Pengembangan Aplikasi *Wrapper* Untuk Ekstraksi Data Pada Halaman Web Dengan Menggunakan Python

Lintang Y. Banowosari[1], Didik Pamungkas[2], I Wayan S.Wicaksana[3], A.Benny Mutiara[4]

{lintang,iwayan,amutiara}@gunadarma.ac.id , depe@yahoo.com

1,2,3,4 Universitas Gunadarma

Jl. Margonda Raya No. 100 Pondok Cina Depok 16424

Abstrak

Web yang tidak terstruktur saat ini berisi berjuta-juta dokumen yang sulit di *query* dan struktur yang bercampur. Data terstruktur menjadi suatu hal yang diperlukan dalam suatu sistem informasi yang saling terhubung. Dengan *Semantic Web*, semua informasi yang tersedia pada suatu *web* akan menjadi lebih terstruktur dan berkualitas. Metode yang dapat digunakan untuk dapat mengekstrak data dari halaman *web* dan mengubahnya menjadi data yang lebih terstruktur adalah dengan menggunakan suatu modul yang disebut *wrapper*.

Pada awalnya proses *wrapping* menggunakan metode manual. Yaitu dengan beberapa tahapan, yang pertama menerima halaman web, kemudian yang kedua melakukan proses *parsing* dari halaman web tersebut (format HTML). Proses *parsing* ini tidak akan mengeluarkan semua tag HTML, tapi memilih hanya pada struktur dokumen, format logical, dan format fisik. Dan yang ketiga adalah menempatkan informasi yang telah diperoleh ke dalam bentuk yang lebih terstruktur, misalnya dalam format XML. Permasalahan yang timbul adalah terkadang menyulitkan untuk memahami isi dari masing-masing halaman *web* tersebut serta sangat membutuhkan tenaga kerja yang banyak.

Untuk mengatasi kesulitan tersebut penulis mengembangkan suatu aplikasi *wrapper* yang digunakan untuk mengekstrak data yang semi-terstruktur menjadi data yang terstruktur. Yang akan memudahkan dalam menentukan *class* dan *properties* dari suatu data pada halaman *web*, sehingga nantinya akan memudahkan dalam memahami isi dari suatu halaman *web*. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman Python 2.5.

Kata Kunci: *Semantic Web, Wrapper, Class, Properties, Python*

Pendahuluan

Di dalam berbagai kegiatan yang ada saat ini, sangat membutuhkan berbagai informasi, seperti untuk pengambilan keputusan, perencanaan, evaluasi dan lain sebagainya. Sumber informasi pada saat ini telah semakin beragam dan banyak. Terlebih dengan semakin berkembangnya teknologi informasi dan internet, yang membuat suatu sistem tidak ada batasan geografis dan waktu. Hal ini yang mendorong semakin memudahkannya pertukaran informasi, dan menimbulkan keragaman sumber informasi.

Sumber informasi yang telah tersedia dewasa ini (yang terdapat dalam suatu *web*) memiliki metode penyajian yang berbeda-beda walaupun tujuan dari pembuatan atau penyajian *web* tersebut sama, hal ini dikarenakan teknologi *web* memiliki tingkat autonomi yang tinggi. Hal ini juga yang membutuhkan suatu metode penafsiran yang berbeda - beda untuk masing-masing *web* tersebut walaupun pada akhirnya akan memiliki hasil yang sama.

Setiap orang yang menginginkan sebuah informasi dapat dengan mudah didapat melalui media internet, walaupun informasi yang didapat tersebut tidak sepenuhnya relevan dengan apa yang diinginkan oleh orang tersebut, oleh karena itu dibutuhkan sebuah metode pencarian informasi yang dapat setidaknya mendekati atau menyaring informasi yang diinginkan oleh orang tersebut.

Karena masalah keragaman tersebut, maka untuk pertukaran informasi akan menjadi suatu kendala yang sulit untuk dipertemukan tanpa ada solusi yang efektif. Salah satu pendekatan yang memungkinkan untuk menjembatani masalah ini adalah menggunakan *Semantic Web* yang memanfaatkan teknologi *Ontology*.

Semantic Web adalah sekumpulan informasi yang dikumpulkan dengan metode tertentu agar dapat dengan mudah diproses dengan mesin, dalam skala yang besar. Ini seperti cara yang efisien dari representasi data pada *World Wide Web*, atau sebagai database global yang saling terhubung.

