ISAPI e ASP

Antonio Lioy
< lioy @ polito.it >

Politecnico di Torino
Dip. Automatica e Informatica

ISAPI

- **Internet Server API**
- **meccanismo proprietario di MS per creazione di pagine dinamiche tramite IIS:**
  - ogni applicazione ISAPI è una DLL
  - … caricata in memoria alla prima richiesta
  - … lasciata in memoria per soddisfare altre richieste
  - nello stesso spazio di memoria di IIS (comunicazione bidirezionale tramite specifici oggetti condivisi tra IIS ed applicazione ISAPI)
  - può essere tolta dalla memoria solo dal sistemista
- **l'applicazione ISAPI deve essere thread-safe**

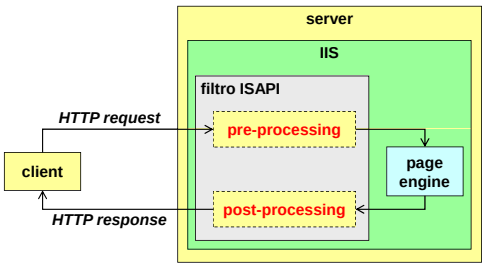
ISAPI

- **Internet Server API (ISAPI) è l'alternativa MS a CGI**
- **CGI crea un processo per ogni richiesta web**
  - consuma molte risorse (CPU e RAM) ed i processi hanno difficoltà a comunicare sia tra loro sia col server web
  - robusto (crash di un processo, non di tutto il server)
- **ISAPI ha prestazioni migliori perché:**
  - usa thread e meccanismi di sincronizzazione per sfruttare al meglio le risorse
  - lavora nello stesso spazio di memoria di IIS
  - rischio di blocco di tutto il server IIS

Applicazioni ISAPI: filtri ed estensioni

- **un filtro ISAPI agisce sul canale HTTP:**
  - può effettuare pre-processing della richiesta
  - può effettuare post-processing della risposta
  - es. compfilt.dll (compressione HTTP), md5filt.dll (HTTP digest authentication), sspifilt.dll (SSL)
- **un'estensione ISAPI è associata ad una pagina con una specifica estensione:**
  - elabora la pagina restituendo al motore HTTP il codice HTML risultante
  - es. asp.dll (pagine ASP), ssinc.dll (SSI)

Filtro ISAPI



Potenzialità dei filtri ISAPI

- **possono per esempio:**
  - reindirizzare la domanda per bilanciare il carico tra diversi server
  - aggiungere funzionalità di sicurezza / log
  - adattare la risposta alle capacità del client (versione di HTML e script supportata)

Configurazione di estensioni ISAPI

- **in base all'estensione della URL**
- **usare MMC per gestire una virtual directory di IIS**
- **in Properties / Home Directory / Application Settings / Configuration / Mappings è possibile associare:**
  - estensioni (es. ".asp")
  - applicazioni (es. asp.dll)
  - comandi HTTP accettati (es. GET, HEAD, POST)
- **possibile anche associare pagine web specifiche per vari errori applicativi**

ASP

- **Active Server Pages**
- **è un'estensione ISAPI (asp.dll, circa 300 KB) associato di default ai file con estensione ".asp"**
- **permette di inserire in una pagina HTML:**
  - degli script server-side in vari linguaggi interpretati (default: VBscript; possibile anche JS)
  - delle variabili di IIS
  - interazione con oggetti ASP built-in

ASP (Active Server Pages)

ASP è una tecnologia
(non un linguaggio di scripting)
proprietaria Microsoft
che permette l'interpretazione
degli script dal lato server

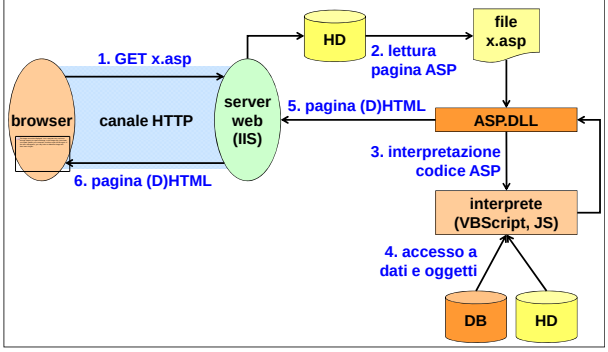
ASP

- **ASP è una tecnologia:**
  - messa a disposizione dal server Microsoft Internet Information Server (IIS)
  - di scripting lato server (server-side) per sviluppare applicazioni web dinamiche
- **una pagina ASP contiene degli script che vengono elaborati da un'estensione ISAPI del server web**
- **il risultato dell'elaborazione viene inviato al client**
- **ASP è indipendente dal linguaggio di scripting**

