Projecte Final de Carrera
Àrea de compiladors

Construcció d'un cercador amb llibreries de Web Scraping

12 Juny, 2011

Consultor: Jordi Ferrer Duran

Alumne: Manuel León Mondéjar

UOC Universitat Oberta de Catalunya

PFC - Compiladors

INDEX

Enunciat

- “Viatja en Temps de Crisi” ens ha encarregat un cercador “low cost”
- El cercador farà consultes a les webs d’operadors de vols i d’allotjaments en hotels
- Es retornen les 5 combinacions (hotel + vol) que en total resultin més econòmiques
- El cercador genera un arxiu XML de sortida del procés amb les dades de la cerca
- A la construcció del cercador s’han d’utilitzar tècniques de Web Scraping
- L’èxit del projecte està vinculat a la potència i qualitat del motor de cerca

Objectius

- Posar en pràctica tècniques de planificació i execució de projectes enfocant bona part dels esforços a la realització de la memòria
- Estudiar les tècniques de Web Scraping i aplicar-les al cas pràctic de l'enunciat
- Aprofundir en els coneixements específics de l'àrea del projecte com son els llenguatges de marques i les expressions regulars
- Aplicar a un cas pràctic tècniques i mètodes ja estudiats a part de les assignatures del pla d'estudis (Enginyeria Informàtica)

Enfocament

- Web Scraping (“raspat de la web”) es un procés d’enginyeria inversa on una eina automàtica extreu informació de llocs remots i la processa obtenint dades útils per als seus objectius.

- A aquest PFC s’utilitza Web Scraping per a mostrar combinacions econòmiques d’hotel i de vol a l’usuari que fa la cerca accedint amb el seu navegador a l’aplicació web

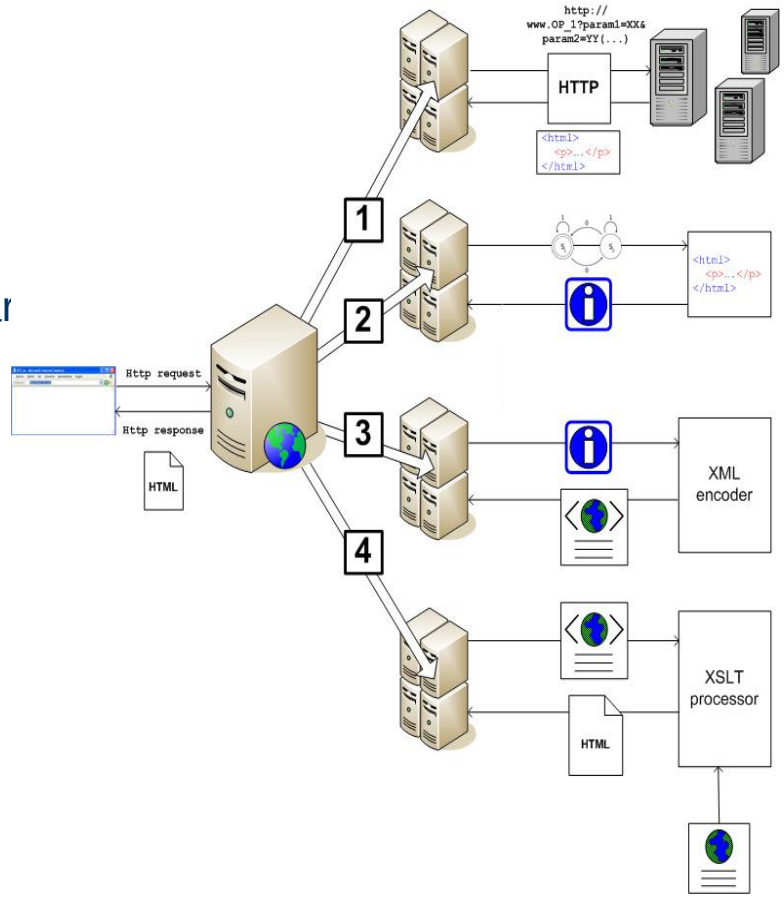
- Dins tot el procés, una de les tasques més importants es dóna al escollir els operadors als quals farem la consulta de preus. Aquests hauran de complir una sèrie de requisits segons el detall dels requeriments del cercador, però és clar que existeixen moltes possibilitats i que el nombre d’operadors haurà de estar limitat si bé un nombre elevat aporta valor afegit al cercador

A continuació es donen les diferents tasques en les quals es pot descompondre el funcionament del cercador.

