

| | | | |
|-----------------|-----------------|--------------------------|--------------|
| ID | 1 | Creation date | May 4, 2008 |
| Platform | S60 3rd Edition | Tested on devices | Nokia N93 |
| Category | Python, C# | Subcategory | Semantic Web |

Contents

- [1 Semantic Address Book](#)
- [2 Overview](#)
- [3 Simple Code Example](#)
- [4 Client Script](#)
- [5 Server Side](#)
- [6 Possible Extensions](#)
- [7 References](#)
- [8 Link](#)

Semantic Address Book

Semantic Address Book is a simple example that shows how we can incorporate the idea of semantic web in the mobile applications. Semantic Address Book is a mobile application that will find relationships between the Mobile User (Owner) and the list of contacts stored in his/her Mobile. Based on the semantic rules, it will detect and alert Owner if any of its existing contact persons change their Mobile Numbers.

In order to make this code example simple, we have tried to limit the domain of this application. However, we can add as many rules as we can to find more relations.

Overview

This article will show how to incorporate Semantic Web concept in mobile applications. There are few technologies that are using semantic web, for example Microformats and RDFa, but they are mainly focused on web pages. Therefore, we tried to incorporate the main principals of semantic web by our custom format that is based on XML and focused on problem in hand (Mobile Phone Address Book Contacts).

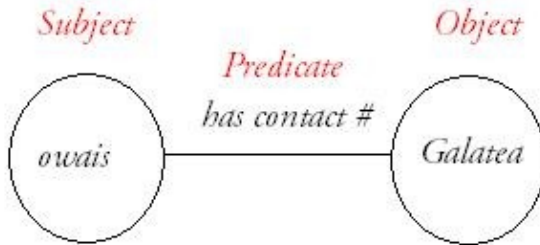
But before starting the walkthrough of the code example, it is important to get the understanding what is semantic web and how we are using semantic web in this example.

Semantic Web is basically provides language so that machines can understand the contents. Contents can be web pages, people, media, place or in our case Mobile Contacts. The advantages of semantic web are, it provides better understanding of contents and can help us in closing the gap between what is desired and what we are getting. Semantic Web deals with three concepts. Subject, predicate and Object. Let's take an example (in our custom format:

Semantic_Address_Book

```
<?xml version="1.0" encoding="utf-8"?>
<SemanticContacts ID="SemanticContacts">
  <Owner Name="owais" Number="923002211046">
    <Contact Name="galatea" Number="923002255667" />
  </Owner>
  <Owner Name="galatea" Number="923002255667">
    <Contact Name="jhosh" Number="923234422112" />
  </Owner>
</SemanticContacts>
```

According to this example, Mobile Phone Owner "Owais" has a contact number of "Galatea".



This is similar to FOAF. Secondly, semantic web also provide a vocabulary so that machines can understand what the things are. For example, according to the above example by the tag of <Owner> machines knows that "Owais" is a Mobile Phone owner and so is "Galatea".

Note: Here we are not using URIs as this example is specifically design for mobile phones. In future we may try to use it.

Simple Code Example

The code example consist of two parts, Client script (which is in Python) and Server Program (which is in .NET C#). Following are the steps:

- Server starts and listens to a specific port.
- Client connects to the server.
- Client send the Client ID to server (we are using ownername&number# string)
- Server will load or create the xml tree of the Client.(e.g, if new client "Ahad" connects to the server, server will add another XML node in the main XML file.)

```
<?xml version="1.0" encoding="utf-8"?>
<SemanticContacts ID="SemanticContacts">
  <Owner Name="owais" Number="923002211046">
    <Contact Name="galatea" Number="923212211212" />
  </Owner>
  <Owner Name="galatea" Number="923002255667">
    <Contact Name="jhosh" Number="923234422112" />
  </Owner>
  <Owner Name="Ahad" Number="923441122116">
    </Owner>
</SemanticContacts>
```

