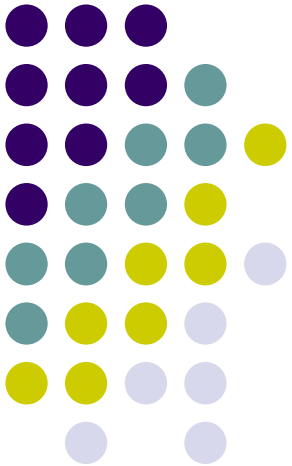
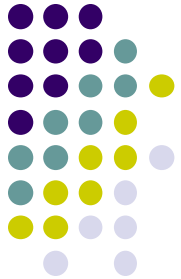
Мрежово програмиране

Сървърни разширения:
CGI, FastCGI,
ISAPI и сървлети



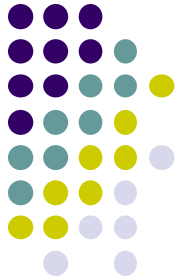


Сеансът на взаимодействието със сървъра за HTTP в най-общ вид се състои от следните стъпки:

1. Установяване на TCP съединение;
2. Заявка на клиента;
3. Отговор на сървъра;
4. Затваряне на TCP съединението.

Заявката на клиента представлява просто изискване за предаване на HTML документ или някакъв друг ресурс.

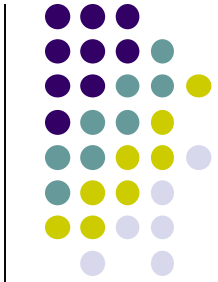
Отговорът на сървъра е кода на заявения ресурс.



Структура на клиентската заявка

- Ред на състоянието;
- Полета на заглавието;
- Празен ред;
- Тяло на заявката.

  Тяло на заявката в повечето случаи няма. Най-често тяло на заявката се използва в случаите, когато се налага да се предаде на сървъра информация, въведена от потребителя.



Редът на състоянието има следния формат:

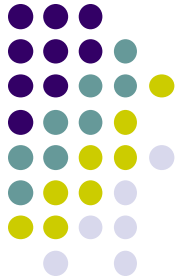
<метод на заявката><URL на ресурса>

<версия на HTTP протокола>

Методът на заявката определя вида на въздействието на ресурса, зададен с помощта на URL. Най-важни са два метода: GET и POST.

Версията на протокола обикновено се задава със следния формат:

HTTP/<версия>



Полетата на заглавието се използват за предаване към сървъра на допълнителна информация. Всяко поле на заглавието има следния формат:

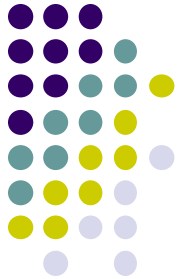
<име на полето>:<стойност>

Предназначение на някои по-често използвани полета на заглавието:

Host – доменно име или IP адрес на сървъра, към който се обръща клиента;

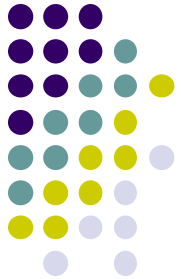
From – адрес на електронната поща на потребителя;

Accept – MIME –типове на данните, обработвани от клиента.



- Accept – Language - идентификатори, с помощта на които си обменят съобщения езиците, поддържани от клиента. Разделят се със запетаи;
- Accept – Charset – идентификатори, съобщаващи на сървъра за поддържаните от клиента кодировки. Разделят се със запетаи;
- Content – Type – MIME – тип на данните, съдържащи се в тялото на заявката;
- Content – Length – брой на символите, съдържащи се в тялото на заявката;
- Connection – управлява TCP съединението. Ако в това поле е зададена стойност Close, тогава след обработването на заявката съединението се затваря. Ако е зададена стойност Keep – Alive, тогава съединението се запазва и може да бъде използвано за следващи заявки;
- User – Agent – информация за клиента.