Enfocament

Dins l'execució del cercador, podem descompondre el procés de cerca en les següents activitats:

1. Executar les connexions HTTP. Amb la darrera connexió amb cada operador s'obté el codi HTML amb els preus (en cas de trobar disponibilitat).
2. Extreure l'informació dels preus, noms d'hotels i altres dades de l'HTML. Muntar els objectes que s'hauràn d'ordenar segons el seu preu.
3. Crear el fitxer XML amb el resultat de la cerca.
4. Generar l'HTML de presentació amb el resultat de la cerca i que el back-end del sistema retorna a l'usuari. S'aplicarà una transformació XSL al codi HTML prenent l'XML generat anteriorment.



Consultes HTTP

- S'utilitza la llibreria d'Apache HttpClient (versió 4.1.1) per a fer les connexions HTTP contra els servidors dels operadors.
- HttpClient NO és l'emulació de cap navegador. El seu propòsit és transmetre i rebre missatges HTTP. No executa codi JavaScript, no dona format al codi, no inclou els arxius externs, etc.
- Existeixen eines per a conèixer les peticions HTTP junt amb els seus paràmetres, capçaleres, cookies, etc, executades durant la navegació a qualsevol lloc web. S'utilitzarà HttpFox per a conèixer la seqüència de peticions HTTP junt amb els paràmetres i tipus (Get, Post) que s'executen a les webs dels operadors escollits.
- Donant flexibilitat al cercador, no es faràn cerques conjuntes de vol i hotel. Es cerquen per separat els hotels i els vols. Posteriorment es calculen totes les combinacions possibles (hotel + vol) per cada hotel i vol tractats individualment.

Consultes HTTP

Els operadors triats son els següents:

HOTELS

Donant flexibilitat al motor, s'han cercat operadors d'hotels amb possibilitat d'indicar per a cada habitació el nombre d'adults i el nombre de nins. No s'accepten operadors amb combinacions ja restringides o sense possibilitat d'indicar el nombre d'ocupants per habitació

Amb excepció de l'operador skoosh, els preus no venen ordenats a la consulta. S'han aplicat dues solucions:
•Als operadors getaroom, Hotelopia, i lastminute: Una vegada tenim el resultat dels preus, establir altra crida HTTP que retorna els resultats ordenats per preu. Sol ser una nova URL o la mateixa indicant el camp d'ordenació (preu).
•A l'operador hoteles.com es llegueixen totes les dades, per a totes les pàgines de resultats (es captura la URL destí dels enllaços "Siguiente").

En general, els paràmetres de cerca (dades dinàmiques) son: Data d'inici, data de fi (o nombre de nits), ciutat de l'hotel (ciutat de destí del viatge) i distribució per habitació: nombre d'habitacions i, per cada habitació, nombre d'adults, nombre de nins i edat de cada nin

VOLS

En general, els paràmetres de cerca (dades dinàmiques) son: Data d'inici, data de fi (o nombre de nits), ciutat de origen, ciutat de destí, nombre d'adults, nombre de nins, nombre d'infants (nins fins a 1 any màxim).
Els operadors de vols retornen l'informació de preus ja ordenada.

Altres filtres:

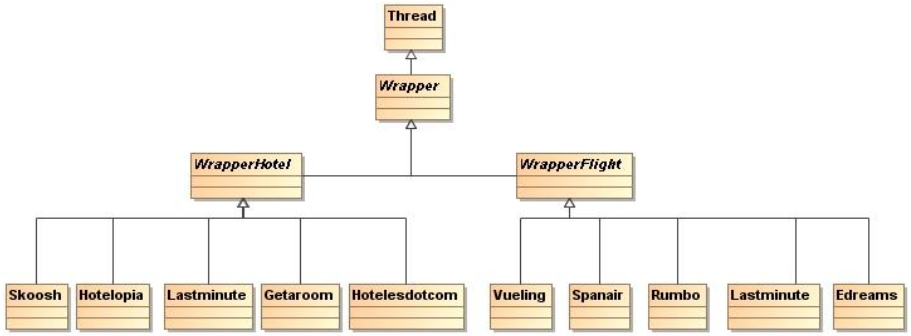
Ciutats configurades: Bona part de les ciutats Espanyoles que tenen aeroport