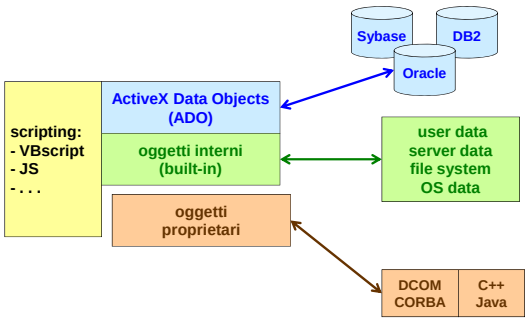
ASP: architettura

- **motore ASP:**
  - ASP.DLL
  - estensione ISAPI del web server che interpreta i file .asp
  - servizio multithread (estensione ISAPI)
- **file ASP:**
  - file di testo con estensione .asp
  - consiste in HTML standard e linguaggio di script racchiuso tra i caratteri speciali "<%" e "%>"

ASP: architettura



ASP: accesso ai dati



Linguaggi di script

- **IIS interpreta nativamente due linguaggi:**
  - JScript / JavaScript
  - VBScript (linguaggio di default)
- **possibile aggiungere PerlScript, Python, REXX ed altri**
- **per specificare l'interprete da usare:**

```
<%@ LANGUAGE="JavaScript" %>
```

```
<%@ LANGUAGE="VBScript" %>
```

Esempio di file ASP (con JS)

```
<html>
<head>
  <title>Saluti</title>
</head>
<body>
<%@ LANGUAGE="JavaScript" %>
<%
for (var i=1; i<=5; i++) {
  Response.write ("<h"+i+">Ciao!</h"+i+">");
}
%>
</body>
</html>
```

Risultato dell'elaborazione

```
<html>
<head>
  <title>Saluti</title>
</head>
<body>
<h1>Ciao!</h1>
<h2>Ciao!</h2>
<h3>Ciao!</h3>
<h4>Ciao!</h4>
<h5>Ciao!</h5>
</body>
</html>
```

parte generata dinamicamente (lato server)

JS: oggetto Enumerator

- **se si vuole ciclare su una Collection (oggetto MS) non si può usare il for-in**
- **necessario un oggetto Enumerator, che esiste solo su piattaforma MS:**
  - IE per client-side
  - IIS per server-side

```
e = new Enumerator(collection)
e.moveFirst();
while (!e.atEnd()) {
     Response.write(e.item());
     e.moveNext();
}
```

JS: oggetto Enumerator, metodi

- **atEnd( )**
  - ritorna un valore Booleano che indica se si è alla fine della collection
- **moveFirst( )**
  - aggiorna l'elemento corrente al primo elemento
- **moveNext( )**
  - sposta l'elemento corrente al successivo all'interno della collection
- **item( )**
  - ritorna l'elemento corrente

JS: esempio di Enumerator

```
var e = new Enumerator(Request.ServerVariables);
e.moveFirst();
while (!e.atEnd())
{
     Response.Write(e.item()+"<BR>");
     e.moveNext();
}
```

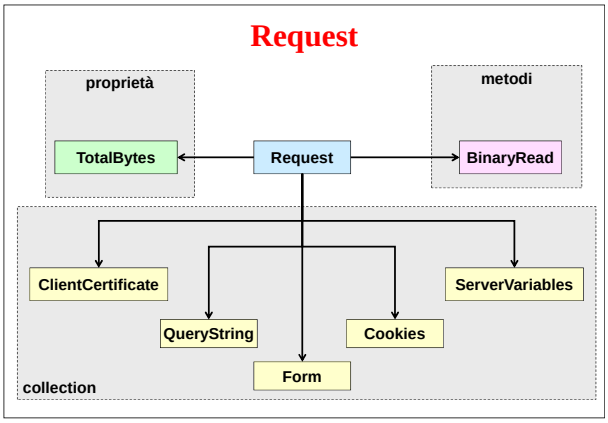
Oggetti interni ASP

- **sono oggetti che non devono essere istanziati**
- **oggetti interni:**
  - Request
  - Response
  - Application
  - Session
  - Server
- **sono oggetti ASP, disponibili in entrambi i linguaggi di script (Javascript e VBscript) ma purtroppo la documentazione di MS è quasi esclusivamente per VBscript …**

