

LINGUAGGI E TECNOLOGIE WEB

By Edoardo

ANALISI SINTATTICA E TRADIZIONE GUIDATA DALLA SINTASSI

PARSING

Linguaggio Formale = insieme di stringhe (anche infinito)

Stringa = sequenza di caratteri su un dato alfabeto (finita)

Alfabeto = insieme di caratteri (non vuoto)

Definizione di un linguaggio

MODALITÀ:

- + enumerazione
- + notazione inieustica
- + formalismi dedicati alla specifica dei linguaggi

Sintassi = aspetti formali del linguaggio

Semantica = significato associato alle frasi del linguaggio

Lessico = parole comuni (tokens), stabilisce le regole di costruzione dei tokens, mentre lo restante parte della sintassi stabilisce le regole di costruzione delle frasi del linguaggio (sequenze di tokens)

Un traduttore parla il linguaggio in due fasi:

1) Analisi INPUT

- lessicale
- sintattica
- semantica

2) SINTESI OUTPUT

Analizzatore lessicale = divide le sequenze di caratteri in parole tramite formalismi.

Classi linguaggi:

- tipo 0 o ricorsivamente不完備的

- tipo 1 o contestuali

- tipo 2 o non contestuali

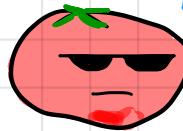
- tipo 3 o regolari

N.B. il nome deriva dalla grammatica su cui si basano

Data una sequenza di token, l'analizzatore sintattico verifica che la sequenza appartiene al linguaggio generato dalla grammatica G costituendone l'**albero sintattico**. In caso negativo restituisce un errore sintattico, altrimenti l'albero sintattico.

Sequenza (caratteri) \rightarrow [Analizzatore lessicale] \rightarrow Sequenza token \rightarrow [Analizzatore Sintattico] \rightarrow albero

$$L = \{ w \in W \mid w \in L((a|b)^*) \} \neq a^* b^*$$



Non è un linguaggio esprimibile con una grammatica non contestuale poiché **bilancia** le parti $w \rightarrow$ sono uguali.

L'analisi sintattica può essere fatta **top-down** (dalla radice alle foglie) o **bottom-up** (dalle foglie alla radice).

La prima corrisponde alla ricerca dei passi di **derivazione**, mentre la seconda i passi di **riduzione**.

Derivazione canonica sinistra = precedenza delle foglie di sinistra.

Riduzione canonica destra = inversa della derivazione **DESTRA**

HTML

tag = testo delimitato da "<" e ">"

HTML è composto da blocchi di testo e tag. I tag sono detti marcatura o comandi.

es.

<nome_tag> tag di apertura

</nome_tag> tag di chiusura

I tag marcano parti di testo:

- di formattazione, ovvero per cambiare l'aspetto (es. font)

- semanticci, per dare un significato particolare

Nelle marcature possono essere aggiunti degli attributi:

es.

Alcuni attributi sono obbligatori a seconda del tag

ELEMENTO: tag apertura + testo da marcare + tag chiusura

Dentro un elemento posso trovare ulteriori elementi

N.B. I tag sono cose insomisibili → =

Il problema di fondo è distinguere tra il vero contenuto del documento e la sua presentazione

INTESTAZIONE <head>

Contiene "meta dati" ovvero informazioni necessarie al browser per la corretta visualizzazione

es. <META name="author" content="Nome Cognome">

es. <TITLE> Nome Pagina </TITLE>

La scelta della visualizzazione dei metadati dipende esclusivamente dal browser: per esempio <title> viene visualizzato sulla barra della finestra, gli altri invece sono ignorati

N.B. Alcuni tag possono non essere chiusi poiché HTML non è troppo facile (vedi META) a differenza di altri che sono obbligati (vedi TITLE), ma comunque dipende dal browser
"Errori" in title non sono visualizzati poiché il browser visualizza tutto: al massimo potranno visualizzare elementi i modo diverso da come li avevo programmati

FORMATTAZIONE TESTO <body>

Ha alcuni attributi fondamentali come BACKGROUND, BGColor, TEXT, VLINK, ALINK e LINK

es. <body bgcolor="color">

I colori possono essere scritti in formato esadecimale (#ff0000) o a parole (solo alcuni colori principali in inglese es "red")