Moneda dels preus: €

Consultes HTTP

S'utilitza herència de classes per a definir les parts comuns de l'algorisme a tots els wrappers i parts específiques implementades per cada wrapper individualment. Un wrapper abstreu a l'aplicació de les especificitats de cada operador a l'hora de iniciar les transmissions HTTP i a l'hora d'extreure informació útil com son els preus. Per exemple, i a molt alt nivell:

```
public abstract class Wrapper extends
Thread {
public void run() {
   getScrapedData();
   getScrapedItems();
}
```




```
public class Hotelopia extends pfc.wrapper.hotel.WrapperHotel {
   protected void getScrapedData() {
      resultatScraping = httpClient.execute(new HttpGet(
            http://www.hotelopia.es),responseHandler,httpContext);
   }
   protected void getScrapedItems() {
      String preu = resultatScraping.substring(10,30);
   }
}
```

La part depenent dels operadors (connexions HTTP i anàlisi de dades) s'executa en processos concurrents: un per operador.
Aquests processos els inicia el fil d'execució del cercador que queda en espera a que finalitzin.
Es pot veure a l'exemple: "*extends Thread*"

Anàlisi de l'HTML

- L'informació extreta amb l'scraping de cada operador s'ha d'analitzar i s'han d'extreure les dades que el cercador necessita per a continuar. La dada més important és el preu, tant per als operadors d'hotels com per als operadors de vols.
- Per a l'extracció de les dades s'utilitzen expressions regulars, uns patrons de text que ens ajuden a definir el llenguatge que volem analitzar. El package *java.util.regex* de l'API de Java proporciona funcions útils per a treballar amb expressions regulars.

S'ha d'analitzar el codi HTML amb els resultats retornats per cada operador amb l'objectiu de definir les expressions regulars que encaixen amb l'informació que lleguim de cada operador.
Es important fer proves a aquesta part entre altres motius perquè la presentació dels resultats no té perquè ser sempre la mateixa per a tots els filtres de cerca. Per exemple, el preu d'una estància a un hotel pot estar a un *div* en cas de que a la consulta hi hagin nins o a un element *span* en cas contrari. De tota l'aplicació, aquesta és la part més canviant i depenent de l'operador.

Anàlisi de l'HTML

Dins la classe *pfc.util.Er*, es defineixen funcions comuns a tot el tractament amb expressions regulars i que son utilitzades internament al codi de cada classe Wrapper.

Un exemple és la funció següent que, en cas de coincidència de l'expressió 'er' amb la cadena 'txt', retorna el primer **grup de captura** trobat:

```
public static String getOptionalValueWithNL
(String er, String txt)
{
   Pattern p = Pattern.compile(er, Pattern.DOTALL);
   Matcher m = p.matcher(txt);
   if (m.find()) {
      for (int i=1;i<=m.groupCount();i++){
         if (m.group(i)!=null) return m.group(i);
      }
   }
    return null;
}
```

Dos comentaris importants amb relació a les expressions regulars:

•El metacaràcter "." coincideix amb qualsevol caràcter exceptuant el salt de línia. Si es vol donar com a coincident també aquest caràcter, a la creació de l'objecte patró (Pattern.compile(…) s'ha d'indicar el flag opcional Pattern.DOTALL. Es pot veure un exemple al codi anterior.

•S'utilitzen grups de captura per a definir grups dins les expressions regulars i poder recuperar el fragment de la cadena que coincideix només amb aquest grup. Per exemple, a l'expressió "Preu: (\d+) €" es defineix un sol grup el lexema coincident del qual es pot obtenir amb el mètode group de la classe matcher:

```
Matcher m = Pattern.compile("Preu: (\d+) €");
if (m.find()) return m.group(1);
```

A l'obtenció de les dades s'ha utilitzat la tècnica "divide & conquer": La cadena HTML resultat del procés anterior es divideix en fragments de resultats individuals (items), els quals son més sencills d'analitzar.

Les expressions regulars haurien de ser lo més sencilles possible, facilitant futures correccions i evitant una font de errades important.