Request

- **gestisce le informazioni ricevute da un client:**
  - contenuto di un form inviato con la GET/POST
  - intestazione del protocollo HTTP
  - cookie (valori inviati dal browser)

Request



Request collection

- **ClientCertificate**
  - estrae i valori delle estensioni di un certificato digitale X.509 inviato dal client
- **QueryString**
  - estrae i valori dei parametri inviati mediante GET
- **Form**
  - estrae i valori dei parametri inviati mediante POST
- **Cookies**
  - estrae i valori dei cookie applicativi

```
<% user = Request.Cookies("username") %>
```

Request.QueryString: esempio

```
<form action="http://a.b.com/x.asp" method="get">
  <input type="text" name="nome">
  <input type="Submit">
</form>
```

x.html

```
GET /x.asp?nome=MARA HTTP/1.1
Host: a.b.com
```

x.asp

```
. . .
n = Request.QueryString("nome")
Response.write("Ciao "+n);
```

Ciao MARA

Request collection

- **ServerVariables**
  - estrae i valori delle variabili dell'intestazione del protocollo HTTP
  - i seguenti esempi restituiscono il modello del browser ed il nome DNS del server (come scritto nella URL)

```
<% b = Request.ServerVariables("HTTP_USER_AGENT")%>
```

```
<% serv = Request.ServerVariables("HTTP_HOST")%>
```

Server Variables: esempio

```
<table border=1>
<tr>
 <td><b>Server variable</b></td>
 <td><b>Value</b></td>
</tr>
<%
e = new Enumerator(Request.ServerVariables)
for ( ; !e.atEnd(); e.moveNext()) {
%>
<tr>
<td><%= e.item() %></td>
<td><%= Request.ServerVariables(e.item()) %></td>
</tr>
<% } %>
</table>
```

Request: proprietà

- **TotalBytes**
  - solo lettura
  - specifica il numero di byte che il client ha mandato nel body della richiesta

```
<% bytecount = Request.TotalBytes %>
```

Request: metodi

- **BinaryRead**
  - riceve i dati inviati dal client in una POST

Importante: parametri dei form in ASP

- **i campi dei form estratti lato server tramite Request.QueryString o Request.Form ...**
- **... non sono stringhe (come invece capita leggendoli in uno script lato client)**
- **... ma sono "oggetti ASP"**
- **dovrebbero essere convertiti automaticamente al tipo necessario per una certa operazione ma talvolta il meccanismo automatico fallisce ed il risultato non è quello desiderato**

(suggerimento) convertire sempre esplicitamente i campi dei form al tipo di oggetto desiderato

Parametri dei form in ASP: esempio

```
<form action="http://a.b.com/x.asp" method="get">
  <input type="text" name="nome">
  <input type="text" name="anni">
  <input type="Submit">
</form>
```

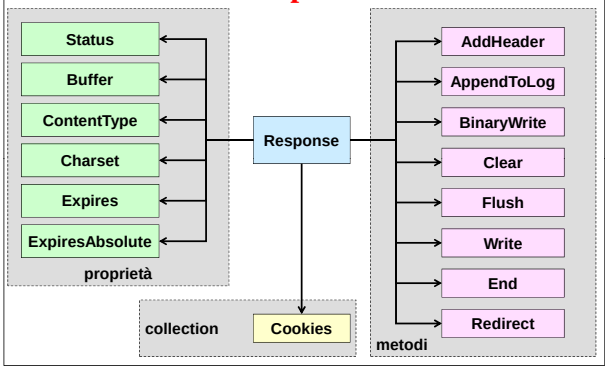


```
. . .
var n = String( Request.QueryString("nome") )
var a = Number( Request.QueryString("anni") )
. . .
```

Response

- invia informazioni al client
- configura i cookie mediante la collection Cookies