Отговор на сървъра



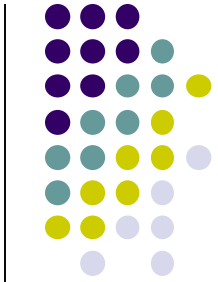
От гледна точка на уеб програмирането, структурата на отговора на сървъра е много по-важна, от структурата на заявката на клиента.

Изпълняваните на сървъра програми трябва да са способни самостоятелно да формират отговор за клиента.

Основните компоненти на отговора съдържат следните елементи:

- Ред на състоянието;
- Полета на заглавието;
- Празен ред;
- Тяло на отговора.

Ред на състоянието



Редът на състоянието има следния формат:

<версия на протокола> <код на отговора> <пояснения>

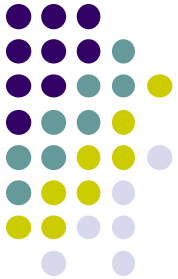
Версията на протокола се задава в същия формат, както в заявката на клиента;

Кодът на отговора представлява тризначно десетично число, обозначаващо резултата от обработката на заявката на клиента от сървъра;

Поясненията представляват разшифроване на кода на отговора в символен вид. Това е ред от символи, които не се обработват от клиента и е предназначен за системния администратор.

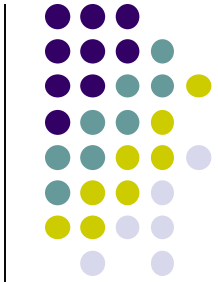
Кодовете на отговорите се разделят в пет групи. Групата, към която се отнася кода на отговора, се определя от старшия разряд на кода:

- 1 – информационно съобщение, означаващо, че сървърът продължава обработването на заявката на клиента (използва се рядко);
- 2 – съобщение за успешна обработка на заявката на клиента;
- 3 – съобщение за пренасочване на заявката;
- 4 – съобщение за грешка в заявката на клиента;
- 5 – съобщение за грешка на сървъра.

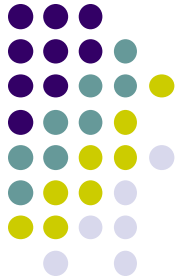


Полетата на заглавието в отговара на сървъра имат същата структура, като в заявката на клиента.

- Server – наименование и номер на версията на уеб сървъра;
- Allow – списък на методите, достъпни за дадения сървър;
- Content – Language – списък на езиците, които трябва да поддържа клиента за коректно изобразяване на предавания ресурс;
- Content –Type – MIME – тип на данните, съдържащи се в тялото на отговора на сървъра;
- Content – Length – размер на данните, съдържащи се в тялото на отговора на сървъра;



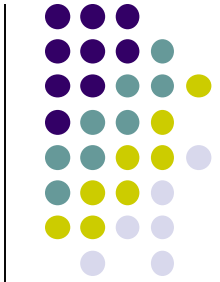
- Last – Modified – дата и време на последните изменения на заявения ресурс;
- Date – дата и време на създаване на отговора на сървъра;
- Expires – дата и време, определящи момента, когато информацията, предадена на клиента, се счита за остаряла;
- Location – адрес на реалното разположение на ресурса. Използва се за преадресация на заявката;
- Cache – Control – директиви за управление на кеширането.



В тялото на отговора се съдържа кода на предавания на клиента ресурс.

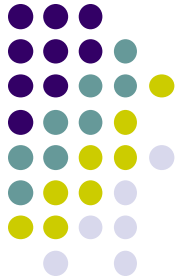
Това може да бъде HTML документ или всеки друг ресурс.

Начинът на обработването на ресурса се задава в полето на заглавието Content – type.