Ordenació dels preus

- Es té un vector amb els items dels hotels i un altre vector amb els items dels vols.
- La classe ItemBean que representa un item (resultat) d'hotel o vol implementa l'interfície *java.lang.Comparable*. Amb l'implementació del mètode *compareTo* dins aquesta classe definim la regla de comparació entre items: un item es "menor" que altre si el seu preu també ho és.
- S'utilitza la classe java.util.Collections i el seu mètode estàtic "sort" per a ordenar una llista de items. La relació d'ordre entre items la defineix el mètode compareTo comentat abans:

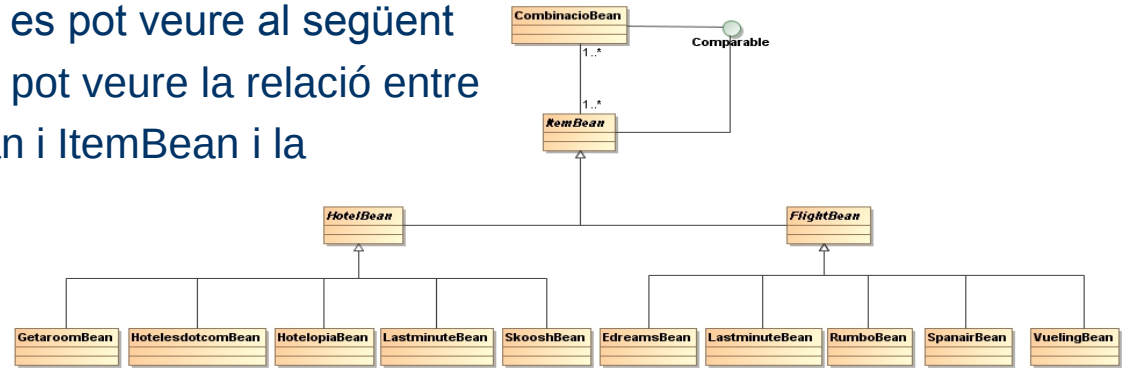
```
Vector<ItemBean> vVolBean = wrapper.getResultatScraping();
Collections.sort(vFinalHotelBean);
```

- Una vegada ordenats els items, es calculen totes les combinacions possibles on cada combinació és compon d'un item d'hotel i d'un item de vol. Aquesta és la classe pfc.bean.CombinacioBean i també implementa la interfície Comparable doncs el resultat final que es mostra son les cinc combinacions més econòmiques, i òbviament, necessitam ordenarles.

```
Vector<CombinacioBean> vCombinacions = new Vector<CombinacioBean>();
for (ItemBean hotelBean: vFinalHotelBean){
  for (ItemBean volBean: vFinalFlightBean)
    vCombinacions.add(new CombinacioBean(hotelBean, volBean));
}
Collections.sort(vCombinacions);
```

Ordenació dels preus

La jerarquia d'items definits es pot veure al següent diagrama de classes, on es pot veure la relació entre les classes CombinacioBean i ItemBean i la interfície Comparable.



Al diagrama es veu una classe descendent d'ItemBean per cada operador definit. Se fa aquesta diferenciació doncs tenim dades (atributs) que son diferents en funció de l'operador. Per exemple, d'un operador d'hotels podem extreure la categoria de l'hotel que altre operador no ens retorna. Aquesta jerarquia segueix una estructura molt similar a la que es donaba en la definició dels wrappers, ja tractat a l'apartat "Consultes HTTP"

A cada item s'implementa el mètode getPreu doncs el càlcul del preu total difereix en funció de l'operador. Per exemple, un operador d'hotels pot retornar el preu per nit i altre el preu total i mentre el primer haurà de multiplicar pel nombre de nits, a l'altre no li farà falta cap càlcul.

JAXB

JAXB (*Java API for Xml Binding*) és l'API que s'utilitza en la creació del fitxer XML

Aquesta API permet crear una jerarquia de classes partint d'un DTD o esquema de l'XML, crear els objectes d'aquestes classes a partir d'un XML amb les dades i serialitzar de nou els objectes a un fitxer XML.

