Decker, S, Crubezy M, Fergerson R W & Musen M A. 2001. Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems*. **16** (2). pp. 60 - 70

[27] Ding, Y & Fensel, D. 2001. Ontology Library Systems: The Key to Successful Ontology Re-use. *Proceedings of the 1st Semantic Web Working Symposium*. pp. 93 - 112

[28] Elmasri, R & Navathe, S B. 2003. *Fundamentals of Database Systems*. California: Pearson/Addison Wesley.

[29] Gray, P M D, Kulkarni, K G & Paton, N W. 1992. *Object Oriented Databases A Semantic Data Model Approach*. New Jersey: Prentice Hall.

[30] Heflin, J & Hendler, J. 2000. Dynamic Ontologies on the Web. *Proceedings of the Seventeenth National Conference on Artificial Intelligence.* pp. 443 - 449.

[31] Hjelm, J. 2001. *Creating the Semantic Web with RDF*. New York: Wiley.

[32] Karvounarakis, G, Alexaki, S C, Vassilis, C, Plexousakis, D & Scholl, M. 2002. *RQL: A Declarative Query Language for RDF*. *Proceedings of the 11th international conference on World Wide Web.* pp. 592 – 603.

[33] Hu, H & Liu, D 2004. Learning OWL Ontologies from Free Texts. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics.* **2**. pp. 1233- 1237.

[34] *DAML.org*. [Accessed November 2007]. Available from World Wide Web: <http://www.daml.org>

[35] Bechhofer, S, Horrocks, I, Goble, C & Stevens, R. 2001. OilEd: a Reason-able Ontology Editor for the Semantic Web. *Lecture Notes in Computer Science*. **2174**. pp. 396.

[36] Karp, R, Chaudhri, V & Thomere, J. 1999. XOL: An XML-Based Ontology Exchange Language. <www.ai.sri.com/pkarp/xol>.

[37] *TopicMaps.Org Home Page*. [Accessed November 2007]. Available from World Wide Web: < http://www.topicmaps.org>

[38] *SHOE*. [Accessed November 2007]. Available from World Wide Web: <http://www.cs.umd.edu/projects/plus/SHOE>

[39] *Science Commons*. [Accessed December 2007]. Available from World Wide Web: <http://sciencecommons.org>

[40] *Friend of a Friend (FOAF) Project*. [Accessed December 2007]. Available from World Wide Web: < http://www.foaf-project.org>

[41] *sioc-project.org | Semantically Interlinked Online communities*. [Accessed June 2007]. Available from World Wide Web: <http://sioc-project.org>

[42] *SIMILE Project*. [Accessed June 2007]. Available from World Wide Web: <http://simile.mit.edu>

[43] *The National Center for Biomedical Ontology*. [Accessed December 2007]. Available from World Wide Web: <http://www.bioontology.org>

[44] Williams, K E & Kotnour, T. 1991. Automated Knowledge Acquisition for Second Generation Knowledge Base Systems: A Conceptual Analysis and Taxonomy. *Proceedings of the 27th Annual Meeting of the Institute of Management Sciences Myrtle Beach,SC, Report no: CONF-9110444—2.*

[45] Deng, P. 1990. Inducing Decision-Making Knowledge from Data Bases: An Approach to Automating Knowledge Acquisition. *Proceedings of the 1990 ACM SIGBDP Conference on Trends and Directions in Expert Systems.* pp. 189 – 211.

[46] Chen, L, Cox, S J, Tao, F, Shadbolt, N R, Puleston, C & Goble, C. 2004. Empowering Resource Provider to Build the Semantic Grid. *Proceedings of the International Conference on Web Intelligence*. pp. 271 – 277.

[47] Maedche, A & Staab, S. 2001. Ontology Learning For the Semantic Web. *IEEE Intelligent Systems, Special Issue on the Semantic Web.* pp. 72 – 79.

[48] Alani, H, Kim, S, Millard, D E, Weal, M J, Hall, W, Lewis, P H & Shadbolt, N R. 2003. Automatic Ontology-Based Knowledge Extraction from Web Documents. *IEEE Intelligent Systems*. **18** (1). pp. 14- 21

[49] Tanaka, M & Ishida, T. 2005. Ontology Extraction from Tables on the Web. *Proceedings of the 2005 Symposium on Applications and the Internet.* pp. 284 – 290.

8. APPENDIXES

Appendix A

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Person"/>
  <owl:Class rdf:ID="Adviser">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </owl:Class>
  <owl:Class rdf:ID="Student">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="advises">
    <rdfs:range rdf:resource="#Student"/>
    <rdfs:domain rdf:resource="#Adviser"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="Title">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Adviser"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Student_ID">
    <rdfs:domain rdf:resource="#Student"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Name">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Surname">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Person"/>
  </owl:DatatypeProperty>
  <Student rdf:ID="Individual_1">
    <Surname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></Surname>
    <Name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></Name>
    <Student_ID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >123456</Student_ID>
```