Response



Response: proprietà

- **Boolean Buffer**
  - se impostata a TRUE il server non invia output al client finché non viene terminata completamente l'elaborazione dello script
- **String ContentType**
  - configura il MIME tipe client (es. "text/html")
- **String Charset**
  - configura il charset della risposta (es. "iso-8859-1")

Response: proprietà

- **Int Expires**
  - configura il tempo in minuti di validità della pagina nella cache del client (default = 10)
- **Date ExpiresAbsolute**
  - configura in termini di tempo assoluti (ossia data e ora di scadenza) la validità della pagina nella cache del client
- **String Status**
  - configura lo status HTTP inviato dal server al client
  - deve contenere sia il codice numerico sia il commento (es. “401 Unauthorized”)

Response: metodi

- **AddHeader (String HeaderName, String HeaderValue)**
  - aggiunge un’intestazione HTTP
- **AppendToLog (String logText)**
  - aggiunge una riga al file di log del web server
- **BinaryWrite (Array Data)**
  - invia al client dati binari, utile ad esempio per inviare un immagine o un file Word
- **Clear**
  - pulisce l’output buffer

Response: metodi

- **End**
  - termina lo script
- **Flush**
  - invia al client il contenuto del buffer output
- **Redirect (String URl)**
  - redirige il client verso un URL
- **Write (data)**
  - scrive i dati nello stream HTML inviato al client
  - i dati non devono contenere “%>” da sostituirsi con “%\>”

Response: metodi

- **i seguenti due costrutti sono equivalenti**

```
<% Response.write("Ciao"); %>

<% ="Ciao" %>
```

Response: Cookies collection

- **per creare un cookie con un certo nome e valore:**
  - Response.Cookies("*cookiename*") = "*cookievalue*"
- **invece di creare cookie distinti, si possono inserire valori multipli in uno stesso cookie specificando delle "chiavi" (key) alla sua creazione:**
  - Response.Cookies("*cookiename*")("*key*") = "*keyval*"
  - le coppie chiavi:valore verranno inserite nel cookie usando la codifica urlencoded
- **le chiavi sono a loro volta una Collection**
- **per sapere se esistono chiavi usare la proprietà:**
  - HasKeys
    - (sola lettura) restituisce il numero di chiavi

Proprietà di un cookie in ASP

- **Expires = *vardate***
  - data e ora di scadenza del cookie
  - se non è impostata, è un cookie "volatile"
  - attenzione! impostare con getVarDate(*date*)
- **Secure = true | false**
  - tramesso solo su canali sicuri (SSL, TLS)
- **Path = *pathprefix***
  - trasmesso solo a pagine col prefisso indicato
- **Domain = *domain***
  - trasmesso solo a pagine nel dominio indicato
- **N.B. proprietà di un cookie, non delle singole chiavi**

Response: esempio impostazione cookie

```
var Nome = Request.Form("yourname");
var Cognome = Request.Form("yourfamilyname");

Response.Cookies("myapp")("nome") = Nome;
Response.Cookies("myapp")("cognome") = Cognome;

var expire = new Date();
expire.setMonth(expire.getMonth()+2);

Response.Cookies("myapp").Expires =
  expire.getVarDate();

Response.Cookies("myapp").Domain = "polito.it";
```

Request: esempio lettura cookie

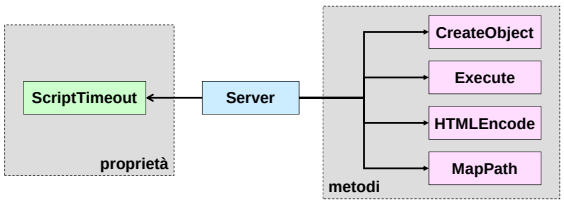
```
var c = new Enumerator(Request.Cookies)
for ( ; !c.atEnd(); c.moveNext()) {
  Response.write("<p>")
  if (!Request.Cookies(c.item()).hasKeys) {
    Response.write(c.item() + "=" +
    Request.Cookies(c.item()))
  }
  else {
    Response.write(c.item() + ":")
    var k = new Enumerator(Request.Cookies(c.item()))
    for ( ; !k.atEnd(); k.moveNext())
      Response.write(" " + k.item() + "=" +
      Request.Cookies(c.item())(k.item()))
  }
  Response.write("</p>")
}
```