Namun *Semantic Web* masih menjadi suatu visi. *Web* yang tidak terstruktur saat ini berisi berjuta-juta dokumen yang sulit di *query* dan struktur yang bercampur. Data terstruktur menjadi suatu hal yang diperlukan dalam suatu sistem informasi yang saling terhubung. Dengan *Semantic Web*, semua informasi yang tersedia pada suatu *web* akan menjadi lebih terstruktur dan berkualitas.

Dalam sebuah halaman *web*, data yang ditampilkan biasanya dalam format HTML. Format ini masih belum terstruktur ataupun semi-terstruktur, yang akan diubah menjadi format yang lebih terstruktur, seperti XML. Metode yang dapat digunakan untuk dapat mengekstrak data dari HTML dan mengubahnya menjadi data yang lebih terstruktur menggunakan suatu modul yang disebut *wrapper*.

Pada awalnya proses *wrapper* menggunakan metode manual. Yaitu dengan beberapa tahapan, yang pertama menerima halaman web, kemudian yang kedua melakukan proses *parsing* dari halaman web tersebut (format HTML). Proses *parsing* ini tidak akan mengeluarkan semua tag HTML. Tapi memilih hanya pada struktur dokumen, format logical, dan format fisik. Informasi tentang warna, latar belakang, jenis font, dan format visual yang lainnya akan diabaikan. Dan yang ketiga adalah menempatkan informasi yang telah diperoleh ke dalam bentuk yang lebih terstruktur, misalnya dalam format XML.

Permasalahan yang timbul adalah terkadang menyulitkan untuk memahami isi dari masing-masing halaman *web* tersebut serta sangat membutuhkan tenaga kerja yang banyak. Salah satu cara untuk mengatasi kesulitan dalam memahami isi dari halaman *web* tersebut, seperti *properties* dan *attributes* dari suatu data pada halaman *web*, adalah dengan dibuat suatu aplikasi *wrapper*.

Data yang ditampilkan dalam halaman *web* biasanya adalah data yang tidak terstruktur atau semi-terstruktur. Permasalahan yang timbul adalah bagaimana melakukan proses *wrapping*

dari data yang tidak terstruktur tersebut menjadi data yang lebih terstruktur. Sehingga nantinya akan memudahkan dalam memahami isi dari suatu halaman *web*.

Dalam penulisan ini akan dibahas mengenai pembuatan modul *wrapper*, yang akan digunakan untuk mengekstrak data yang semi-terstruktur sehingga menjadi data yang terstruktur. Sehingga memudahkan dalam menentukan *properties* dan *attributes* dari suatu data pada halaman *web*. Serta dilakukan pengujian terhadap aplikasi *wrapper* yang dihasilkan.

Penulisan ini adalah membahas bagaiamana membuat suatu aplikasi *wrapper* untuk memudahkan dalam memahami isi dari suatu halaman *web*. Sehingga nantinya dapat digunakan oleh pengguna lain untuk memudahkan dalam memahami isi dari suatu halaman *web*.

Metode penelitian yang digunakan adalah:

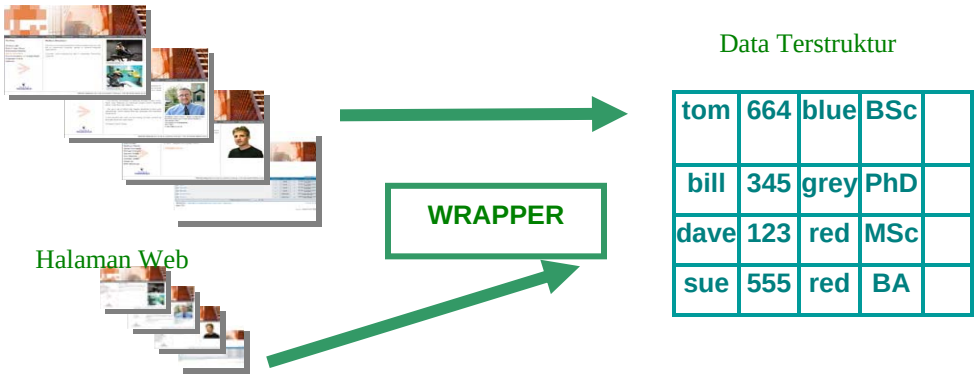
1. Studi kepustakaan*,* dari berbagai artikel dan sumber mengenai *Semantic Web* maupun *wrapper* itu sendiri;
2. Menentukan alat perancangan yang digunakan untuk merancang aplikasi, yaitu menggunakan Unified Modelling Language (UML);
3. Menentukan bahasa pemrograman yang digunakan dan mengimplementasi rancangan ke dalam *coding*, yaitu Python;
4. Melakukan pengujian terhadap aplikasi yang telah dibuat dan melakukan perbaikan kesalahan yang terjadi;
5. Implementasi aplikasi;