```
  </Student>
  <Adviser rdf:ID="Individual_4">
    <Name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Jane</Name>
    <Surname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Brown</Surname>
    <advises>
      <Student rdf:ID="Individual_2">
        <Surname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Smith</Surname>
        <Name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >John</Name>
        <Student_ID rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >123457</Student_ID>
      </Student>
    </advises>
    <Title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Prof. Dr.</Title>
  </Adviser>
  <Adviser rdf:ID="Individual_3">
    <Title rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Dr.</Title>
    <Surname rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></Surname>
    <Name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></Name>
    <advises rdf:resource="#Individual_1"/>
  </Adviser>
</rdf:RDF>
```

Appendix B

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
using System.Text;
using System.Net;
using System.Collections;
using System.Xml;

public partial class _Default : System.Web.UI.Page
{
    string[] adres;
    string[] adres2;
    string[] a1,a2;
    string uniler,text; string cumle;
    ArrayList key = new ArrayList();
    ArrayList cumleici = new ArrayList();
    string[] cumleici2;
    int sayac; string[] keys,keys1,keys2,keys3,aranacak;
    string t1, t2, t3, tt1, tt2, tt3;
    DataRow yRowadresler;
    public DataTable find = new DataTable();
    public DataTable ilkbulunan = new DataTable();
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    public void btfind_Click(object sender, EventArgs e)
    {
        keys = txtkey.Text.Split(',');
        DataTable adresler = new DataTable();
        adresler.Columns.Add("name");
        adresler.Columns.Add("link");
        if (adresler.Rows.Count == 0)
        {
            yRowadresler = adresler.NewRow();
            yRowadresler[0] = txtadres.Text;
            yRowadresler[1] = txtadres.Text;
            adresler.Rows.Add(yRowadresler);
            Session["seciliwww"] = txtadres.Text;
```

```
    Session["seciliad"]= txtadres.Text;
}
for (int i = 0; i < cb.Items.Count; i++)
{
    if (cb.Items[i].Selected == true)
    {
        Session["seciliwww"] = cb.Items[i].Value;
        Session["seciliad"] = cb.Items[i].Value;
    }
}
if (Session["seciliwww"] != null)
{
    adresler.Rows.Clear();
    yRowadresler = adresler.NewRow();
    yRowadresler[0] = Session["seciliad"].ToString();
    string adw3edfwerf = Session["seciliad"].ToString();
    yRowadresler[1] = Session["seciliwww"].ToString();
    string adw3edfwefe = Session["seciliwww"].ToString();
    adresler.Rows.Add(yRowadresler);

}
string[] uni = txtadres.Text.ToString().Split('.');
uniler = uni[1].ToString();

for (int i = 0; i < adresler.Rows.Count; i++)   /* Adresler */
{
    Session["adreskok"] = adresler.Rows[i][1].ToString();
    try
    {
        string sdasdsa = adresler.Rows[i][1].ToString();
        WebRequest wreq = WebRequest.Create(adresler.Rows[i][1].ToString());
        WebResponse wres = wreq.GetResponse();
        Encoding enc = Encoding.GetEncoding("iso-8859-9");
        StreamReader sr = new StreamReader(wres.GetResponseStream(), enc);
        Session["yazi"] = sr.ReadToEnd();
        wreq.GetResponse().Close();
        adres = Session["yazi"].ToString().Split('<');
    }
    catch (System.Net.WebException ex)
    {

    }
    for (int j = 0; j < adres.Length; j++)   /* Adresin HTML Kodu */
    {
        if (adres[j].ToString() != "" && adres[j].ToString().StartsWith("a href"))
        {
            if (adres[j + 1].ToString().StartsWith("font"))
            {
                a1 = adres[j + 2].ToString().Split('>');
                text = a1[a1.Length - 1].ToString();
```

```
}
else /* Buraya Font sonrası olayını ekle cnm */
{
   a1 = adres[j].ToString().Split('>');
   text = a1[a1.Length - 1].ToString();
}

if(a1[0].ToString().Contains("""))
{
   if (a1[0].IndexOf("""") == 7)
   {
      try
      {
         a2 = a1[0].ToString().Substring(8, a1[0].IndexOf("""", 8) - 8).Split("""");
      }
      catch
      {
      }
   }
   else if (a1[0].IndexOf("""") == 7)
   {
      try
      {
         a2 = a1[0].ToString().Substring(8, a1[0].IndexOf("""", 8) - 8).Split('#');
      }
      catch
      {
      }
   }
}
else
{
   try
   {
   a2 = a1[0].ToString().Substring(8, a1[0].IndexOf("""", 8) - 8).Split('#');
   }
   catch
   {
   }
   //a2 = a1[0].ToString().Split("""");
}

for (int m = 0; m < keys.Length; m++)
{
   if (text.Contains(keys[m].ToString()))
   {
      if (a1[1].ToString() != """" && (text != """") && a2.Length > 0)
      {

         if (a2[0].ToString().StartsWith(txtadres.Text))
```

```
{
    sayac = 0;

    for (int k = 0; k < adresler.Rows.Count; k++)
    {
        string asdasd = a2[0].ToString();
        string asdasdasdas = adresler.Rows[k][1].ToString();
        if (a2[0].ToString() == adresler.Rows[k][1].ToString())
        {
            sayac = 1;
            break;
        }
    }
    if (sayac == 0)
    {
        yRowadresler = adresler.NewRow();
        yRowadresler[0] = text;
        yRowadresler[1] = a2[0].ToString();
        adresler.Rows.Add(yRowadresler);
    }
}
else if (a2[0].ToString().StartsWith("/") || a2[0].ToString().StartsWith("../"))
{
    sayac = 0;
    for (int k = 0; k < adresler.Rows.Count; k++)
    {
        if (a2[0].ToString() == adresler.Rows[k][1].ToString())
        {
            sayac = 1;
            break;
        }
    }
    if (sayac == 0)
    {
        yRowadresler = adresler.NewRow();
        yRowadresler[0] = text;
        if (a2[0].ToString().StartsWith("../"))
        {
            yRowadresler[1] = Session["adreskok"].ToString() +
a2[0].ToString().Remove(0, 2);
        }
        else
        {
            yRowadresler[1] = Session["adreskok"].ToString() + a2[0].ToString();
        }
        adresler.Rows.Add(yRowadresler);
    }
}
else if (a2[0].ToString().Contains(uniler))
{
```

```
                                    sayac = 0;
                                    for (int k = 0; k < adresler.Rows.Count; k++)
                                    {
                                        if (a2[0].ToString() == adresler.Rows[k][1].ToString())
                                        {
                                            sayac = 1;
                                            break;
                                        }
                                    }
                                    if (sayac == 0)
                                    {
                                        yRowadresler = adresler.NewRow();
                                        yRowadresler[0] = text;
                                        yRowadresler[1] = a2[0].ToString();
                                        adresler.Rows.Add(yRowadresler);
                                    }
                                }
                                else if (!a2[0].ToString().StartsWith("/"))
                                {
                                    sayac = 0;
                                    for (int k = 0; k < adresler.Rows.Count; k++)
                                    {
                                        string asdasdasdas = adresler.Rows[k][1].ToString();
                                        string asdefwesfesf = Session["seciliwww"].ToString() + "/" + a2[0].ToString();
                                        if (Session["seciliwww"].ToString() + "/" + a2[0].ToString() ==
adresler.Rows[k][1].ToString())
                                        {
                                            sayac = 1;
                                            break;
                                        }
                                    }
                                    if (sayac == 0)
                                    {
                                        yRowadresler = adresler.NewRow();
                                        yRowadresler[0] = text;
                                        yRowadresler[1] = Session["seciliwww"].ToString() + "/" + a2[0].ToString();
                                        adresler.Rows.Add(yRowadresler);
                                    }
                                }



                            }
                        }
                    }
                }
                break;
            }
            cb.DataSource = adresler;
```

```
    cb.DataTextField = "name";
    cb.DataValueField = "link";
    cb.DataBind();

}
public void btkeybul_Click(object sender, EventArgs e)
{
    int kacarama = 0;
    string bulunan = "";
    string[] cumleici;
    if (txta1.Text != "")
    {
        keys1 = txta1.Text.Split(',');
        if (txta2.Text != "")
        {
            keys2 = txta2.Text.Split(',');
            if (txta3.Text != "")
            {
                keys3 = txta3.Text.Split(',');
                for (int i = 0; i < keys1.Length; i++)
                {
                    for (int j = 0; j < keys2.Length; j++)
                    {
                        for (int k = 0; k < keys3.Length; k++)
                        {
                            key.Add(keys1[i].ToString() + "-" + keys2[j].ToString() + "-" + keys3[k].ToString());
                        }
                    }
                }
                kacarama = 3;
            }
            else
            {
                for (int i = 0; i < keys1.Length; i++)
                {
                    for (int j = 0; j < keys2.Length; j++)
                    {
                        key.Add(keys1[i].ToString() + "-" + keys2[j].ToString());
                    }
                }
                kacarama = 2;
            }
        }
        else if (txta3.Text != "")
        {
            keys3 = txta3.Text.Split(',');
            for (int i = 0; i < keys1.Length; i++)
            {
                for (int j = 0; j < keys3.Length; j++)
                {
```

```
                    key.Add(keys1[i].ToString() + "-" + keys3[j].ToString());
                }
            }
            kacarama = 2;
        }
        else
        {
            for (int i = 0; i < keys1.Length; i++)
            {
                key.Add(keys1[i].ToString());
            }
            kacarama = 1;
        }
    }
    ilkbulunan.Columns.Add("D");
    find.Columns.Add(txtm1.Text);
    find.Columns.Add("Deger1");
    find.Columns.Add(txtm2.Text);
    find.Columns.Add("Deger2");
    find.Columns.Add(txtm3.Text);
    find.Columns.Add("Deger3");
    for (int i = 0; i < cb.Items.Count; i++)
    {
        if (cb.Items[i].Selected == true)
        {
            try
            {
                WebRequest wreq = WebRequest.Create(cb.Items[i].Value);
                WebResponse wres = wreq.GetResponse();
                Encoding enc = Encoding.GetEncoding("iso-8859-9");
                StreamReader sr = new StreamReader(wres.GetResponseStream(), enc);
                Session["yazi"] = sr.ReadToEnd();
                string asdasdwef = Session["yazi"].ToString();
                adres2 = Session["yazi"].ToString().Split('>');
                wreq.GetResponse().Close();
            }
            catch (System.Net.WebException ex)
            {

            }
            for (int j = 0; j < adres2.Length; j++)   /* Adresin HTML Kodu */
            {

                if (adres2[j].ToString() != "" && !(adres2[j].ToString().StartsWith("<")) &&