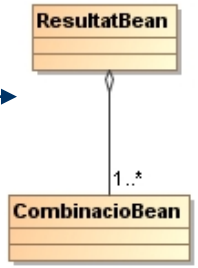
Ens centrarem a l'operació Marshal d'un objecte Marshaller creat. Aquesta operació rep un objecte ResultBean i genera el resultat (l'XML) a un buffer de caràcters:

```
JAXBContext contextObj = JAXBContext.newInstance(pfc.bean.ResultatBean.class,
 ItemBean.class,pfc.bean.hotel.HotelBean.class,pfc.bean.flight.FlightBean.class);
 Marshaller marshallerObj = contextObj.createMarshaller();
 marshallerObj.marshal(resultatBean, new StringWriter());
```

Aquest fitxer XML conté el resultat de la cerca i també un objecte amb totes les dades de la petició (dates, ciutats, habitacions, …) i queda emmagatzemat a la màquina en la qual s'executa l'aplicació (el path del directori dels arxius és configurable).

La classe ResultBean representa l'arrel del document XML i té l'informació que s'escriu a l'XML: un objecte amb les dades de la petició i un vector d'instàncies CombinacioBean.

Es pot veure la relació entre les classes ResultBean i CombinacioBean al diagrama annex →

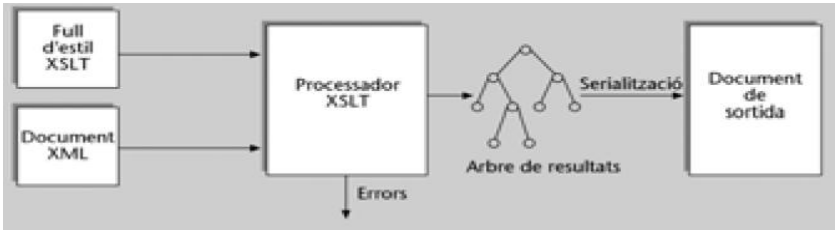


S'utilitzen anotacions per a guiar el procés de serialització dels objectes. En aquest sentit, l'única anotació obligatoria és la anotació *@XmlRootElement* que defineix la classe arrel del document XML:

```
@XmlRootElement(name="resultat")
public class ResultatBean {
}
```

JAXP

JAXP (*Java API for Xml Processing*) és l'API que s'utilitza en la creació de l'HTML. S'aplica una transformació XSL que rep el document XML generat i una plantilla XSLT. Aquesta transformació genera l'HTML final de presentació. S'ha de definir la plantilla XSLT amb totes les transformacions que s'apliquen als nodes del document XML.



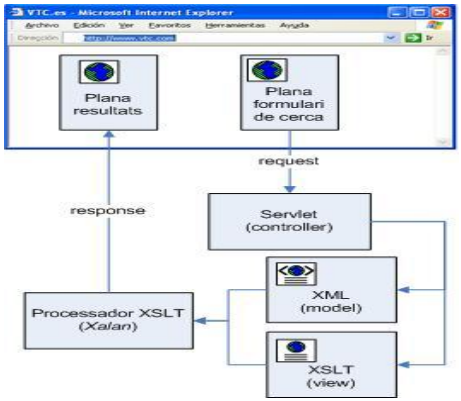
JAXP ens proporciona unes classes que ens serveixen d'interfície a l'invocació del processador XSLT, que és el programa que aplica la transformació XSL. S'utilitza el processador XSLT configurat per defecte, que és el processador Xalan (producte d'Apache). El codi de la transformació no és massa complex, és el següent:

```
StringReader oReader = new StringReader(xml);
StreamSource streamSource = new StreamSource(oReader);
StringWriter oWriter = new StringWriter();
StreamResult streamResult = new StreamResult(oWriter);
Transformer transformer = templatesXSLT.newTransformer();
transformer.transform(streamSource, streamResult);
```

Al codi anterior s'utiltiza la variable "templatesXSLT" que és una instància de la classe javax.xml.transform.Templates. Aquest objecte s'inicialitza una sola vegada, on es fa l'anàlisi sintàctic de la plantilla XSL. Es pretén estalviar en temps d'execució, doncs donat que la plantilla XSL que s'aplica és sempre la mateixa no cal fer l'anàlisi sintàctic amb cada transformació. D'altra banda, el temps d'execució de la transformació XSLT és petit, més encara en comparació amb altres accions més costoses en temps d'execució.