HTTP съобщенията се състоят от заявки на клиента към сървъра и отговори на сървъра за клиента:

HTTP-message=Simple-request |
Simple-Response |
Full-Request |
Full-Response

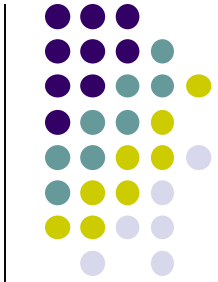


Простите заявки и простите отговори Simple-Request и Simple-Response не могат да прехвърлят информация в заглавията на HTTP съобщенията и се ограничават с използването на метода GET:

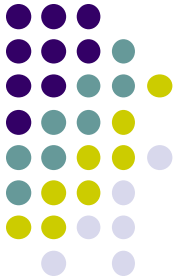
Simple-Request=@GET@ SP

Request-URL CRLF

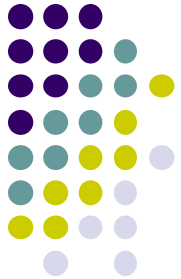
Simple-Response= [Entity-Body]



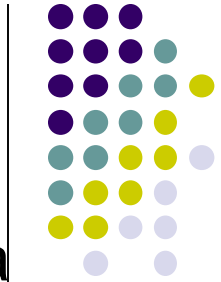
```
Full-Request =Request-Line
                    *( General-Header
                    | Request-Header
                    | Entity-Header )
                      CRLF
                    [Entity-Body]
Full-Response=Status-Line
                     *( General-Header
                      | Response-Header
                      | Entity-Header )
                        CRLF
                      [Entity-Body]
```



- За разширяването на възможностите на клиент-сървърното взаимодействие в рамките на HTTP протокол освен създаването на клиентската страна на разширения на стандартните възможности, предоставяни от хипертекстовите езици и от браузърите, могат също така да се разработват приложения, плъгини и скриптове на страната на уеб сървъра, разширяващи възможностите на самия уеб сървър



- До появата на CGI всички нови възможности са се реализирали във вид на модули, включени в библиотека с общи кодове на ЦЕРН.
- Разработчиците на сървърите е трябвало да използват тези кодове за реализация на програмите или да ги заменят със свои собствени аналози.
- Това означавало, че след компилацията на сървъра добавянето на нови функционалности е било невъзможно.
- CGI коренно променя тази практика.

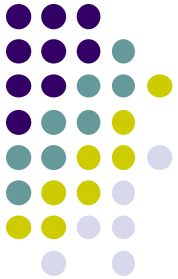


- *Common Gateway Interface* спецификацията е била разработена в Центъра за Суперкомпютърни Приложения в Университета на щата Илинойс (NCSA).
- От гледната точка на общата архитектура на програмното осигуряване на World Wide Web, CGI е определила цялото по-нататъшно развитие на системните средства.
- CGI е стартирал от работна група под ръководството на Ken Coar през 1997 г.

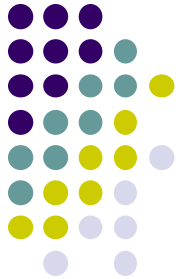
Спецификации за разширяване на функционалните възможности на сървъра:

1) CGI, ISAPI, Java – сървлети;
2) Скриптови езици: PHP, ASP, JSP.

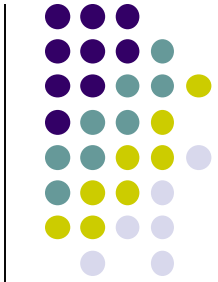
Достоинства на разработването на приложения на уеб сървърната страна във формата на скриптове:



- Понеже скриптовете се интерпретират (не се компилират) грешките в скриптовете ще генерират само диагностично съобщение и няма да доведат до дестабилизация на уеб сървъра или на операционната система
- Притежават добри изразителни средства
- Скриптовете като правило имат собствен проблемно-ориентиран набор от команди и един ред от скрипта може да свърши същото, което правят десетки редове на традиционен език за програмиране
- Като следствие на скриптов език могат да пишат програмисти с ниска квалификация
- Кросплатформеност