!(adres2[j].ToString().StartsWith("&")) && !(adres2[j].ToString().StartsWith("/")) )
                {
                    string asdasdasd = adres2[j].ToString();
                    if (adres2[j].ToString().Contains("<"))
                    {
                        string[] no = adres2[j].ToString().Split('<');
```

```
                    cumle = no[0].ToString();
                }
                else
                {
                    cumle = adres2[j].ToString();
                }
                cumleici = cumle.ToString().Split(' '); string cumleicikelime="";
                for (int w = 0; w < cumleici.Length; w++)
                {
                    if (cumleici[w].ToString() != "" && cumleici[w].ToString() != " ")
                    {
                        cumleicikelime += cumleici[w].ToString() + " ";
                    }
                }
                cumleici = cumleicikelime.ToString().Split(' ');
                /* Key arama */
                for (int m = 0; m < key.Count; m++)
                {
                    if (kacarama == 1)
                    {
                        if (cumle.ToString().Contains(key[m].ToString()))
                        {
                            for (int n = 0; n < cumleici.Length; n++)
                            {
                                if (cumleici[n].ToString() != "" && !(cumleici[n].ToString().StartsWith("(")) &&
!(cumleici[n].ToString().StartsWith(")")))
                                {
                                    if (n > 0)
                                    {
                                        if (!(cumleici[n - 1].ToString().StartsWith("(")))
                                        {
                                            bulunan += cumleici[n].ToString() + " ";
                                        }
                                    }
                                    else
                                    {
                                        bulunan += cumleici[n].ToString() + " ";
                                    }
                                }
                            }
                            yRowadresler = ilkbulunan.NewRow();
                            yRowadresler[0] = bulunan;
                            ilkbulunan.Rows.Add(yRowadresler);
                            bulunan = "";
                        }
                    }
                    else if(kacarama == 2)
                    {
                        string[] aranacak = key[m].ToString().Split('-');
```

```
                    if (cumle.ToString().Contains(aranacak[0].ToString()) &&
cumle.ToString().Contains(aranacak[1].ToString()))
                    {
                        for (int n = 0; n < cumleici.Length; n++)
                        {
                            if (cumleici[n].ToString() != "" && !(cumleici[n].ToString().StartsWith("\r")) &&
!(cumleici[n].ToString().StartsWith("(")) && !(cumleici[n].ToString().StartsWith(")")))
                            {
                                if (n > 0)
                                {
                                    if (!(cumleici[n - 1].ToString().StartsWith("(")))
                                    {
                                        bulunan += cumleici[n].ToString() + " ";
                                    }
                                }
                                else
                                {
                                    bulunan += cumleici[n].ToString() + " ";
                                }
                            }
                        }
                        yRowadresler = ilkbulunan.NewRow();
                        yRowadresler[0] = bulunan;
                        ilkbulunan.Rows.Add(yRowadresler);
                        bulunan = "";
                    }
                }
                else if (kacarama == 3)
                {
                    aranacak = key[m].ToString().Split('-');
                    if (cumle.ToString().Contains(aranacak[0].ToString()) &&
cumle.ToString().Contains(aranacak[1].ToString()) && cumle.ToString().Contains(aranacak[2].ToString()))
                    {
                        for (int n = 0; n < cumleici.Length; n++)
                        {
                            if (cumleici[n].ToString() != "" && !(cumleici[n].ToString().StartsWith("\r")) &&
!(cumleici[n].ToString().StartsWith("(")) && !(cumleici[n].ToString().StartsWith(")")))
                            {
                                if (n > 0)
                                {
                                    if (!(cumleici[n - 1].ToString().StartsWith("(")))
                                    {
                                        bulunan += cumleici[n].ToString() + " ";
                                    }
                                }
                                else
                                {
                                    bulunan += cumleici[n].ToString() + " ";
                                }
                            }
```

```
                        }
                        yRowadresler = ilkbulunan.NewRow();
                        yRowadresler[0] = bulunan;
                        ilkbulunan.Rows.Add(yRowadresler);
                        bulunan = "";
                    }
                }
            }
            /* Key Arama bitişi */
        }
      }
      string txt1 = "";
      string txt = "";
      for (int b = 0; b < ilkbulunan.Rows.Count; b++)
      {
        txt1 += ilkbulunan.Rows[b][0].ToString() + " <BR />";
        txt += ilkbulunan.Rows[b][0].ToString() + "-";
      }
        lbfind.Text = txt1;
        HiddenField1.Value = txt;

    }
  }
}

public void btbulyaz_Click(object sender, EventArgs e)
{
  find.Columns.Add(txtm1.Text);
  find.Columns.Add("Deger1");
  find.Columns.Add(txtm2.Text);
  find.Columns.Add("Deger2");
  find.Columns.Add(txtm3.Text);
  find.Columns.Add("Deger3");
  ArrayList cb1 = new ArrayList();
  ArrayList cb2 = new ArrayList();
  ArrayList cb3 = new ArrayList();
  tt1 = rbm1.SelectedItem.Text;
  tt2 = rbm2.SelectedItem.Text;
  tt3 = rbm3.SelectedItem.Text;
  for (int i = 0; i < cbm1.Items.Count; i++)
  {
    if (cbm1.Items[i].Selected == true)
    {
      cb1.Add(cbm1.Items[i].Value);
    }
  }
  for (int i = 0; i < cbm2.Items.Count; i++)
  {
    if (cbm2.Items[i].Selected == true)
    {
```

```
            cb2.Add(cbm2.Items[i].Value);
        }
    }
    for (int i = 0; i < cbm3.Items.Count; i++)
    {
        if (cbm3.Items[i].Selected == true)
        {
            cb3.Add(cbm3.Items[i].Value);
        }
    }
        if (rbm1.SelectedValue == "1")   /* Metod1 Integer*/
        {
            if (rbm2.SelectedValue == "1")   /* Metod2 Integer*/
            {
                if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
                {
                    string[] hepsi = HiddenField1.Value.Split('-');
                    for (int i = 0; i < hepsi.Length; i++)
                    {
                        cumleici.Clear();
                        cumleici2 = hepsi[i].ToString().Split(' ');
                        for (int j = 0; j < cumleici2.Length; j++)
                        {
                            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                            {
                                cumleici.Add(cumleici2[j].ToString());
                            }
                        }
                            for (int n = 0; n < cumleici.Count; n++)
                            {
                                try
                                {
                                    if (Convert.ToInt32(cumleici[n].ToString()) / 1 ==
Convert.ToInt32(cumleici[n].ToString()) && Convert.ToInt32(cumleici[n+1].ToString()) / 1 ==
Convert.ToInt32(cumleici[n+1].ToString()) && Convert.ToInt32(cumleici[n+2].ToString()) / 1 ==
Convert.ToInt32(cumleici[n+2].ToString()))
                                    {
                                        t1 = cumleici[n].ToString();
                                        t2 = cumleici[n+1].ToString();
                                        t3 = cumleici[n+2].ToString();
                                        break;
                                    }
                                }
                                catch
                                {
                                }

                            }
                            yRowadresler = find.NewRow();
                            yRowadresler[0] = t1;
```

```
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
        }
        else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n+1].ToString().Contains(",")) ) && Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
                    }
                    catch
                    {
                    }

                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
```

```
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
            else  /* Metod3 String*/
            {
                string[] hepsi = HiddenField1.Value.Split('-');
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        try
                        {
                            if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (!cumleici[n + 2].ToString().Contains(".")))
                            {
                                t1 = cumleici[n].ToString();
                                t2 = cumleici[n + 1].ToString();
                                t3 = cumleici[n + 2].ToString();
                                break;
                            }
                        }
                        catch
                        {
                        }

                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
```

```
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
        }
        else if (rbm2.SelectedValue == "2")  /* Metod2 Decimal*/
        {
            if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
            {
                string[] hepsi = HiddenField1.Value.Split('-');
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        try
                        {
                            if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString()))
                            {
                                t1 = cumleici[n].ToString();
                                t2 = cumleici[n + 1].ToString();
                                t3 = cumleici[n + 2].ToString();
                                break;
                            }
                        }
                        catch
                        {
                        }

                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
```

```
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
            else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
            {
                string[] hepsi = HiddenField1.Value.Split('-');
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        try
                        {
                            if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains","))))
                            {
                                t1 = cumleici[n].ToString();
                                t2 = cumleici[n + 1].ToString();
                                t3 = cumleici[n + 2].ToString();
                                break;
                            }
                        }