Esempio: elenco dei parametri di un form

```
// indipendente da GET o POST
// elenca i nomi dei campi ed il loro valore

var m = Request.ServerVariables("REQUEST_METHOD")

if (m == "GET")
 var form_data = Request.QueryString
else // POST
  var form_data = Request.Form

var x = new Enumerator(form_data)
for ( ; !x.atEnd(); x.moveNext())
  Response.write(x.item()+"="+form_data(x)+"<br>")
```

Server

- **fornisce metodi e proprietà per accedere alle risorse del server**
- **usato per istanziare componenti**
  - i componenti sono pacchetti di oggetti



Server: Proprietà

- **Int ScriptTimeout**
  - definisce un timeout (in secondi) per l'esecuzione dello script

Server: metodi

- **Execute (String)**
  - esegue il file .asp che si trova in string (path relativo o assoluto; se assoluto lo script deve appartenere alla medesima application)
- **Component CreateObject (String)**
  - istanzia un componente (può essere un qualsiasi componente installato sul server, es. ActiveX)

```
<% MyAd =
new Server.CreateObject("MSWC.AdRotator"); %>
```

JS

Server: metodi

- **String HTMLEncode (String)**
  - codifica una stringa in HTML usando gli opportuni caratteri di escape (es. è)
- **String MapPath (String)**
  - mappa una virtual directory sulla directory fisica del server (importante per agganciare un file o un DB)
- **String URLEncode (String)**
  - codifica una stringa in modo appropriato per essere usata come una URL (es. %20 al posto degli spazi)

Oggetti sul server - esempio

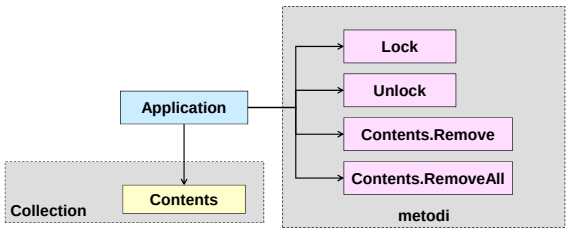
- **calcolo automatico data ultima modifica di un file:**

```
<%
var fso =
  Server.CreateObject("Scripting.FileSystemObject")
var file = fso.GetFile(
  Server.MapPath("avvisi.txt"))
var date = new Date(
  Date.parse(file.DateLastModified))

Response.write (
  "Document: " + file.name +
  " / Last update: " + date.toGMTString())
%>
```

Application

- **un'applicazione è un insieme configurabile dall'amministratore di risorse del server IIS**
- **per default c'è una sola applicazione che comprende tutte le pagine ASP**
- **oggetto condiviso da tutti gli utenti (=browser che si collegano ad una qualunque pagina ASP dell'applicazione)**
- **le informazioni perdurano fintanto che il server IIS rimane attivo**
- **usato per condividere informazioni tra diversi client che richiedono risorse appartenenti alla stessa applicazione**

Application



Application: collection

- **Contents**
  - collezione delle variabili di applicazione

```
<% Application("visitors") = 0 %>
```

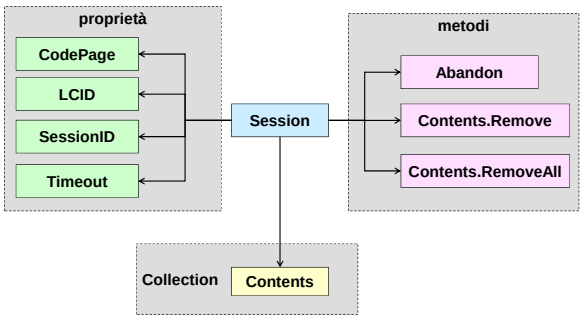
Application: metodi

- **Lock**
  - blocca la scrittura della collection (sincronizzazione)
- **Unlock**
  - sblocca la scrittura della collection
- **Contents.Remove (nome_variabile)**
  - cancella la variabile dalla collezione
- **Contents.RemoveAll**
  - cancella tutte le variabili della collezione

Session

- mantiene informazioni sulla sessione attiva di un singolo specifico client
- ogni client che si collega genera automaticamente un'istanza di un oggetto Session
- gestito tramite il cookie volatile ASPSESSIONID (un indice ai dati di sessione in RAM sul server IIS)