Pada bagian awal diuraikan latar belakang masalah, rumusan masalah, batasan masalah, serta tujuan penulisan. Bagian selanjutnya memberikan sekilas latar belakang teori yang melandasi *Semantic Web* dan aplikasi *wrapper* serta *tools* yang digunakan dalam pembuatan aplikasi ini yaitu, bahasa pemrograman *Python*. Bagian tiga menjelaskan mengenai langkah-langkah pembuatan aplikasi *wrapper,* mulai dari perancangan sampai pengimplementasian ke dalam *coding*. Pengaplikasian dan pengujian terhadap aplikasi *wrapper* yang telah dibuat akan dibahas pada bagian empat. Dan terakhir pada bagian lima yang merupakan penutup akan merupakan kesimpulan serta saran untuk penelitian yang akan datang.

Ekstraksi Data dalam Halaman Web

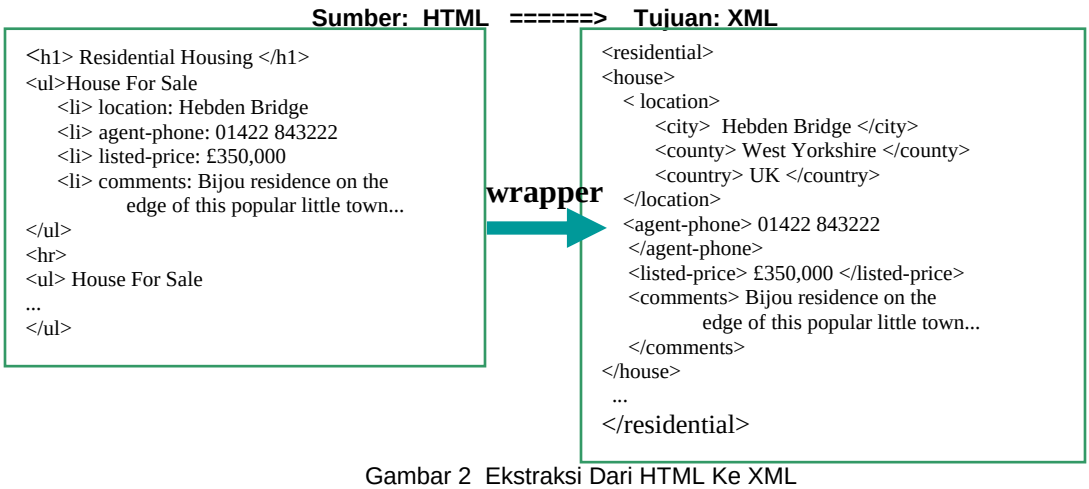
Wrapper

Wrapper adalah suatu proses ektraksi data dari suatu halaman *web* yang semula tidak terstruktur atau semi terstruktur menjadi data yang lebih terstruktur. Dalam sebuah halaman *web*, data yang ditampilkan biasanya dalam format HTML. Format ini masih belum terstruktur atau semi terstruktur, yang akan diubah menjadi format yang lebih terstruktur, seperti XML.



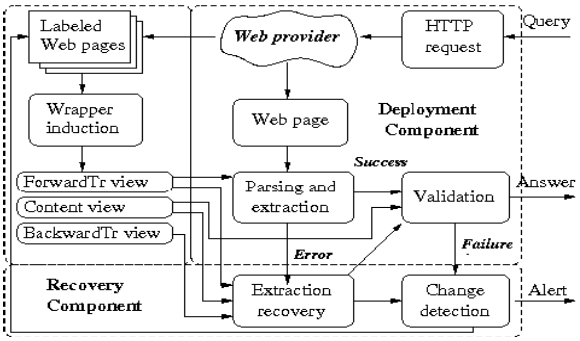
Gambar 1 *Ekstrasi Data dari Halaman Web*

Informasi yang ditampilkan dalam sebuah halaman web merupakan informasi yang tidak terstruktur atau yang semi terstruktur. Proses yang dilakukan dalam kegiatan *wrapping* meliputi menerima halaman *web* kemudian mengekstrak informasi dari halaman *web*, dan yang terakhir adalah menempatkan informasi yang telah diekstrak ke bentuk XML sebagai hasil yang didapat dari proses *wrapping*.