                        catch
                        {
                        }

                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;
```

```
            }
        }
        else  /* Metod3 String*/
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (!cumleici[n + 2].ToString().Contains(".")))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
                    }
                    catch
                    {
                    }

                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
        }
```

```
                }
                else  /* Metod2 String*/
                {
                    if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
                    {
                        string[] hepsi = HiddenField1.Value.Split('-');
                        for (int i = 0; i < hepsi.Length; i++)
                        {
                            cumleici.Clear();
                            cumleici2 = hepsi[i].ToString().Split(' ');
                            for (int j = 0; j < cumleici2.Length; j++)
                            {
                                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                                {
                                    cumleici.Add(cumleici2[j].ToString());
                                }
                            }
                            for (int n = 0; n < cumleici.Count; n++)
                            {
                                try
                                {
                                    if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (!cumleici[n + 1].ToString().Contains(".")) && Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString()))
                                    {
                                        t1 = cumleici[n].ToString();
                                        t2 = cumleici[n + 1].ToString();
                                        t3 = cumleici[n + 2].ToString();
                                        break;
                                    }
                                }
                                catch
                                {
                                }

                            }
                            yRowadresler = find.NewRow();
                            yRowadresler[0] = t1;
                            yRowadresler[1] = tt1;
                            yRowadresler[2] = t2;
                            yRowadresler[3] = tt2;
                            yRowadresler[4] = t3;
                            yRowadresler[5] = tt3;
                            find.Rows.Add(yRowadresler);
                            t1 = ""; t2 = ""; t3 = "";
                            break;

                        }
                    }
                    else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
```

```
{
   string[] hepsi = HiddenField1.Value.Split('-');
   for (int i = 0; i < hepsi.Length; i++)
   {
      cumleici.Clear();
      cumleici2 = hepsi[i].ToString().Split(' ');
      for (int j = 0; j < cumleici2.Length; j++)
      {
         if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
         {
            cumleici.Add(cumleici2[j].ToString());
         }
      }
      for (int n = 0; n < cumleici.Count; n++)
      {
         try
         {
            if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (!cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
            {
               t1 = cumleici[n].ToString();
               t2 = cumleici[n + 1].ToString();
               t3 = cumleici[n + 2].ToString();
               break;
            }
         }
         catch
         {
         }

      }
      yRowadresler = find.NewRow();
      yRowadresler[0] = t1;
      yRowadresler[1] = tt1;
      yRowadresler[2] = t2;
      yRowadresler[3] = tt2;
      yRowadresler[4] = t3;
      yRowadresler[5] = tt3;
      find.Rows.Add(yRowadresler);
      t1 = ""; t2 = ""; t3 = "";
      break;

   }
}
else  /* Metod3 String*/
{
   if (cb2.Contains("1") || cb3.Contains("1"))//m2 ve/veya m3 kısaltma var
   {
      if (cb2.Contains("1"))//m2
```

```
{
    string[] hepsi = HiddenField1.Value.Split('-');
    for (int i = 0; i < hepsi.Length; i++)
    {
        cumleici.Clear();
        cumleici2 = hepsi[i].ToString().Split(' ');
        for (int j = 0; j < cumleici2.Length; j++)
        {
            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
            {
                cumleici.Add(cumleici2[j].ToString());
            }
        }
        for (int n = 0; n < cumleici.Count; n++)
        {
            try
            {
                if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n + 1].ToString().Contains(".")) && (!cumleici[n + 2].ToString().Contains(".")))
                {
                    t1 = cumleici[n].ToString();
                    t2 = cumleici[n + 1].ToString();
                    t3 = cumleici[n + 2].ToString();
                    break;
                }
            }
            catch
            {
            }

        }
        yRowadresler = find.NewRow();
        yRowadresler[0] = t1;
        yRowadresler[1] = tt1;
        yRowadresler[2] = t2;
        yRowadresler[3] = tt2;
        yRowadresler[4] = t3;
        yRowadresler[5] = tt3;
        find.Rows.Add(yRowadresler);
        t1 = ""; t2 = ""; t3 = "";
        break;

    }
}
else if (cb3.Contains("1"))//m3
{
    string[] hepsi = HiddenField1.Value.Split('-');
    for (int i = 0; i < hepsi.Length; i++)
    {
```

```
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if (Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (!cumleici[n + 1].ToString().Contains(".")) && (cumleici[n + 2].ToString().Contains(".")))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
                    }
                    catch
                    {
                    }

                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
        }
        else
        {
            if (cb2.Contains("2"))//m2 tek kelime
            {
            }
            else if (cb3.Contains("2"))//m3 tek kelime
            {
            }
            else
```

```
                        {
                            //if()//keyword m2 ye girilmisse
                            //{
                            //}
                            //else if()//keyword m3
                            //{
                            //}
                            //else
                            //{
                            //    if()//2 kelime kaldıysa
                            //    {
                            //    }
                            //    else
                            //    {
                            //        //son 2 kelime m3 ye gerisi m2 e
                            //    }
                            //}
                        }
                    }
                }
            }
        }
        else if (rbm1.SelectedValue == "2")  /* Metod1 Decimal*/
        {
            if (rbm2.SelectedValue == "1")   /* Metod2 Integer*/
            {
                if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
                {
                    string[] hepsi = HiddenField1.Value.Split('-');
                    for (int i = 0; i < hepsi.Length; i++)
                    {
                        cumleici.Clear();
                        cumleici2 = hepsi[i].ToString().Split(' ');
                        for (int j = 0; j < cumleici2.Length; j++)
                        {
                            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                            {
                                cumleici.Add(cumleici2[j].ToString());
                            }
                        }
                        for (int n = 0; n < cumleici.Count; n++)
                        {
                            try
                            {
                                if ((Convert.ToInt32(cumleici[n].ToString()) / 1 ==
Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) &&
(Convert.ToInt32(cumleici[n+1].ToString()) / 1 == Convert.ToInt32(cumleici[n+1].ToString())) &&
(Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString())))
                                {
```

```
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n+1].ToString()) / 1 == Convert.ToInt32(cumleici[n+1].ToString())) && (Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
```

```
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else  /* Metod3 String*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString())) && (!cumleici[n + 2].ToString().Contains(".")))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
```

```
                    catch
                    {
                    }

                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
        }
    }
    else if (rbm2.SelectedValue == "2")  /* Metod2 Decimal*/
    {
        if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n+1].ToString()) / 1 == Convert.ToInt32(cumleici[n+1].ToString()) && (cumleici[n+1].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString())))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
```

```
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains","))))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
```

```
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else  /* Metod3 String*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (!cumleici[n + 2].ToString().Contains(".")))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }
```

```
                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
        }
    }
    else  /* Metod2 String*/
    {
        if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (!cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString())))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
                    }
                    catch
                    {
                    }
```

```
            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (!cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains","))))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
```

```
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
        }
        else  /* Metod3 String*/
        {
            if (cb2.Contains("1") || cb3.Contains("1"))//m2 ve/veya m3 kısaltma var
            {
                if (cb2.Contains("1"))//m2
                {
                    string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (cumleici[n + 1].ToString().Contains(".")) && (!cumleici[n + 2].ToString().Contains(".")))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
                    }
                    catch
                    {
                    }

                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
```

```
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
                }
                else if (cb3.Contains("1"))//m3
                {
                    string[] hepsi = HiddenField1.Value.Split('-');
                    for (int i = 0; i < hepsi.Length; i++)
                    {
                        cumleici.Clear();
                        cumleici2 = hepsi[i].ToString().Split(' ');
                        for (int j = 0; j < cumleici2.Length; j++)
                        {
                            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                            {
                                cumleici.Add(cumleici2[j].ToString());