Session



Session: collection

- **Contents**
  - collezione delle variabili di sessione

```
<%  Session("nome") = "Antonio"  %>
```

Session: proprietà

- **SessionID**
  - contiene l'identificativo della sessione (uint32)
- **Int Timeout**
  - specifica un valore di tempo di inattività in minuti per la sessione (default: 10')
  - un valore troppo piccolo (es. minore di 4') fa perdere lo stato
  - un valore troppo grande (es. maggiore di 20') sovraccarica il server perché lo obbliga a tenere tante sessioni attive in memoria
  - impostarlo al tempo massimo che l'utente impiega per passare da una pagina all'altra

Session: metodi

- **Abandon**
  - abbatte la sessione (e quindi cancella tutti i Contents relativi)
- **Contents.Remove (nome_variabile)**
  - cancella la variabile dalla collezione della sessione
- **Contents.RemoveAll**
  - cancella tutte le variabili della collezione della sessione

File Global.asa

- **il file Global.asa contiene eventi legati alle applicazioni ed alle sessioni**
- **all'avvio di una nuova sessione il server lancia la procedura Session_OnStart**
- **alla chiusura di una sessione lancia la procedura Session_OnEnd**
- **all'avvio di un'applicazione (dopo il restart del server IIS) lancia la procedura Application_OnStart**
- **alla chiusura di un'applicazione lancia la procedura Application_OnEnd**

File Global.asa

```
<script language="JScript" runat="server">

function Application_OnStart(){
      Application("visitors")=0;
}
function Application_OnEnd(){
}
function Session_OnStart(){
      Application.Lock();
      Application("visitors")=Application("visitors")+1;
      Application.UnLock();
}
function Session_OnEnd(){
      Application.Lock();
      Application("visitors")=Application("visitors")-1;
      Application.UnLock();
}
</script>
```

JS

Direttive ASP @

- **@LANGUAGE**
  - imposta il linguaggio di scripting
- **@ENABLESESSIONSTATE**
  - impostare a False per disabilitare le sessioni ASP (per risparmiare tempo di esecuzione e memoria)
- **@CODEPAGE**
  - imposta la codepage di default (es. 65001 è UTF-8)
- **@LCID**
  - ID per impostazione locale (data, ora, valiuta, …)
- **@TRANSACTION**
  - imposta il tipo di transazione usato dallo script ASP

#include

- **ASP riconosce una sola direttiva SSI: #include**
- **file esterno incluso prima di passare la pagina all'interprete ASP: deve essere nella parte HTML ma può contenere sia HTML sia script ASP**
- **con tag "virtual" si usa pathname assoluto, con / equivalente alla radice del server web**
- **con tag "file" si usa pathname relativo a partire dalla cartella ove è presente il file con #include**
- **sintassi:**

```
<!--#include virtual="pathname_assoluto" -->
<!--#include file="pathname_relativo" -->
```

<script> in ASP

- **invece di usare <% e %> si può delimitare il codice ASP dicendo che si tratta di scripting:**
  - da eseguirsi lato server
  - con l'interprete per un certo linguaggio
- **IIS5 ha introdotto il parametro SRC per includere codice ASP da un file esterno**
  - MOLTO utile per includere funzioni JS esterne
  - http://support.microsoft.com/kb/224963
- **sintassi:**

```
<script language="javascript" runat="server"
   [ src="..." ] >
```

Esempio ASP (in JS)

form.html

```
<form method="post" action="e1.asp" name="F1">
Nome: <input type="text" name="nome"><br>
Email: <input type="text" name="email"><br>
<input type="submit" name="Submit" value="Invia">
</form>
```

e1.asp

```
<%@ LANGUAGE="JavaScript"%>
<%
if (Request.Form("nome")=="" || Request.Form("email")=="") {
  Response.Redirect("form.html");
}
else {
  Response.Write ("Nome: " + Request.Form("nome")
    + "<br> E-mail: " + Request.Form("email"));
}
%>
```

Riferimenti per ASP

```
http://msdn.microsoft.com/library/default.asp?
  url=/library/en-us/iissdk/iis/iis_web_pages.asp

http://aspjavascript.com

http://www.w3schools.com/asp/
(attenzione: è in VBscript)

http://www.comptechdoc.org/
independent/web/cgi/javamanual/
```