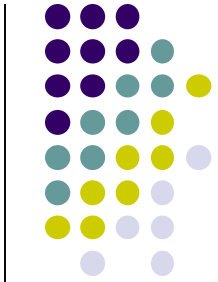


- CGI — стандарт на интерфейс, използван за връзка на външно приложение с уеб сървъра
- Приложението, използващо такъв интерфейс с уеб сървъра, е прието да се нарича шлюз, скрипт или CGI програма
- С помощта на този интерфейс за разработването на приложенията може да се използва всеки език за програмиране, който разполага със средства за взаимодействие със стандартния входящ и изходящ поток



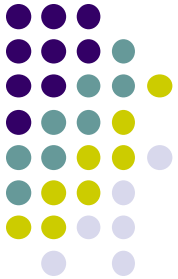
CGI не е език за програмиране, а множество от спецификации, които позволяват на уеб сървъра да комуникира с други приложения и програми

- CGI е прост протокол, с помощта на който уеб сървърът предава данни чрез stdin и stdout.
- CGI определя протокола за обмен на данни между сървъра и програмата.
- Затова CGI интерфейсът може да използва всяка сървърна програма, способна да работи със стандарните потоци за вход-изход.



Интерфейсът на CGI програмата не налага ограничения за избора на езика за програмиране. Последният трябва да притежава:

- *Средства за обработка на текст.* Необходими са за декодиране на входящата информация
- *Средства за достъп до променливите на обкръжението.* С тяхна помощ данните се предават на входа на CGI програмата
- *Възможност за взаимодействие с други програми.* Необходима е за достъп до СУБД, програми за обработка на графика и други

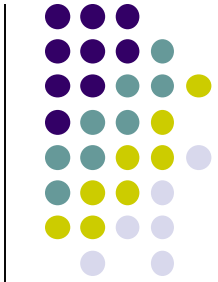


Всеки път когато уеб сървърът получи заявка от клиента, той анализира съдържимото на заявката и трябва да върне съответния отговор

- Ако заявката съдържа указание за файл, който се намира на твърдия диск, тогава сървърът връща в отговора си този файл
- Ако заявката съдържа указание за изпълнение на програма и необходимите за нея аргументи, тогава сървърът изпълнява тази програма и резултатът от нейната работа се връща на клиента

CGI определя:

- По какъв начин информацията за сървъра в заявката на клиента се предава на програмата във формата на аргументи и променливи на обкръжението
- По какъв начин програмата може да предава назад допълнителна информация за резултатите (например за типа на данните) във формата в заглавията на отговорите на сървъра



- Понеже скриптовете се интерпретират от изходния код динамично при всяко изпълнение, те се изпълняват обикновено значително по-бавно от готовите програми, транслирани в машинен код на етапа на компилацията. Скриптовите езици не се използват за написване на програми, изискващи оптималност и бързина на изпълнение. Поради простотата си те често се използват за написване на неголеми еднократни (проблемни) програми.
- Скриптовите езици според бързодействието си се делят на:

➢ *Езици с динамичен анализ (например sh, command.com)*
Интерпретаторът прочита инструкциите от файла на програмата на блокове и изпълнява тези блокове, без да чете кода по-нататък

➢ *Предварително компилируеми езици (например Python, Perl, Ruby)*
Първо се прочита цялата програма, след което се компилира или в машинен код или в един от вътрешните формати, след което полученият код се изпълнява. В последния случай резултатът от предкомпилацията е байт код