                            }
                        }
                        for (int n = 0; n < cumleici.Count; n++)
                        {
                            try
                            {
                                if ((Convert.ToInt32(cumleici[n].ToString()) / 1 == Convert.ToInt32(cumleici[n].ToString()) && (cumleici[n].ToString().Contains(","))) && (!cumleici[n + 1].ToString().Contains(".")) && (cumleici[n + 2].ToString().Contains(".")))
                                {
                                    t1 = cumleici[n].ToString();
                                    t2 = cumleici[n + 1].ToString();
                                    t3 = cumleici[n + 2].ToString();
                                    break;
                                }
                            }
                            catch
                            {
                            }

                        }
                        yRowadresler = find.NewRow();
                        yRowadresler[0] = t1;
                        yRowadresler[1] = tt1;
                        yRowadresler[2] = t2;
                        yRowadresler[3] = tt2;
                        yRowadresler[4] = t3;
```

```
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
            else
            {
                if (cb2.Contains("2")) //m2 tek kelime
                {
                }
                else if (cb3.Contains("2"))//m3 tek kelime
                {
                }
                else
                {
                    //if()//keyword m2 ye girilmisse
                    //{
                    //}
                    //else if()//keyword m3
                    //{
                    //}
                    //else
                    //{
                    //    if()//2 kelime kaldıysa
                    //    {
                    //    }
                    //    else
                    //    {
                    //        //son 2 kelime m3 ye gerisi m2 e
                    //    }
                    //}
                }
            }
        }
    }
}
else  /* Metod1 String*/
{
    if (rbm2.SelectedValue == "1")   /* Metod2 Integer*/
    {
        if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
```

```
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        try
                        {
                            if ((!cumleici[n].ToString().Contains(".")) &&
(Convert.ToInt32(cumleici[n+1].ToString()) / 1 == Convert.ToInt32(cumleici[n+1].ToString())) &&
(Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString())))
                            {
                                t1 = cumleici[n].ToString();
                                t2 = cumleici[n + 1].ToString();
                                t3 = cumleici[n + 2].ToString();
                                break;
                            }
                        }
                        catch
                        {
                        }

                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
            else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
            {
                string[] hepsi = HiddenField1.Value.Split('-');
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
```

```
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((!cumleici[n].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString())) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else  /* Metod3 String*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
```

```
                    {
                        try
                        {
                            if ((!cumleici[n].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString())) && (!cumleici[n+2].ToString().Contains(".")))
                            {
                                t1 = cumleici[n].ToString();
                                t2 = cumleici[n + 1].ToString();
                                t3 = cumleici[n + 2].ToString();
                                break;
                            }
                        }
                        catch
                        {
                        }

                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
        }
        else if (rbm2.SelectedValue == "2")  /* Metod2 Decimal*/
        {
            if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
            {
                string[] hepsi = HiddenField1.Value.Split('-');
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        try
```

```
                {
                    if ((!cumleici[n].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (Convert.ToInt32(cumleici[n+2].ToString()) / 1 == Convert.ToInt32(cumleici[n+2].ToString())))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((!cumleici[n].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) &&
```

```
(Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
                                    {
                                        t1 = cumleici[n].ToString();
                                        t2 = cumleici[n + 1].ToString();
                                        t3 = cumleici[n + 2].ToString();
                                        break;
                                    }
                                }
                                catch
                                {
                                }

                            }
                            yRowadresler = find.NewRow();
                            yRowadresler[0] = t1;
                            yRowadresler[1] = tt1;
                            yRowadresler[2] = t2;
                            yRowadresler[3] = tt2;
                            yRowadresler[4] = t3;
                            yRowadresler[5] = tt3;
                            find.Rows.Add(yRowadresler);
                            t1 = ""; t2 = ""; t3 = "";
                            break;

                        }
                    }
                    else  /* Metod3 String*/
                    {
                        string[] hepsi = HiddenField1.Value.Split('-');
                        for (int i = 0; i < hepsi.Length; i++)
                        {
                            cumleici.Clear();
                            cumleici2 = hepsi[i].ToString().Split(' ');
                            for (int j = 0; j < cumleici2.Length; j++)
                            {
                                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                                {
                                    cumleici.Add(cumleici2[j].ToString());
                                }
                            }
                            for (int n = 0; n < cumleici.Count; n++)
                            {
                                try
                                {
                                    if ((!cumleici[n].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 1].ToString()) / 1 == Convert.ToInt32(cumleici[n + 1].ToString()) && (cumleici[n + 1].ToString().Contains(","))) && (!cumleici[n+2].ToString().Contains(".")))
                                    {
                                        t1 = cumleici[n].ToString();
```

```
                    t2 = cumleici[n + 1].ToString();
                    t3 = cumleici[n + 2].ToString();
                    break;
                }
            }
            catch
            {
            }

        }
        yRowadresler = find.NewRow();
        yRowadresler[0] = t1;
        yRowadresler[1] = tt1;
        yRowadresler[2] = t2;
        yRowadresler[3] = tt2;
        yRowadresler[4] = t3;
        yRowadresler[5] = tt3;
        find.Rows.Add(yRowadresler);
        t1 = ""; t2 = ""; t3 = "";
        break;

    }
  }
}
else  /* Metod2 String*/
{
  if (rbm3.SelectedValue == "1")   /* Metod3 Integer*/
  {
    if (cb1.Contains("1") || cb2.Contains("1"))//m1 ve/veya m2 kısaltma var
    {
      if (cb1.Contains("1"))//m1
      {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
          cumleici.Clear();
          cumleici2 = hepsi[i].ToString().Split(' ');
          for (int j = 0; j < cumleici2.Length; j++)
          {
            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
            {
              cumleici.Add(cumleici2[j].ToString());
            }
          }
          for (int n = 0; n < cumleici.Count; n++)
          {
            try
            {
```

```
                    if ((cumleici[n].ToString().Contains(".")) &&
(!cumleici[n+1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n+2].ToString()) / 1 ==
Convert.ToInt32(cumleici[n+2].ToString())))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else if (cb2.Contains("1"))//m2
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if (!(cumleici[n].ToString().Contains(".")) && (cumleici[n +
1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n +
2].ToString())))
                    {
```

```
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((cumleici[n].ToString().Contains(".")) && (cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString())))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
```

```
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
        if (cb1.Contains("2")) //m1 tek kelime
        {
        }
        else if (cb2.Contains("2"))//m2 tek kelime
        {
        }
        else
        {
            //if()//keyword m1 ye girilmisse
            //{
            //}
            //else if()//keyword m2
            //{
            //}
            //else
            //{
            //    if()//2 kelime kaldıysa
            //    {
            //    }
            //    else
            //    {
            //        //son 2 kelime m2 ye gerisi m1 e
            //    }
            //}
        }
      }
   }
}
else if (rbm3.SelectedValue == "2")  /* Metod3 Decimal*/
{
   if (cb1.Contains("1") || cb2.Contains("1"))//m1 ve/veya m2 kısaltma var
```

```
{
    if (cb1.Contains("1"))//m1
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                try
                {
                    if ((cumleici[n].ToString().Contains(".")) && (!cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
                    {
                        t1 = cumleici[n].ToString();
                        t2 = cumleici[n + 1].ToString();
                        t3 = cumleici[n + 2].ToString();
                        break;
                    }
                }
                catch
                {
                }

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;

        }
    }
    else if (cb2.Contains("1"))//m2
    {
        string[] hepsi = HiddenField1.Value.Split('-');
```

```
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        try
                        {
                            if (!(cumleici[n].ToString().Contains(".")) && (cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
                            {
                                t1 = cumleici[n].ToString();
                                t2 = cumleici[n + 1].ToString();
                                t3 = cumleici[n + 2].ToString();
                                break;
                            }
                        }
                        catch
                        {
                        }

                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
            else
            {