Gambar 2 Ekstraksi Dari HTML Ke XML

Komponen *Wrapper*



Gambar 3 Arsitektur Wrapper

Gambar di atas menampilkan arsitektur *wrapper* yang terdiri dari tiga komponen utama yaitu *Generation, Deployment* dan *Recovery component*. Bermula dari *Generation component* dengan contoh halaman HTML yang sudah diberi label dari sebuah *web provider*. Kemudian halaman HTML yang sudah diberi label tersebut dimasukkan ke dalam sebuah *wrapper* yang mana digunakan dalam *Deployment component* untuk memproses segala halaman baru dari *provider*. Semua data yang telah diekstrak oleh *wrapper* memperoleh validasi. Jika *provider* merubah format halaman HTML, *wrapper* akan gagal untuk menyelesaikan ekstraksi data. Dalam kasus tertentu, prosedur *extraction recovery* mencoba untuk mengekstrak sebanyak mungkin data dari halaman. Suksesnya hasil suatu *recovery* dalam *automatic re-labeling* dari halaman baru dapat digunakan untuk menghasilkan sebuah *wrapper* versi baru yang mengakomodasi format halaman baru.

Metode Pengembangan *Wrapper*

Tiga proses utama yang dilakukan dalam pengolahan bahasa alami ialah analisa sintaksis, interpretasi semantik dan interpretasi kontekstual. Analisa sintaksis atau *parsing* ialah proses penentuan struktur sebuah kalimat berdasarkan *grammar* dan *lexicon* tertentu. *Parsing* dapat dilakukan secara *top-down* maupun *bottom-up*, masing-masing memiliki kelebihan dan kekurangannya sendiri. *Top-down parsing* tidak dapat menangani *grammar* dengan *left-recursion*, sedangkan *bottom-up parsing* tidak dapat menangani *grammar* dengan *empty production*. Karena itu metode *parsing* yang terbaik ialah yang dapat menggabungkan kedua cara ini.

Interpretasi semantik ialah proses penerjemahan sebuah kalimat menjadi bentuk representasi artinya yang umum disebut *logical form* tanpa memperhatikan konteks. Dua proses utama yang diperlukan dalam membentuk *logical form* ialah penentuan peran tiap kata dan frase dalam kalimat, serta pemilihan arti kata yang tepat untuk membentuk kalimat yang masuk akal. Peranan kata-kata dan frase dalam kalimat dapat direpresentasikan dalam bentuk predikat-argumen biasa ataupun menggunakan *thematic roles*. Sedangkan proses pemilihan arti kata yang tepat dapat dilakukan dengan *selectional restrictions* ataupun *context activation*.

Metode yang dipakai dalam proses *wrapping* adalah metode *parsing* yang dilakukan kata perkata yang diambil dari halaman *web* yang di*download*. Halaman *web* yang di*download* dan yang akan di-*parsing* hanya halaman *web* teks dan teks/html yang merupakan tipe MIME. Untuk mem-*parsing* halaman HTML digunakan *event-oriented parser*. Di mana *event-oriented parser* ini tidak membangun struktur dari dokumen, tapi mengeneralisasi fungsi, seperti pada gambar di bawah ini:

Gambar 4 Event –Oriented parsing dari HTML

Selama melakukan proses *parsing* duplikasi dilakukan berdasarkan konten halaman, tetapi *link* dari halaman yang ditemukan terduplikasi akan diabaikan karena akan mengurangi *bandwidth*.