RGB

BACKGROUND differentiabile prende come valore un URL dell'immagine che vogliamo usare come sfondo

N.B. Gli attributi di body sono deprecati ufficialmente: usare alternative come XML

<div> allinea il documento nel tag es <DIV align="left">

<p> paragrafo

 break, a capo

Il tag div senza l'attributo "align" per il browser è un semplice paragrafo e viene chiamato tag NEUTRO

LINK

<A> Anchor

Attributo **HREF** = "URL"

Il testo nel tag anchor verrà associato alla url nell'attributo HREF

Attributo **NAME** = "#zona": specifica una post:colore zona del documento raggiungibile da un altro link es. miporto.it/#zona

Attributo **TARGET**: specifica in che modo visualizzare il link

es. TARGET = "_NEW" apre il link in una nuova finestra

Specifica un'immagine che verrà visualizzata

Attributi height e width (in pixel o percentuale) e ALT per associare un commento

Applet (JAVA) sono programmi Java esterni al documento HTML

Script codice scritto direttamente nel file HTML

Il primo è codice compilato, mentre gli script sono codice sorgente interpretato e facilmente modificabile.

TABELLE

- TABLE definisce la tabella

- TD dato

- TR riga

- TH attributi, nome colonna

- THEAD, TFOOT header e footer della tabella

IFRAME

<iframe src = "URL" width = "500" height = "500">

In un quadrato 500x500 verrà visualizzata la risorsa dell'url, anche se essa è una pagina web

FORM

Permette di gestire moduli compilati dall'utente

<form>

Attributo method = get || post

Nel tag form posso inserire altri elementi come INPUT (campi editabili, checkbox...), TEXTAREA e SELECT (menu a tendina)

ES. <INPUT TYPE = "TEXT"> → casella di testo

HTML DINAMICO

- LATO CLIENT (es. JavaScript) html dinamico

- LATO SERVER (es. PHP, ASP) html generato dinamicamente

Nel primo caso il documento è generato staticamente ma la sua esecuzione da parte del web client (browser) è dinamica ovvero gli script sono eseguiti quando richiesto

Nel secondo caso quando un web client interroga un server, il server genera l'html corrispondente alla richiesta.

Gli script lato client sono eseguiti senza filo nudo all'utente o quindi hanno poteri limitati (solitamente grafia)

Lato server ha compiti: un po più complessi come la gestione e elaborazione delle richieste del client

Navigator

Rappresenta l'istanza del browser in cui lo script è in esecuzione

- appCodeName codice identificativo del browser
- appName nome browser
- appVersion versione client

Window

Rappresenta l'istanza della finestra in cui il documento è visualizzato.

- title
- statusbar
- location: URL documento

JAVASCRIPT

FUNZIONI ANONIME

val p = `function (a,b) { ... }`

p diventa il nome della funzione.

Lo chiama alla funzione può essere fatta tramite la variabile p → p(x,y)
o usando direttamente il corpo della funzione tra parentesi: `(function (a,b){...})(x,y)`

Le funzioni hanno due proprietà:

- nome: nome funzione
- length: # parametri

e dei metodi per invocarle:

- call
- bind
- apply

Le funzioni possono essere passate come parametri per un'altra funzione
es. addEventListener("click", f) all'evento "click" esegue f → CALLBACK
Allo stesso modo possono essere anche usate come tipo di ritorno

CSS

I CSS si occupano di gestire la formattazione del documento esternamente allo stesso.