                string[] hepsi = HiddenField1.Value.Split('-');
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
```

```
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    try
                    {
                        if (!(cumleici[n].ToString().Contains(".")) && (!cumleici[n + 1].ToString().Contains(".")) && (Convert.ToInt32(cumleici[n + 2].ToString()) / 1 == Convert.ToInt32(cumleici[n + 2].ToString()) && (cumleici[n + 2].ToString().Contains(","))))
                        {
                            t1 = cumleici[n].ToString();
                            t2 = cumleici[n + 1].ToString();
                            t3 = cumleici[n + 2].ToString();
                            break;
                        }
                    }
                    catch
                    {
                    }

                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }
            if (cb1.Contains("2"))//m1 tek kelime
            {
            }
            else if (cb2.Contains("2"))//m2 tek kelime
            {
            }
            else
            {
                //if()//keyword m1 ye girilmisse
                //{
                //}
                //else if()//keyword m2
```

```
                        //{
                        //}
                        //else
                        //{
                        //    if()//2 kelime kaldıysa
                        //    {
                        //    }
                        //    else
                        //    {
                        //        //son 2 kelime m2 ye gerisi m1 e
                        //    }
                        //}
                    }
                }
            }
            else  /* Metod3 String*/
            {
                if (cb1.Contains("1") || cb2.Contains("1") || cb3.Contains("1"))//m1 ve/veya m2 ve/veya m3
kısaltma
                {
                    if (cb1.Contains("1"))//m1 kısaltma
                    {
                        if (cb2.Contains("1"))//m2 kısaltma
                        {
                            if (cb3.Contains("1"))//m3 kısaltma
                            {
                                if (cb1.Contains("2"))//m1 tek kelime
                                {
                                    if (cb2.Contains("2"))//m2 tek kelime
                                    {
                                        string[] hepsi = HiddenField1.Value.Split('-');
                                        for (int i = 0; i < hepsi.Length; i++)
                                        {
                                            cumleici.Clear();
                                            cumleici2 = hepsi[i].ToString().Split(' ');
                                            for (int j = 0; j < cumleici2.Length; j++)
                                            {
                                                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                                                {
                                                    cumleici.Add(cumleici2[j].ToString());
                                                }
                                            }
                                            if                    (cumleici[0].ToString().EndsWith(".")                    &&
cumleici[cumleici.Count].ToString().EndsWith("."))
                                            {
                                                for (int n = 2; n < cumleici.Count; n++)
                                                {
                                                    t1 = cumleici[0].ToString();
                                                    t2 = cumleici[1].ToString();
```

```
                    if (cumleici[n].ToString().Contains("."))
                    {
                        t3 += cumleici[n].ToString() + " ";
                    }
                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;
            }
        }
    }
    else if (cb3.Contains("2"))//m3 tek kelime
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith("."))
            {
                t1 = cumleici[0].ToString();
                t2 = cumleici[1].ToString();
                t3 = cumleici[2].ToString();
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;
            }
        }
```

```
                }
                else
                {
                    //if()//m2 keyword
                    //{
                    //}
                    //else if()//m3 keyword
                    //{
                    //}
                    //else
                    //{
                    //    if()//2 kelime kaldıysa
                    //    {
                    //    }
                    //    else
                    //    {
                    //        //son 2 kelime m3 ye gerisi m2 e
                    //    }
                    //}
                }
            }
            else
            {
                if (cb3.Contains("2"))//m3 tek kelime
                {
                    if (cb2.Contains("2"))//m2 tek kelime
                    {
                        string[] hepsi = HiddenField1.Value.Split('-');
                        for (int i = 0; i < hepsi.Length; i++)
                        {
                            cumleici.Clear();
                            cumleici2 = hepsi[i].ToString().Split(' ');
                            for (int j = 0; j < cumleici2.Length; j++)
                            {
                                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                                {
                                    cumleici.Add(cumleici2[j].ToString());
                                }
                            }
                            if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith("."))
                            {
                                for (int n = 0; n < cumleici.Count-2; n++)
                                {
                                    if (cumleici[n].ToString().Contains("."))
                                    {
                                        t1 += cumleici[n].ToString()+" ";
                                    }

                                    t2 = cumleici[cumleici.Count - 2].ToString();
```

```
                t3 = cumleici[cumleici.Count-1].ToString();

            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;
          }
        }
      }
      //else if()//m2 keyword
      //{
      //}
      //else if()//m1 keyword
      //{
      //}
      //else
      //{
      //    if()//2 kelime kaldıysa
      //    {
      //    }
      //    else
      //    {
      //        //son 2 kelime m2 ye gerisi m1 e
      //    }
      //}
    }
    else
    {
      //if()//m1 keyword
      //{
      //    if()//2 kelime kaldıysa
      //    {
      //    }
      //    else if()//m2 tek kelime
      //    {
      //    }
      //    else//son 2 kelime m3e gerisi m2 ye
      //    {
      //    }
      //}
      //else if()//m2 keyword
      //{
      //    //keyword m2 ye öncesi m1e sonrası m3e
```

```
                //}
                //else
                //{
                //    if()//2 kelime kaldıysa
                //    {
                //    }
                //    else
                //    {
                //        //son 2 kelime m2 ye gerisi m1 e
                //    }
                //}
            }
        }
    }
    else
    {
        if (cb1.Contains("2"))//m1 tek kelime
        {

            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                if (cumleici[0].ToString().EndsWith(".") &&
cumleici[1].ToString().EndsWith("."))
                {
                    for (int n = 1; n < cumleici.Count; n++)
                    {
                        t1 = cumleici[0].ToString();
                        if (cumleici[n].ToString().Contains("."))
                        {
                            t2 += cumleici[n].ToString() + " ";
                        }
                        else if (!cumleici[n].ToString().Contains("."))
                        {
                            t3 += cumleici[n].ToString() + " ";
                        }
                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
```

```
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;
                }
            }
        }
        else if (cb2.Contains("2"))//m2 tek kelime
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                if (cumleici[0].ToString().EndsWith(".") && !cumleici[cumleici.Count-
1].ToString().EndsWith("."))
                {
                    for (int n = cumleici.Count-1; n >= 0; n--)
                    {
                        if (!cumleici[n].ToString().Contains("."))
                        {
                            t3 += cumleici[n].ToString() + " ";
                        }
                        else if (cumleici[n].ToString().Contains(".") && sayac == 0)
                        {
                            t2 = cumleici[n].ToString() + " ";
                            sayac = 1;
                        }
                        else if (cumleici[n].ToString().Contains(".") && sayac == 1)
                        {
                            t1 += cumleici[n].ToString() + " ";
                        }
                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
```

```
                        t1 = ""; t2 = ""; t3 = "";
                        break;
                    }
                }
            }
            //else if()//m1 keywordse
            //{
            //}
            //else if()//m2 keywordse
            //{
            //}
            //else
            //{
            //    if()//2 kelime kaldıysa
            //    {
            //    }
            //    else
            //    {
            //        //son 2 kelime m2 ye gerisi m1 e
            //    }
            //}
        }
    }
    else  //m1 kısaltma, m2 kısaltma değil
    {
        if (cb3.Contains("1"))// m1 kısaltma, m2 kısaltma değil, m3 kısaltma
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith(".") && (!cumleici[1].ToString().EndsWith(".") ||
!cumleici[2].ToString().EndsWith(".")))
                {
                    int sayac = 0;
                    for (int n = 0; n < cumleici.Count; n++)
                    {
                        if (cumleici[n].ToString().Contains(".") && sayac ==0 )
                        {
                            t1 += cumleici[n].ToString() + " ";
                        }
```

```
                    else if (cumleici[n].ToString().Contains("."))
                    {
                        sayac =1;
                        t2 += cumleici[n].ToString() + " ";
                    }
                    else if (cumleici[n].ToString().Contains(".") && sayac == 1)
                    {
                        t3 += cumleici[n].ToString() + " ";
                    }
                }
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;
            }
        }
    }
    else// m1 kısaltma, m2 kısaltma değil, m3 kısaltma değil
    {
        if (cb2.Contains("2"))//m2 tek kelime
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                for (int n = 0; n < cumleici.Count; n++)
                {
                    if (cumleici[n].ToString().Contains("."))
                    {
                        t1 += cumleici[n].ToString() + " ";
                    }