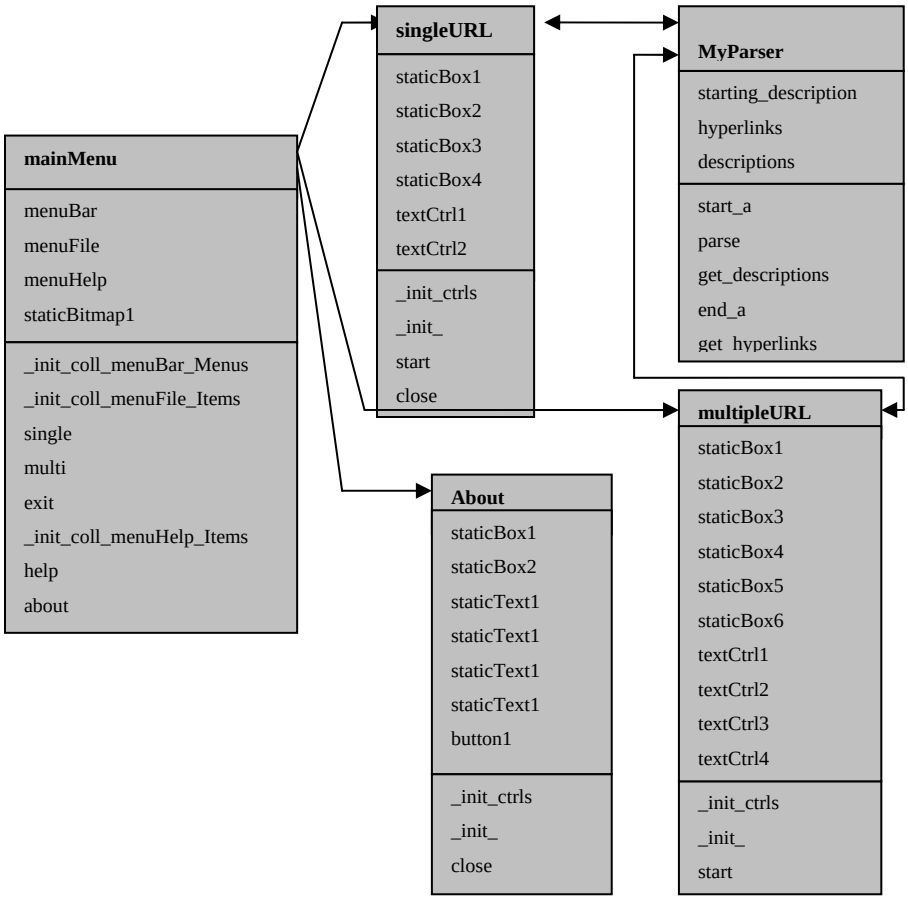
Parser ini tidak akan mengeluarkan semua tag HTML. Tapi memilih hanya pada struktur dokumen, format logikal, dan format fisik seperti huruf tebal dan huruf miring. Informasi tentang warna, latar belakang, jenis font, dan format visual yang lainnya akan diabaikan.

Bahasa Pemrograman Python

Bahasa pemrograman *Python* bersifat *freeware* sehingga tidak ada batasan dalam penyalinan atau mendistribusikannya. Paket instalasi *Python* dapat di-*download* dari situs http://www.python.org/download/. *Python* dapat digunakan dalam beberapa sistem operasi, seperti UNIX, Linux, DOS, Windows, OS/2, Macintosh dan lainnya.

Perancangan Aplikasi

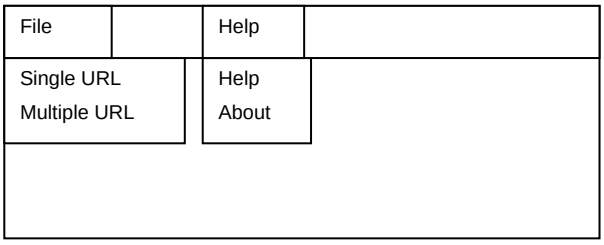
Untuk memudahkan dalam mendesain tampilan berbasis Graphical User Interface (GUI), digunakan komponen pendukung lainnya, yaitu *Boa Constructor*. *Boa Constructor* yang digunakan merupakan versi 0.4.4 – Alpha.