Semantic_Address_Book

- Server sends "SendContacts" command to client.
- Client access ContactDB and send entire contacts (Note: we can optimize this routine, but its not covered in this example).
- Server synchronizes the contacts to make an xml tree of the Client.(e.g, if Connected Client "Ahad" contains two contact numbers, then it will be added in the tree as well.)

```
<?xml version="1.0" encoding="utf-8"?>
<SemanticContacts ID="SemanticContacts">
  <Owner Name="owais" Number="923002211046">
    <Contact Name="galatea" Number="923212211212" />
  </Owner>
  <Owner Name="galatea" Number="923002255667">
    <Contact Name="jhosh" Number="923234422112" />
  </Owner>
  <Owner Name="Ahad" Number="923441122116">
    <Contact Name="jhosh" Number="923234422112" />
    <Contact Name="galatea" Number="923002255667" />
  </Owner>
</SemanticContacts>
```

- Now, server starts the rule base inference of the whole Semantic Address Book. If server find any alters it will send notification to the clients.

Notice in the above xml, Owner "Owais" contains the old number of "Galatea". Server will find this difference and notify "Owais" to update its contact.

- Client gets notification and updates the address book.
- Client closes connection.

Client Script

```
import contacts
import socket
import time

# script 1, iterate through contact entries in the default database.

# open the default database.
def script1():
    db=contacts.open()
    for id in db:
        print 'Contact:%s'%db[id]
    print 'number of entries:%i'%len(db)

HOST = 'someURL'      # The remote host
PORT = 3000           # The same port as used by the server
print "define socket"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "trying to connect to socket"
s.connect((HOST, PORT))
print "connected"
print "1: send MobileUserName and Number like \"owais&923002211046#\" "
```

Semantic_Address_Book

```
s.send('owais&923002211046#')
print "data send"
data = s.recv(1024)

if data == 'SendContacts#':
    for id in db:
        s.send(db[id]['first_name'] + '&' + db[id]['mobile_number'] + '#')
    s.send('##')

data = s.recv(1024)
print "2: If UpdateExist the program will modify the Contact DB ...."
```

Server Side

```
/// <summary>
/// Demo program on "Semantic Address Book" for "Code Example"
/// By: Owais Zahid
/// </summary>
///
class Program
{
    #region Private Members
    private TcpListener tcpMobileListener = null;
    private MobileClient IncomingClient = null;
    private XmlDocument SemanticDocument = null;
    private XmlNode ClientNode = null;
    #endregion

    static void Main(string[] args)
    {
        Program instance = new Program();

        // 1: Start the TCP Server
        instance.StartServer();

        // 2: Load the Xml File that contain Contacts Data
        instance.LoadXml();

        // 3: Get the Client Contacts from Mobile and Update Xml File
        instance.GetClientCredentials();

        // 4: Match Finder Algorithm will traverse Xml File for updates
        instance.MatchFinder();

        // 5: Server Send Updates to Client (if any)
        instance.SendClientUpdates();

        Console.WriteLine("Work Finished");
        Console.ReadLine();
        // 6: Stop Server
        instance.StopServer();

        // 7: Save Xml
        instance.SaveXml();

    }

    private void SaveXml()
```

Semantic_Address_Book

```
{
    SemanticDocument.Save("SemanticAddressBook.xml");
}

private void LoadXml()
{
    // Call Method to add Contacts to the xml File
    SemanticDocument = new XmlDocument();
    SemanticDocument.Load("SemanticAddressBook.xml");
}

private void StopServer()
{
    IncomingClient.ClientSender.WriteLine("ConnectionClose");
    IncomingClient.Close();
    tcpMobileListener.Stop();
}

private void SendClientUpdates()
{
    foreach (XmlNode node in ClientNode.ChildNodes)
    {
        if (node.Attributes.Count == 3)
        {
            IncomingClient.ClientSender.WriteLine("UpdatesExists#");
            IncomingClient.ClientSender.Flush();

            string strClientResponse = IncomingClient.ClientReader.ReadLine();
            if (strClientResponse == "SendUpdates#")
            {
                IncomingClient.ClientSender.WriteLine(node.Attributes[0].Value + "&" + no
                IncomingClient.ClientSender.Flush();
            }
            node.Attributes.RemoveAt(2);
        }
    }
}

private void GetClientCredentials()
{
    // Decided Protocol.
    // Mobile Will Send "Owner Name", "Cell Phone Number"
    // e.g owais&923002211046#
    string strMobileID = IncomingClient.ClientReader.ReadLine();

    XmlNodeList ownerNodeList = SemanticDocument.GetElementsByTagName("Owner");
    ClientNode = null;
    foreach (XmlNode node in ownerNodeList)
    {
        if (strMobileID == (node.Attributes["Name"].Value + "&" + node.Attributes["Number
        {
            ClientNode = node;
            break;
        }
    }
    if (ClientNode == null)
    {
        strMobileID = strMobileID.Substring(0, strMobileID.Length - 1);

        XmlElement newOwner = SemanticDocument.CreateElement("Owner");
```