- In linea: nel body
- Incorporati: nell'intestazione
- Esterne: singolo file agisce su più documenti

1)

```
<DIV style = "stile o proprietà"></DIV>
```

Maschera non semantică: <DIV>,

Maschera con semantică: <p>

2) <style type = "text/css">

```
nome1 { proprietà }
```

```
nome2 { proprietà }
```

```
</style>
```

"nome1" è chiamato selettore e potrà essere usato nel body come tag

Al contrario del CSS in linea separa forma da contenuto, ma comunque non adatto per gestire più documenti (sitoweb)

3) Documento CSS separato e linkato nel documento

Poco formattore più file con un singolo foglio di stile

Nel header: <link rel = "stylesheet" href = "linkdoc.css" type = "text/css" >

class

. nomeclasse { ... } seleziona tutti gli elementi che hanno l'attributo class uguale a "nomeclasse"

Prima del punto inserire il nome del tag (selettore)

Pseudoclassi:

tag : nomeclasse { ... }

Selettore di ID.

elem { ... } seleziona il singolo elemento con attributo id = "elem"

Memorizzazione Persistente (HTML5)

- local storage (sopravvive alla sessione)
- session Storage (muore con la finestra del browser)

Entrambi sono oggetti a cui assegno elementi chiave - valore (stringa)

Ese. localStorage.nome = "Mario"

NB. Salvo oggetti "generici" JSON con la funzione `JSON.stringify(ogg.)` → `(toString())`

Il Local Storage ha un limite di 5MB per ogni dominio.

- `setItem(Key)`
- `getItem(Key, Value)`
- `removeItem(Key)`
- `clear()`

AJAX (ejax)

Consente l'interazione **asincrona** tra web-client e web-server: il client non si ferma ad attendere la risposta del server

Scommesso di dati in XML o JSON

- XML HTTP Request Object (xhr): oggetto che permette la comunicazione tra client e server

Utilizzato negli script Javascript.

N.B. `elem.onclick = func.`

`func(e)`

e sarà "onclick" ovvero l'evento che genera la funzione

e.target sarà elem ovvero l'oggetto sul quale si scatena l'evento

JQUERY

Libreria che velocizza la scrittura in Javascript
Formati:

development → non compresso
production → compresso

La libreria si importa esattamente come un normale script Javascript (anche da indirizzi esterni)

SINTASSI BASE: `$(selettore css).azione()`

L'oggetto su cui si applica l'"azione" dipende dal selettore. Il selettore identificherà elementi per nome, id, classe ecc.

Metodi:

`text`: legge/scrive il contenuto testuale dell'elemento

`html`: legge/scrive il contenuto html

`val`: legge il valore dello form

NB. Senza parametri legge, con parametri scrive

`css (proprietà)`: legge il valore della proprietà dell'elemento, se aggiungo un parametro setta a quel valore la proprietà.

GESTIONE EVENTI

`$(selettore).evento(function() { ... })`

ES.

`$("#button").click(f1)`

Spesso tutte le funzioni sono all'interno di un metodo

`$(document).ready(function() {`

`});`

Tale metodo specifica il caricamento completo della pagina web da parte del browser

Nelle funzioni JQuery è possibile richiamare funzioni Ajax

es. `$("#button").click(function() {
 $("#myDiv").load("xyz.html");
})`

Per gestire errori load permette di inserire una funzione di callback come secondo parametro

La funzione di callback avrà come parametri argomenti relativi all'oggetto xhr

XML

Problema: informazioni diverse con gli stessi tag → programmi non riescono ad interpretarli

Vengono usati "tag" con "nomi" appropriati al caso: la visualizzazione non concerne XML

È diventato un linguaggio per lo scambio di dati tra applicazioni

< nome-elem > ... < /nome-elem >

gli attributi funzionano come in HTML con la differenza che non assecondoci nulla di predefinito posso inventarli come i tag.

ES. < empleado dn = "1/1/1990">
 <nom> M. Romi </nom>
 </empleado>

ENTITÀ'

Porte di documento a cui è stato dato un nome

DOCUMENT TYPE DEFINITION

Un documento XML può avere un prologo che contiene un DTD.

Il DTD definisce la struttura del file XML dichiarando tipi di elementi, attributi ed entità.

<! OGG. ... >

le definizioni possono essere scritte direttamente nel file XML o importarlo da un file esterno

```
<!DOCTYPE DATA [  
    <!ELEMENT DATA [  
        name elemento  
    ]>  
    (GIORNO, MESE, ANNO)  
]> content Modul
```

Il content model può contenere altri elementi o #PCDATA che indica testo libero
Il content model usa gli stessi operatori delle espressioni regolari (es. |, +, ?, ...)