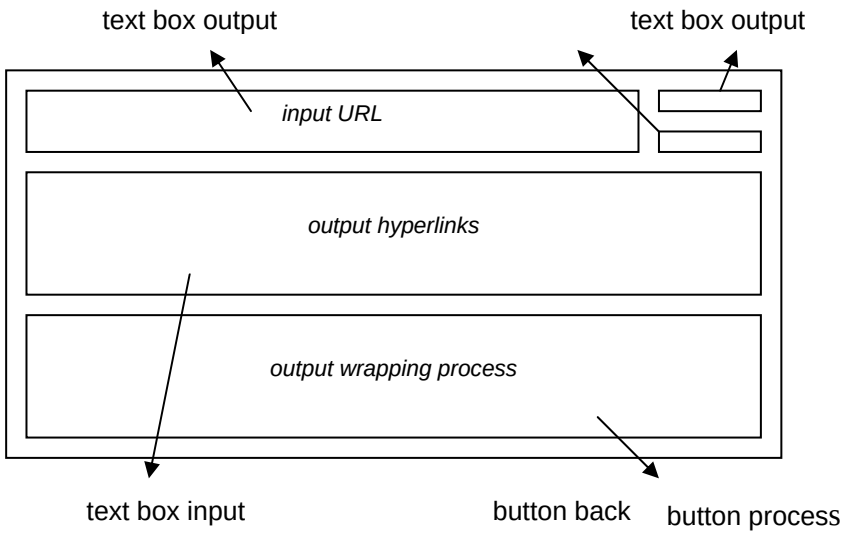
Gambar 5 *Class Diagram*

Rancangan Tampilan Aplikasi

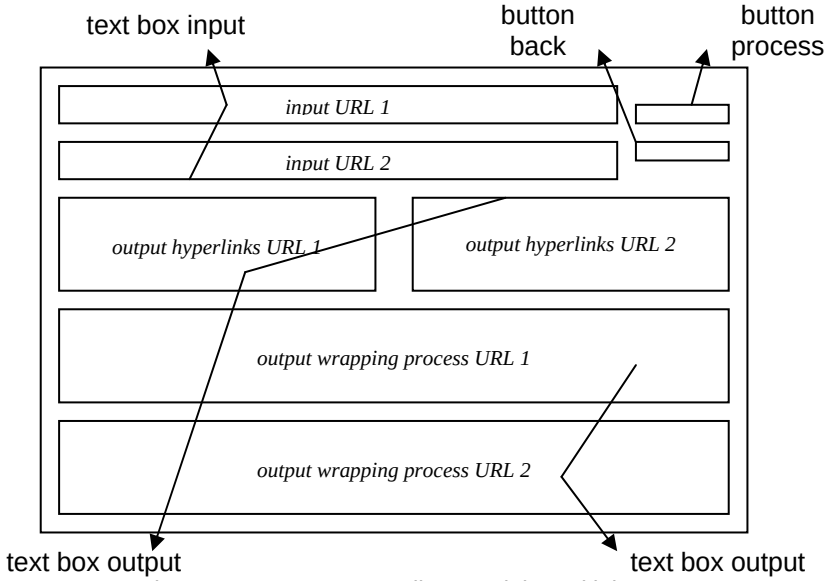
Berikut ini adalah rancangan tampilan Aplikasi *Wrapper*:



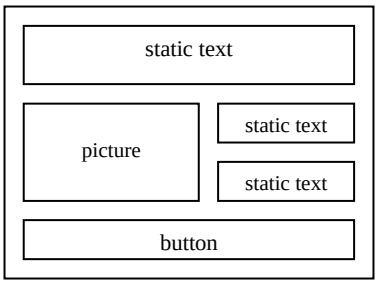
Gambar 6 Rancangan Tampilan Jendela Main Window



Gambar 7 Rancangan Tampilan Jendela Single URL



Gambar 8 Rancangan Tampilan Jendela Multiple URL



Gambar 9 Rancangan Tampilan Jendela *About*

Aplikasi *Wrapper* ini dibangun dengan menggunakan *Python* 2.4.3 sebagai bahasa pemrograman utama dan dengan *toolkit wxPython* 2.6 sebagai pembuat tampilan/GUI (*Graphical User Interface*).

Aplikasi Wrapper ini terdiri dari beberapa *frame* dan *module*, yaitu:

- **Wrapper.py**, merupakan program utama yang berfungsi sebagai eksekutor dari aplikasi *Wrapper*
- **mainMenu.py**, merupakan *frame* menu utama dari aplikasi *Wrapper*
- **SingleURL.py**, merupakan *frame* untuk *input* satu URL
- **MultipleURL.py**, merupakan *frame* untuk *input* dua URL
- **About.py**, merupakan *frame* keterangan tentang aplikasi *Wrapper*
- **wall.py**, merupakan *module* untuk tampilan *background* halaman utama

Wrapper.py

Wrapper.py merupakan program utama yang berfungsi sebagai eksekutor dari keseluruhan program. Pada Wrapper.py dilakukan *import frame* dan *module* yang akan digunakan pada program ini. Kemudian ditentukan *frame* mana yang akan dijadikan *frame* utama dari aplikasi ini. Pada Wrapper.py juga dideklarasikan *main class* dari keseluruhan program.