Common Gateway Interface

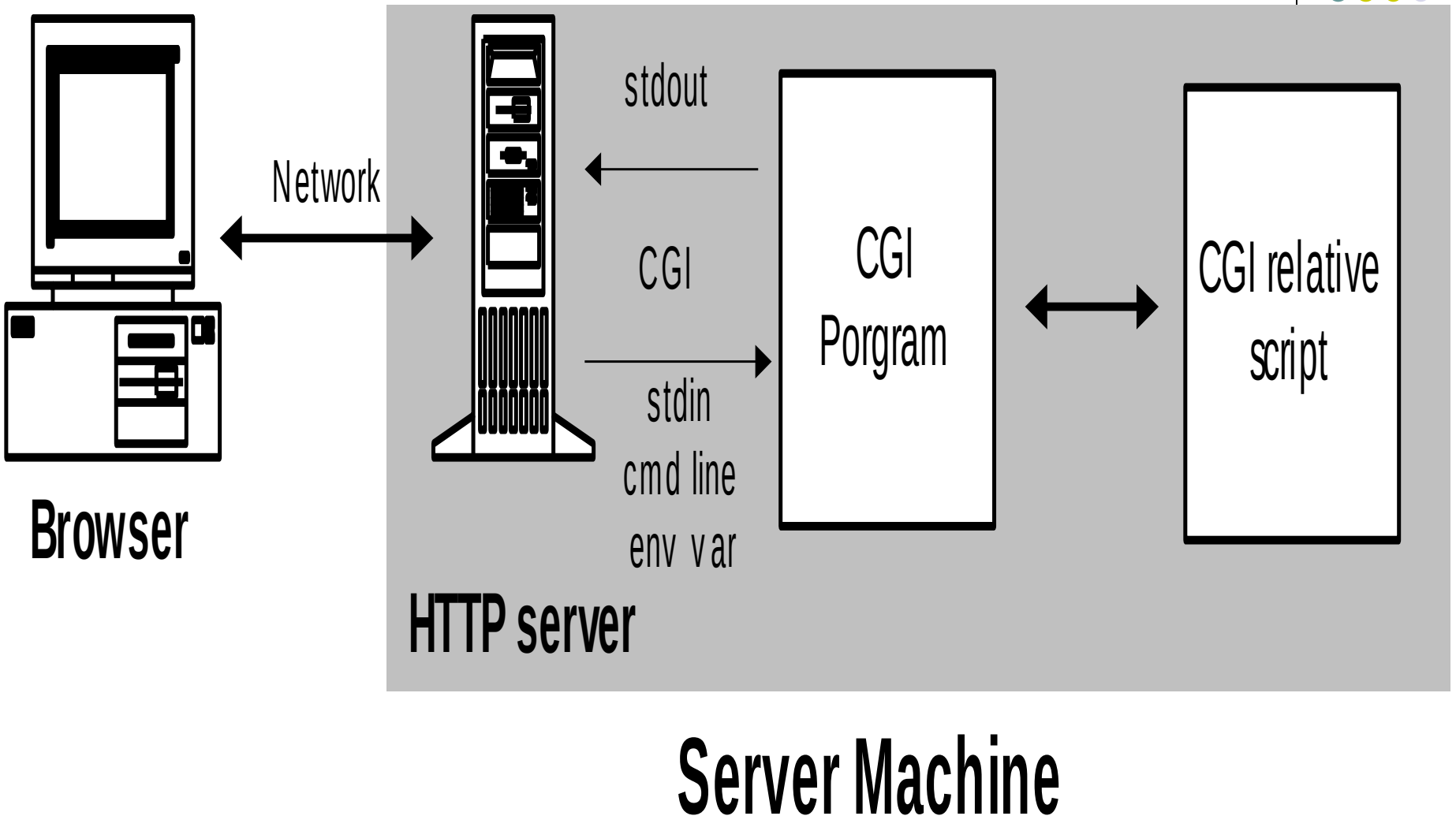
Основни моменти:

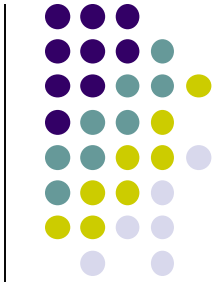
- понятие за CGI скрипт
- типове заявки
- механизъм за приемане на данни от скрипта
- механизъм за генериране на отговор на скрипта

Понятието CGI скрипт

CGI скрипт се нарича програмата, написана на някакъв език за програмиране или на команден език, която осъществява обмен на данни с HTTP сървър в съответствие със спецификацията Common Gateway Interface