                    else if (cumleici.Count - n >= 2 && cumleici[n].ToString() != "")
                    {
                        if (!(cumleici[cumleici.Count - 1].ToString().StartsWith("(")) &&
t1.ToString() != "")
                        {
```

```
    if (cumleici.Count - n == 3 && t3 != "")
    {
       t2 += cumleici[n].ToString() + " " + cumleici[n + 1].ToString() + " ";
       t3 += cumleici[n + 2].ToString() + " ";
       yRowadresler = find.NewRow();
       yRowadresler[0] = t1;
       yRowadresler[1] = tt1;
       yRowadresler[2] = t2;
       yRowadresler[3] = tt2;
       yRowadresler[4] = t3;
       yRowadresler[5] = tt3;
       find.Rows.Add(yRowadresler);
       t1 = ""; t2 = ""; t3 = "";
       break;
    }
    else
    {

       t2 += cumleici[n].ToString() + " ";
       t3 += cumleici[n + 1].ToString() + " ";
       yRowadresler = find.NewRow();
       yRowadresler[0] = t1;
       yRowadresler[1] = tt1;
       yRowadresler[2] = t2;
       yRowadresler[3] = tt2;
       yRowadresler[4] = t3;
       yRowadresler[5] = tt3;
       find.Rows.Add(yRowadresler);
       t1 = ""; t2 = ""; t3 = "";
       break;

    }
}
else if (cumleici.Count - n >= 2 && t1.ToString() != "")
{

    t2 += cumleici[n].ToString() + " ";
    t3 += cumleici[n + 1].ToString() + " ";
    yRowadresler = find.NewRow();
    yRowadresler[0] = t1;
    yRowadresler[1] = tt1;
    yRowadresler[2] = t2;
    yRowadresler[3] = tt2;
    yRowadresler[4] = t3;
    yRowadresler[5] = tt3;
    find.Rows.Add(yRowadresler);
    t1 = ""; t2 = ""; t3 = "";
    break;

}
```

```
                }


            }
        }
    }
    else if (cb3.Contains("2"))//m3 tek kelime
    {
        string[] hepsi = HiddenField1.Value.Split('-');
        for (int i = 0; i < hepsi.Length; i++)
        {
            cumleici.Clear();
            cumleici2 = hepsi[i].ToString().Split(' ');
            for (int j = 0; j < cumleici2.Length; j++)
            {
                if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                {
                    cumleici.Add(cumleici2[j].ToString());
                }
            }
            for (int n = 0; n < cumleici.Count; n++)
            {
                if (cumleici[n].ToString().Contains("."))
                {
                    t1 += cumleici[n].ToString() + " ";
                }
                else if (cumleici.Count - n >= 2 && cumleici[n].ToString() != "")
                {
                    if (!(cumleici[cumleici.Count - 1].ToString().StartsWith("(")) &&
t1.ToString() != "")
                    {
                        if (cumleici.Count - n == 3 && t3 != "")
                        {
                            t2 += cumleici[n].ToString() + " " + cumleici[n + 1].ToString() + " ";
                            t3 += cumleici[n + 2].ToString() + " ";
                            yRowadresler = find.NewRow();
                            yRowadresler[0] = t1;
                            yRowadresler[1] = tt1;
                            yRowadresler[2] = t2;
                            yRowadresler[3] = tt2;
                            yRowadresler[4] = t3;
                            yRowadresler[5] = tt3;
                            find.Rows.Add(yRowadresler);
                            t1 = ""; t2 = ""; t3 = "";
                            break;
                        }
                        else
                        {
```

```
                    t2 += cumleici[n].ToString() + " ";
                    t3 += cumleici[n + 1].ToString() + " ";
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;

                }
            }
            else if (cumleici.Count - n >= 2 && t1.ToString() != "")
            {

                t2 += cumleici[n].ToString() + " ";
                t3 += cumleici[n + 1].ToString() + " ";
                yRowadresler = find.NewRow();
                yRowadresler[0] = t1;
                yRowadresler[1] = tt1;
                yRowadresler[2] = t2;
                yRowadresler[3] = tt2;
                yRowadresler[4] = t3;
                yRowadresler[5] = tt3;
                find.Rows.Add(yRowadresler);
                t1 = ""; t2 = ""; t3 = "";
                break;

            }

        }


    }
  }
}
else if (txta2.Text != "") //m2 keyword
{
}
else if (txta3.Text != "") //m3 keyword
{
}
else
{
   //if() //2 kelime kaldıysa
   //{
   //}
```

```
                    //else
                    //{
                    //    //son 2 kelime m3 ye gerisi m2 e
                    //}
                }
            }
        }
    }
    else   //m1 kısaltma DEĞİL
    {
        if (cb2.Contains("1"))//m2 kısaltma
        {
            if (cb3.Contains("1"))//m3 kısaltma
            {
                if (cb2.Contains("2"))//m2 tek kelime
                {
                    string[] hepsi = HiddenField1.Value.Split('-');
                    for (int i = 0; i < hepsi.Length; i++)
                    {
                        cumleici.Clear();
                        cumleici2 = hepsi[i].ToString().Split(' ');
                        for (int j = 0; j < cumleici2.Length; j++)
                        {
                            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                            {
                                cumleici.Add(cumleici2[j].ToString());
                            }
                        }
                        if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith("."))
                        {
                            sayac = 0;
                            for (int n = 0; n < cumleici.Count; n++)
                            {
                                if (!cumleici[n].ToString().Contains("."))
                                {
                                    t1 += cumleici[n].ToString() + " ";
                                }
                                else if (cumleici[n].ToString().Contains(".") && sayac == 0)
                                {
                                    t2 = cumleici[n].ToString();
                                    sayac = 1;
                                }
                                else if (cumleici[n].ToString().Contains(".") && sayac == 1)
                                {
                                    t3 += cumleici[n].ToString() + " ";
                                }
                            }
                            yRowadresler = find.NewRow();
                            yRowadresler[0] = t1;
```

```
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;
                }
            }
        }
        else if (cb3.Contains("2"))//m3 tek kelime
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith("."))
                {
                    for (int n = cumleici.Count-1; n >= 0; n--)
                    {
                        if (cumleici[cumleici.Count].ToString().Contains("."))
                        {
                            t3 = cumleici[cumleici.Count].ToString() + " ";
                        }
                        else if (cumleici[n].ToString().Contains("."))
                        {
                            t2 += cumleici[n].ToString() + " ";
                        }
                        else if (!cumleici[n].ToString().Contains("."))
                        {
                            t1 += cumleici[n].ToString() + " ";
                        }
                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
```

```
                        find.Rows.Add(yRowadresler);
                        t1 = ""; t2 = ""; t3 = "";
                        break;
                    }
                }
            }
            //else if()//keyword m2
            //{
            //}
            //else if()//keyword m3
            //{
            //}
            //else
            //{
            //    if()//2 kelime kaldıysa
            //    {
            //    }
            //    else
            //    {
            //        //son 2 kelime m3 ye gerisi m2 e
            //    }
            //}
        }
        else //m1 ve m2 kısaltma m3 Değil
        {
        }
    }
    else    // m1 ve m2 Kısaltma değil
    {
        if (cb1.Contains("2"))//m1 tek kelime
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith("."))
                {
                    for (int n = cumleici.Count - 1; n >= 0; n--)
                    {
                        if (cumleici[cumleici.Count].ToString().Contains("."))
                        {
```

```
                    t3 = cumleici[cumleici.Count].ToString();
                }
                else if (cumleici[n].ToString().Contains("."))
                {
                    t2 += cumleici[n].ToString() + " ";
                }
                else if (!cumleici[n].ToString().Contains("."))
                {
                    t1 += cumleici[n].ToString() + " ";
                }
            }
            yRowadresler = find.NewRow();
            yRowadresler[0] = t1;
            yRowadresler[1] = tt1;
            yRowadresler[2] = t2;
            yRowadresler[3] = tt2;
            yRowadresler[4] = t3;
            yRowadresler[5] = tt3;
            find.Rows.Add(yRowadresler);
            t1 = ""; t2 = ""; t3 = "";
            break;
        }
    }
}
else if (cb2.Contains("2"))//m2 tek kelime
{

    string[] hepsi = HiddenField1.Value.Split('-');
    for (int i = 0; i < hepsi.Length; i++)
    {
        cumleici.Clear();
        cumleici2 = hepsi[i].ToString().Split(' ');
        for (int j = 0; j < cumleici2.Length; j++)
        {
            if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
            {
                cumleici.Add(cumleici2[j].ToString());
            }
        }
        if (cumleici[0].ToString().EndsWith(".") &&
cumleici[cumleici.Count].ToString().EndsWith("."))
        {
            sayac = 0;
            for (int n = cumleici.Count - 1; n >= 0; n--)
            {
                if (cumleici[n].ToString().Contains("."))
                {
                    t3 += cumleici[cumleici.Count].ToString() + " ";
                }
                else if (!cumleici[n].ToString().Contains(".") && sayac ==0)
```

```
                        {
                            t2 = cumleici[n].ToString();
                            sayac = 1;
                        }
                        else if (!cumleici[n].ToString().Contains(".") && sayac == 1)
                        {
                            t1 += cumleici[n].ToString() + " ";
                        }
                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;
                }
            }
        }
        //else if()//m1 keyword
        //{
        //}
        //else if()//m2 keyword
        //{
        //}
        //else
        //{
        //    if()//2 kelime kaldıysa
        //    {
        //    }
        //    else
        //    {
        //        //son 2 kelime m2 ye gerisi m1 e
        //    }
        //}
    }
}
else  //Hiç Kısaltma Yoksa
{
    if (cb1.Contains("2"))//m1 tek kelime
    {
        if (cb2.Contains("2"))//m2 tek kelime
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
```

```
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
                if (!cumleici[0].ToString().EndsWith(".") && !cumleici[cumleici.Count-
1].ToString().EndsWith("."))
                {
                    sayac = 0;
                    for (int n = 2; n <cumleici.Count; n++)
                    {
                        t1 = cumleici[0].ToString();