mainMenu.py

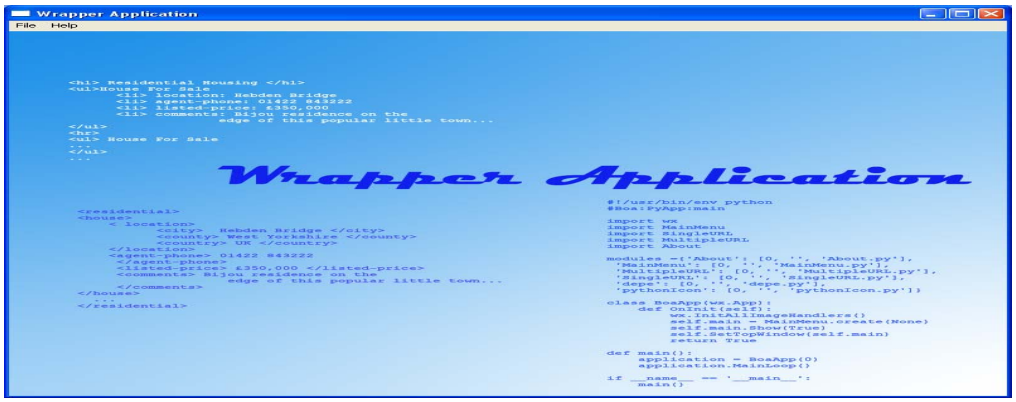
MainMenu.py merupakan *frame* utama dari aplikasi ini yang akan menghubungkan antar *frame*. Pertama-tama dideklarasikan terlebih dahulu *frame* **mainMenu** sebagai wx.Frame, dan juga ditentukan *properties*, seperti nama, posisi, dan ukuran dari *frame* **mainMenu**.

Kemudian dibuat menu dan komponen-komponennya yang akan menghubungkan antara *frame* utama dengan *frame* lainnya. Menu terdiri dari menu **File** dengan komponen **SingleURL**, **MultipleURL**, dan **Exit**, serta menu **Help** dengan komponen **Help** dan **About**.

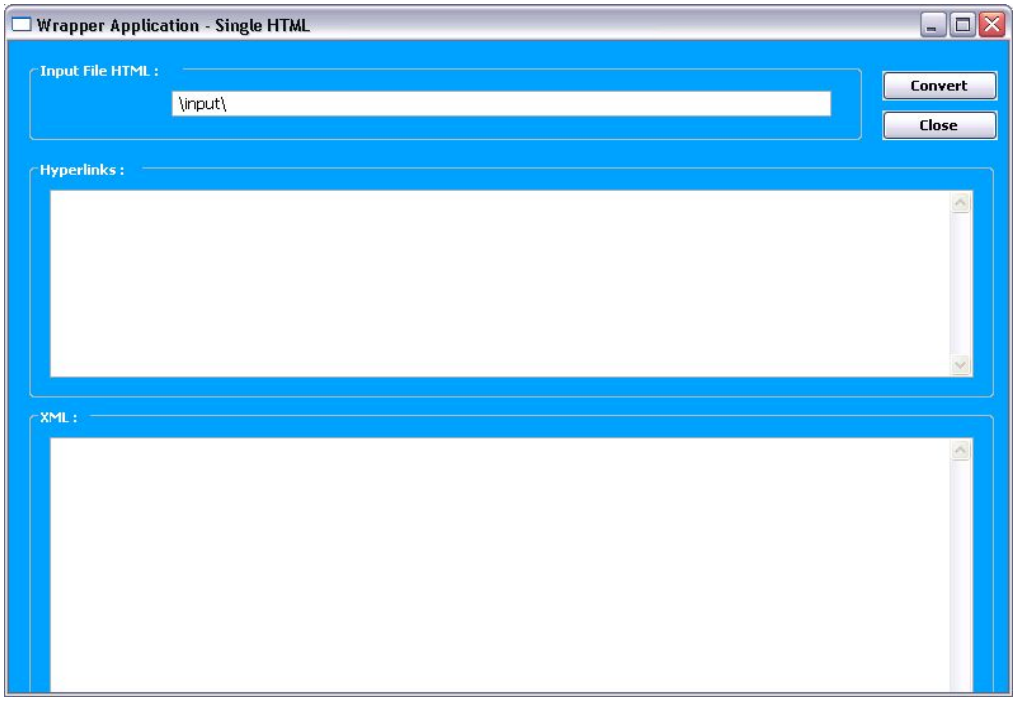
Untuk mengarahkan jalannya program, maka digunakan *event* untuk tiap-tiap menu yang telah dibuat. Kemudian juga dideklarasikan fungsi untuk masing-masing *event*. Khusus fungsi **help**, ia akan memanggil *class* lain, yaitu *class* **HelpMessage**.