Semantic_Address_Book

```
XmlAttribute newAttribute = SemanticDocument.CreateAttribute("Name");
newAttribute.Value = strMobileID.Split('&')[0];
newOwner.Attributes.Append(newAttribute);
newAttribute = SemanticDocument.CreateAttribute("Number");
newAttribute.Value = strMobileID.Split('&')[1];
newOwner.Attributes.Append(newAttribute);

SemanticDocument.GetElementsByTagName("SemanticContacts")[0].AppendChild(newOwner);
ClientNode = newOwner;
}
else
{
    ClientNode.InnerXml = "";
}

//server will request client to send contacts.
// {
IncomingClient.ClientSender.WriteLine("SendContacts#");
IncomingClient.ClientSender.Flush();

while (true)
{
    string strContacts = IncomingClient.ClientReader.ReadLine();
    if (strContacts == "##") // Contacts List is finished
        break;
    strContacts = strContacts.Substring(0, strContacts.Length - 1);

    XmlElement newContact = SemanticDocument.CreateElement("Contact");
    XmlAttribute newAttribute = SemanticDocument.CreateAttribute("Name");

    newAttribute.Value = strContacts.Split('&')[0];
    newContact.Attributes.Append(newAttribute);
    newAttribute = SemanticDocument.CreateAttribute("Number");
    newAttribute.Value = strContacts.Split('&')[1];
    newContact.Attributes.Append(newAttribute);

    ClientNode.AppendChild(newContact);
}
}

private void StartServer()
{
    // Make a TCP Listener that will listen to the incoming mobile request.
    try
    {
        if (tcpMobileListener == null)
        {
            tcpMobileListener = new TcpListener(IPAddress.Any, 4000);
            tcpMobileListener.Start();
            IncomingClient = new MobileClient(tcpMobileListener.AcceptTcpClient());
        }
    }
    catch (Exception e1)
    {
        // Send Appropriate Error Message....
    }
}

// Matching Algorithm
private void MatchFinder()
{
    XmlNodeList ownerlist = SemanticDocument.GetElementsByTagName("Owner");
```

Semantic_Address_Book

```
foreach (XmlNode childNode in ClientNode.ChildNodes)
{
    foreach (XmlNode ownerNode in ownerlist)
    {
        if (childNode.Attributes[0].Value == ownerNode.Attributes[0].Value)
        {
            // Contact having the same name.
            if (childNode.Attributes[1] != ownerNode.Attributes[1])
            {
                // Must Send Notification
                XmlAttribute att = SemanticDocument.CreateAttribute("Change");
                childNode.Attributes.Append(att);
            }
        }
    }
}
}
```

Possible Extensions

The stated code example is a simple example that shows what we can achieve by integrating Semantics to mobile applications. However, there are some limitations on this code example, but we can extend this idea and add more realistic rules to make this application more usable.

One way to extend this application is to add "Learning Algorithms" like supervised or reward based learning algorithm so that machines can evolve themselves and helps in finding the correct answer.

We have written server side code in .NET C#, but we can easily use Java , Python or anything on server side. I am also uploading the zip file of my server side.

References

Following are the links and url that are used while making this code example:

- [\[1\]](#)
- [\[2\]](#)
- [\[3\]](#)
- [\[4\]](#)
- [\[5\]](#)

Link

[File:SemanticAddressBook.zip](#)

--[Hypatia](#) 21:52, 7 May 2008 (EEST)