                        t2 = cumleici[1].ToString();
                        if (!cumleici[n].ToString().Contains("."))
                        {
                            t3 += cumleici[n].ToString() + " ";
                        }
                    }
                    yRowadresler = find.NewRow();
                    yRowadresler[0] = t1;
                    yRowadresler[1] = tt1;
                    yRowadresler[2] = t2;
                    yRowadresler[3] = tt2;
                    yRowadresler[4] = t3;
                    yRowadresler[5] = tt3;
                    find.Rows.Add(yRowadresler);
                    t1 = ""; t2 = ""; t3 = "";
                    break;
                }
            }
        }
        else if (cb3.Contains("2"))//m3 tek kelime
        {
            string[] hepsi = HiddenField1.Value.Split('-');
            for (int i = 0; i < hepsi.Length; i++)
            {
                cumleici.Clear();
                cumleici2 = hepsi[i].ToString().Split(' ');
                for (int j = 0; j < cumleici2.Length; j++)
                {
                    if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                    {
                        cumleici.Add(cumleici2[j].ToString());
                    }
                }
```

```
                    if (!cumleici[0].ToString().EndsWith(".") && !cumleici[cumleici.Count-
1].ToString().EndsWith("."))
                    {
                        sayac = 0;
                        for (int n = 1; n < cumleici.Count-1; n++)
                        {
                            t1 = cumleici[0].ToString();
                            t3 = cumleici[cumleici.Count-1].ToString();
                            if (!cumleici[n].ToString().Contains("."))
                            {
                                t2 += cumleici[n].ToString() + " ";
                            }
                        }
                        yRowadresler = find.NewRow();
                        yRowadresler[0] = t1;
                        yRowadresler[1] = tt1;
                        yRowadresler[2] = t2;
                        yRowadresler[3] = tt2;
                        yRowadresler[4] = t3;
                        yRowadresler[5] = tt3;
                        find.Rows.Add(yRowadresler);
                        t1 = ""; t2 = ""; t3 = "";
                        break;
                    }
                }
            }
            //else if()//m2 keyword
            //{
            //}
            //else if()//m3 keyword
            //{
            //}
            //else
            //{
            //    if()//2 kelime kaldıysa
            //    {
            //    }
            //    else
            //    {
            //        //son 2 kelime m3 ye gerisi m2 e
            //    }
            //}
        }
        else
        {
            if (cb3.Contains("2"))//m3 tek kelime
            {
                if (cb2.Contains("2"))//m2 tek kelime
                {
                    string[] hepsi = HiddenField1.Value.Split('-');
```

```
                for (int i = 0; i < hepsi.Length; i++)
                {
                    cumleici.Clear();
                    cumleici2 = hepsi[i].ToString().Split(' ');
                    for (int j = 0; j < cumleici2.Length; j++)
                    {
                        if (cumleici2[j].ToString() != "" && !cumleici2[j].StartsWith("\r"))
                        {
                            cumleici.Add(cumleici2[j].ToString());
                        }
                    }
                    if      (!cumleici[0].ToString().EndsWith(".")       &&       !cumleici[cumleici.Count-
1].ToString().EndsWith("."))
                    {
                        sayac = 0;
                        for (int n = cumleici.Count-2; n >= 0; n--)
                        {
                            t2 = cumleici[cumleici.Count - 2].ToString();
                            t3 = cumleici[cumleici.Count - 1].ToString();
                            if (!cumleici[n].ToString().Contains("."))
                            {
                                t1 += cumleici[n].ToString() + " ";
                            }
                        }
                        yRowadresler = find.NewRow();
                        yRowadresler[0] = t1;
                        yRowadresler[1] = tt1;
                        yRowadresler[2] = t2;
                        yRowadresler[3] = tt2;
                        yRowadresler[4] = t3;
                        yRowadresler[5] = tt3;
                        find.Rows.Add(yRowadresler);
                        t1 = ""; t2 = ""; t3 = "";
                        break;
                    }
                }
            }
            //else if()//m2 keyword
            //{
            //}
            //else if()//m1 keyword
            //{
            //}
            //else
            //{
            //    if()//2 kelime kaldıysa
            //    {
            //    }
            //    else
            //    {
```

```
    //        //son 2 kelime m2 ye gerisi m1 e
    //    }
    //}
}
else
{
    //if()//m1 keyword
    //{
    //    if()//m2 tek kelime
    //    {
    //    }
    //    else
    //    {
    //        if()//2 kelime kaldıysa
    //        {
    //        }
    //        else
    //        {
    //            //son 2 kelime m3 ye gerisi m2 e
    //        }
    //    }
    //}
    //else if()//m2 keyword
    //{
    //}
    //else
    //{
    //    if()//m2 tek kelime
    //    {
    //    }
    //    else if()//m2 keyword
    //    {
    //    }
    //    else if()//m1 keyword
    //    {
    //        if()//2 kelime kaldıysa
    //        {
    //        }
    //        else
    //        {
    //            //son 2 kelime m3 ye gerisi m2 e
    //        }
    //    }
    //    else
    //    {
    //        if()//2 kelime kaldıysa
    //        {
    //        }
    //        else
    //        {
```

```
                        //          //son 2 kelime m2 ye gerisi m1 e
                        //      }
                        //  }
                        //}
                    }
                }
            }
        }
    }
    if (find.Rows.Count > 0)
    {
        System.IO.StreamWriter yazalim = new System.IO.StreamWriter("c:\\XML\\" + txtkey.Text +
".xml", false, System.Text.Encoding.GetEncoding(1254));
        System.IO.StreamWriter yazalim2 = new System.IO.StreamWriter("c:\\OWL\\" + txtkey.Text +
".owl", false, System.Text.Encoding.GetEncoding(1254));
        yazalim.WriteLine("<?xml version=\"1.0\" encoding=\"ISO-8859-9\" ?>");
        yazalim2.WriteLine("<?xml version=\"1.0\"?>");
        yazalim2.WriteLine("<rdf:RDF");
        yazalim2.WriteLine("xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"");
        yazalim2.WriteLine("xmlns:xsd=\"http://www.w3.org/2001/XMLSchema#\"");
        yazalim2.WriteLine("xmlns:rdfs=\"http://www.w3.org/2000/01/rdf-schema#\"");
        yazalim2.WriteLine("xmlns:owl=\"http://www.w3.org/2002/07/owl#\"");
        yazalim2.WriteLine("xmlns=\"http://www.owl-ontologies.com/unnamed.owl#\"");
        yazalim2.WriteLine("xml:base=\"http://www.owl-ontologies.com/unnamed.owl\">");
        yazalim2.WriteLine("<owl:Ontology rdf:about=\"\"/>\"");

        yazalim.WriteLine("<" + txtmbas.Text + ">");
        yazalim2.WriteLine("<owl:Class rdf:ID=\" " + txtmbas.Text + "\"/>");

        yazalim2.WriteLine("<owl:DatatypeProperty rdf:ID=\"" + txtm1.Text + "\">");
        yazalim2.WriteLine("<rdfs:domain rdf:resource=\"#" + txtmbas.Text + "\"/>");
        yazalim2.WriteLine("<rdfs:range      rdf:resource=\"http://www.w3.org/2001/XMLSchema#"      +
find.Rows[0][1].ToString() + "\"/>");
        yazalim2.WriteLine("</owl:DatatypeProperty>");

        yazalim2.WriteLine("<owl:DatatypeProperty rdf:ID=\"" + txtm2.Text + "\">");
        yazalim2.WriteLine("<rdfs:domain rdf:resource=\"#" + txtmbas.Text + "\"/>");
        yazalim2.WriteLine("<rdfs:range      rdf:resource=\"http://www.w3.org/2001/XMLSchema#"      +
find.Rows[0][3].ToString() + "\"/>");
        yazalim2.WriteLine("</owl:DatatypeProperty>");

        yazalim2.WriteLine("<owl:DatatypeProperty rdf:ID=\"" + txtm3.Text + "\">");
        yazalim2.WriteLine("<rdfs:domain rdf:resource=\"#" + txtmbas.Text + "\"/>");
        yazalim2.WriteLine("<rdfs:range      rdf:resource=\"http://www.w3.org/2001/XMLSchema#"      +
find.Rows[0][5].ToString() + "\"/>");
        yazalim2.WriteLine("</owl:DatatypeProperty>");


        for (int i = 0; i < find.Rows.Count; i++)
```

```
                    {
                        yazalim2.WriteLine("<" + txtmbas.Text + " rdf:ID=\"" + txtmbas.Text + "_" + (i + 1) + "\">");
                        yazalim2.WriteLine("<" + txtm1.Text + " rdf:datatype=\"http://www.w3.org/2001/XMLSchema#"
+ find.Rows[i][1].ToString() + "\">" + find.Rows[i][0].ToString() + "</" + txtm1.Text + ">");
                        yazalim2.WriteLine("<" + txtm2.Text + " rdf:datatype=\"http://www.w3.org/2001/XMLSchema#"
+ find.Rows[i][3].ToString() + "\">" + find.Rows[i][2].ToString() + "</" + txtm2.Text + ">");
                        yazalim2.WriteLine("<" + txtm3.Text + " rdf:datatype=\"http://www.w3.org/2001/XMLSchema#"
+ find.Rows[i][5].ToString() + "\">" + find.Rows[i][4].ToString() + "</" + txtm3.Text + ">");
                        yazalim2.WriteLine("</" + txtmbas.Text + ">");
                    }
                    yazalim2.WriteLine("</rdf:RDF>");
                    yazalim2.Close();
                    for (int i = 0; i < find.Rows.Count; i++)
                    {
                        yazalim.WriteLine("<" + txtmbas.Text + (i + 1) + ">");
                        yazalim.WriteLine("<" + txtm1.Text + ">");
                        yazalim.WriteLine("<" + find.Rows[i][1].ToString() + ">");
                        yazalim.WriteLine(find.Rows[i][0].ToString());
                        yazalim.WriteLine("</" + find.Rows[i][1].ToString() + ">");
                        yazalim.WriteLine("</" + txtm1.Text + ">");
                        yazalim.WriteLine("<" + txtm2.Text + ">");
                        yazalim.WriteLine("<" + find.Rows[i][3].ToString() + ">");
                        yazalim.WriteLine(find.Rows[i][2].ToString());
                        yazalim.WriteLine("</" + find.Rows[i][3].ToString() + ">");
                        yazalim.WriteLine("</" + txtm2.Text + ">");
                        yazalim.WriteLine("<" + txtm3.Text + ">");
                        yazalim.WriteLine("<" + find.Rows[i][5].ToString() + ">");
                        yazalim.WriteLine(find.Rows[i][4].ToString());
                        yazalim.WriteLine("</" + find.Rows[i][5].ToString() + ">");
                        yazalim.WriteLine("</" + txtm3.Text + ">");
                        yazalim.WriteLine("</" + txtmbas.Text + (i + 1) + ">");
                    }

                    yazalim.WriteLine("</" + txtmbas.Text + ">");
                    yazalim.Close();

                }
                else
                {
                    string alert = "<script language='javascript' type='text/javascript'>";
                    alert += "function alert(){";
                    alert += "alert('Kriterlerinizi kontrol edip tekrar deneyiniz..');";
                    alert += "}</script>";
                    Label1.Text = alert;
                }

            }
            protected void btprint_Click(object sender, EventArgs e)
            {
```

```
    }
}
```