Agar tampilan pada aplikasi ini lebih menarik, dimasukkan gambar sebagai latar belakang dari tampilan menu utama ini. Pemasukkan gambar dilakukan dengan melakukan pemanggilan terhadap *module* **wall.py** yang berisikan gambar yang telah di-*encode* menjadi bentuk yang dapat dibaca oleh *Python*. *Module* **wall.py** dibuat dengan menggunakan *module* yang tersedia dari *wxPython*, yaitu **encode_bitmaps.py**. Setelah didapat *module* **wall.py** dari hasil *encode* wall.jpg, selanjutnya dilakukan pemanggilan *module* **wall.py** kedalam program.

Tampilan Program



Gambar 11.mainMenu frame



Gambar 12 Tampilan untuk Masukan

SingleURL.py

SingleURL merupakan frame proses wrapping untuk input satu alamat URL. Pertama-tama dideklarasikan terlebih dahulu *frame* **SingleURL** sebagai wx.Frame, dan juga ditentukan *properties*, seperti nama, posisi, dan ukuran dari *frame* **SingleURL**.

Kemudian dibuat *object-object* dari *frame* **SingleURL** dan juga *properties* dari masing-masing *object*.

Langkah selanjutnya adalah memasukkan fungsi event ke tiap button, sehingga nanti apabila button ditekan akan melakukan aksi seperti yang dideklarasikan pada fungsi event.

Penutup

Aplikasi *Wrapper* ini digunakan untuk mengekstrak data dari suatu halaman *web* dan mengubahnya menjadi data yang lebih terstruktur, sehingga memudahkan dalam menentukan *class* dan *properties* yang akan membantu pengguna dalam memahami isi dari suatu halaman *web*.

Aplikasi ini dijalankan dengan memasukkan halaman *web* yang sudah disimpan terlebih dahulu, yang akan menghasilkan *output* berupa struktur XML serta *class* dan *properties* dari halaman *web* tersebut. Halaman *web* yang bisa dimasukkan paling banyak dua halaman *web*.

Aplikasi *Wrapper* ini masih memiliki banyak kekurangan, diantaranya pengguna tidak dapat memasukkan langsung suatu alamat web (*online*), melainkan harus melakukan penyimpanan halaman web terlebih dahulu. Diharapkan kedepannya dikembangkan Aplikasi *Wrapper* yang dapat berjalan secara *online*, sehingga akan lebih memudahkan pengguna dalam menggunakan aplikasi ini.

Daftar Pustaka

Baumgartner Robert, Gottlob George, Herzog Marcus, and Slany Wolfgang. *Annotating the Legacy Web with Lixto*. 2006.

Hogue Andrew and Karger David. *Wrapper Induction for End User Semantic Content Development*. Massachusetts Institute of Technology, Cambridge. 2004.

Hun Wei, Buttler David and Pu Calton. *Wrapping Web Data into XML*. In Proceedings of ACM SIGMOD Conference. September 2001.

J. Jansen Bernard, Ramadoss Raghavan, Zhang Mimi and Zang Nan. *Wrapper: An Application for Evaluating Exploratory Searching Outside of the Lab*. The Pennsylvania State University. 2004.

Mark Pilgrim. *Dive Into Python*, http://diveintopython.org, Maret 2007.

McCluskey Lee. *Information Extraction from the WWW using Machines Learning Techniques*. Department of Informatics, University of Huddersfield, 2005.

Ogbuji Uche. *Wrestling HTML*. http://www.xml.com/pub/a/2004/09/08/pyxml.html. Maret 2007. 2004.

Python. http://www.python.org. Januari 2007.

Python. http://www.vex.net. Januari 2007.

Python. http://www.pythonware.com. Januari 2007.

Schindler Christian, Arya Pranjal, Rath Andreas, and Slany Wolfgang. *htmlButler -Wrapper Usability Enhancement through Ontology Sharing and Large Scale Cooperation*, Graz University. 2006.

University of Southampton IT Innovation Centre. *Creating a Start Application Wrapper Script*. http://www.gria.org/docs/5.0.1/user-guide/service-provider-management/job/create_start_app_wrapper.html. Februari 2007. 2006.

Vercoustre Anne-Marie and Jabbour Sabine. *Wrapping Web Pages into XML Documents: A Practical Experience and Comparison of Two Tools*, CMIS Technical Report 01/199. TED Group, CMIS. Februari 2001.

W3C. *Reformulating HTML in XML*. http://www.w3.org/TR/1998/WD-html-in-xml-19981205. Maret 2007. 1998.

Wikipedia. http://en.wikipedia.org, Januari 2007.

Winwaed Software Technology LLC. *A Simple HTML Parser For Python*. http://www.winwaed.com/info/python_html/parser.shtml. Februari 2007. 2003.