Interfície web

- Tot el desenvolupament fet fins ara es pot fer a una aplicació local d'escriptori i executar-se per exemple amb l'IDE Eclipse.
- S'utilitza el patró MVC on el controlador serà un servlet (ho implementa la classe *pfc.Pfc*) i la presentació seran les JSPs. El servlet rep les dades del formulari de cerca als paràmetres del request, executa el cercador, i determina la vista següent que pot ser la pàgina inicial (index.jsp) en cas d'error o la pàgina cerca.jsp en cas de que la execució hagi trobat resultats.
- El model de l'arquitectura es pot veure a la figura següent, on l'XML representa el model (el filtre i resultat de la cerca) i la plantilla XSLT la vista en relació a la presentació de resultats:



A la construcció de l'aplicació, es poden resumir les accions fetes als següents passos:

• Configuració del servlet (fitxer web.xml)
• Implementació del servlet (mètodes init, destroy i doPost que s'executa al executar una petició HTTP tipus POST a la direcció del servlet)
• Disseny del formulari de cerca (index.jsp)
• Adaptació dels paths dels arxius (els paths canvien quan l'aplicació es genera directament sobre el servidor d'aplicacions).

Instal·lació

- Tots els arxius de l'aplicació es comprimeixen a un únic arxiu amb extensió WAR (Web ARchive): **vtc.war**
- La generació del fitxer vtc.war es pot fer manualment amb l'eina JAR inclosa al JDK o es pot utilitzar altra eina, com Ant. Amb Ant (del projecte Apache) s'ha de crear un fitxer build.xml que conté les instruccións de compilació i generació necessàries per a la construcció del war.
- Aquest arxiu es pot copiar al directori deploy del servidor d'aplicacions instal·lat, que en aquest cas es JBoss.
- Tenint el JBoss arrencat, cada vegada que s'escriu el arxiu vtc.war al directori deploy, es detecta automàticament i s'inicia el procés de "undeploy" primer i de "deploy" després. Si es modifica un arxiu de l'aplicació per a poder veure els canvis, s'ha de generar de nou el war i fer el desplegament al servidor.

A qualsevol aplicació és important el tipus d'eina o mètode que utilitzem per a fer les traces de debug quan aquesta s'executa, fet que facilita molt la detecció de errades al sistema. S'ha utilitzat Apache Log4j per a implementar el logging de l'aplicació.
Log4j es parametritza al fitxer log4j.properties i queda inicialitzat al mètode init() de l'únic servlet configurat.

Productes

Durant tota l'assignatura del PFC s'han lliurat 3 entregues parcials i una final.

Les entregues parcials es corresponen amb les 3 PACs

A la primera PAC l'objectiu més important és la planificació del PFC.

Les PACs 2 i 3 representen entregues parcials del PFC on el contingut queda fixat segons la planificació feta a la PAC 1 i les dades d'entrega de les PAC

Finalment, el lliurament final es composa dels següents arxius:

- Producte amb tots els arxius necessaris per a executar el cercador (*mleonmo_producte.zip*)
- Memòria seguint un format específic i amb un límit de 90 planes (*mleonmo_memoria.zip*)
- Presentació: Document que sintetitza tot el treball realitzat (*mleonmo_presentacio.zip*)

La data límit del lliurament final és el **12 de Juny.**

Conclusions

- Durant el desenvolupament del PFC s'han estudiat diferents tècniques que es poden aplicar a un procés d'enginyeria inversa de planes HTML (Web Scraping)
- Ha estat important comprovar de forma periòdica que el progrés i estat del projecte s'ajusta a la planificació lliurada a la PAC 1.
- A part del seguiment del projecte, una tasca també periòdica és el desenvolupament de la memòria del projecte. La redacció d'aquest tipus de document forma part dels objectius del PFC.
- Podem observar que al igual que a la construcció d'un compilador es distingueix una part "front-end" independent de la plataforma i una part "back-end" depenent, al cercador s'observa també aquest fet. Tenim una part molt important depenent dels operadors externs (connexions HTTP i extracció de dades amb expressions regulars) i altra que no depén d'aquests operadors, com és la incialització del cercador, l'ordenació d'elements, la generació de l'XML o la transformació XSL.
- Resumint, s'ha aplicat a un cas complex un conjunt de tècniques aplicades a l'scraping d'informació present a servidors web. Informació que constitueix la base de dades del sistema, amb la particularitat que l'accés es fa remotament per HTTP.