Предназначението на CGI програмите е *да се създаде нов (динамичен, виртуален) HTML документ*, използвайки данните, съдържащи се в заявката и той да се върне обратно на клиента или да формира препратка към вече съществуващ документ и да предаде резултата на браузъра


Network
Browser
stdout
CGI
stdin
cmd line
env var
HTTP server
CGI
Porgram
CGI relative
script
Server Machine



Предаване на информация на CGI програма
Кодиране и прехвърляне на данните във формата

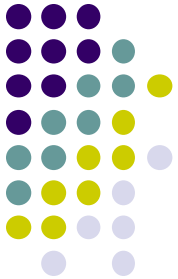
Съществуват два метода за кодиране на информацията, съдържаща се във формата:

- стандартен метод application/x-www-form-urllencoded, използван по премълчаване;
- допълнителен метод multipart/form-data.

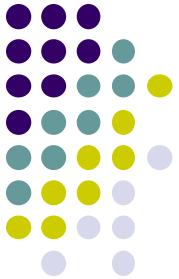
Вторият метод е необходим само в случай, че към съдържимото на формата се присъединява локален файл, избран с елемента от формата <INPUT TYPE=FILE>.

В останалите случаи се използва метода за кодиране по премълчаване.

Схема на кодиране



- За всеки елемент на формата, притежаващ име, зададено с атрибута NAME, се фомира двойката "name=value", където value е стойността на елемента, въведена от потребителя или зададена по премълчаване
- Всички двойки се обединяват в стринг, в качеството на разделител служи символа "&“. Кодираната информация се предава на сървъра с един от методите GET или POST
- При използване на метода GET данните на формата се предават на сървъра в състава на URL заявката

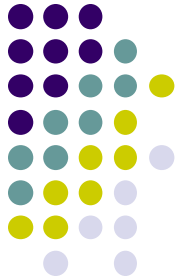


Типове заявки

Различават се два типа заявки към CGI скриптовете (изпращането на данните от формата):

- по метода GET
  - *Идемпотентен метод:* Резултатът на заявката може да бъде кеширан.
- по метода POST
  - *Неидемпотентен метод*. Ефектът от N идентични заявки е същия, както от една такава заявка.

HEAD и PUT са модификации на GET и POST



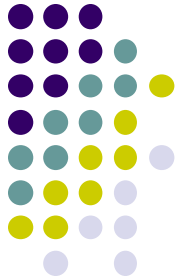
В заявките по метода GET данните от клиента се предават на скрипта с променливата на обкръжението QUERY_STRING.

```
GET /path/script.cgi?name=John+Doe&email=john%40example.com HTTP/1.0
```

В заявките по метода POST данните от скрипта се предават в потока на стандартния вход на скрипта. При предаването чрез потока на стандартния вход в променливата на обкръжението CONTENT_LENGHT задава броя на предаваните символи.

```
POST /path/script.cgi
Content-Type: application/x-www-form-urlencoded
Content-Length: 38

name=John+Doe&email=john%40example.com
```

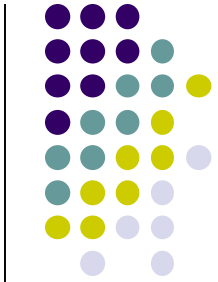
GET:



- За получаването на параметрите от заявката се използват променливите на средата на обкръжението
- Данните се изпращат като низ (query string)
- Клиентската заявка напълно се съдържа в URL, изпратен към сървъра
- Обемът на заявката обикновено е лимитиран (1KB)
- Тази информация се съхранява в реда на заявката
- Метод по подразбиране, когато активираме препратка с href HTML link

  ПРИМЕР: <a href="http://host.bg">

Метод GET



Заявките са според типа на кодирането: isindex и form-urlencoded

Заявка от тип ISINDEX:

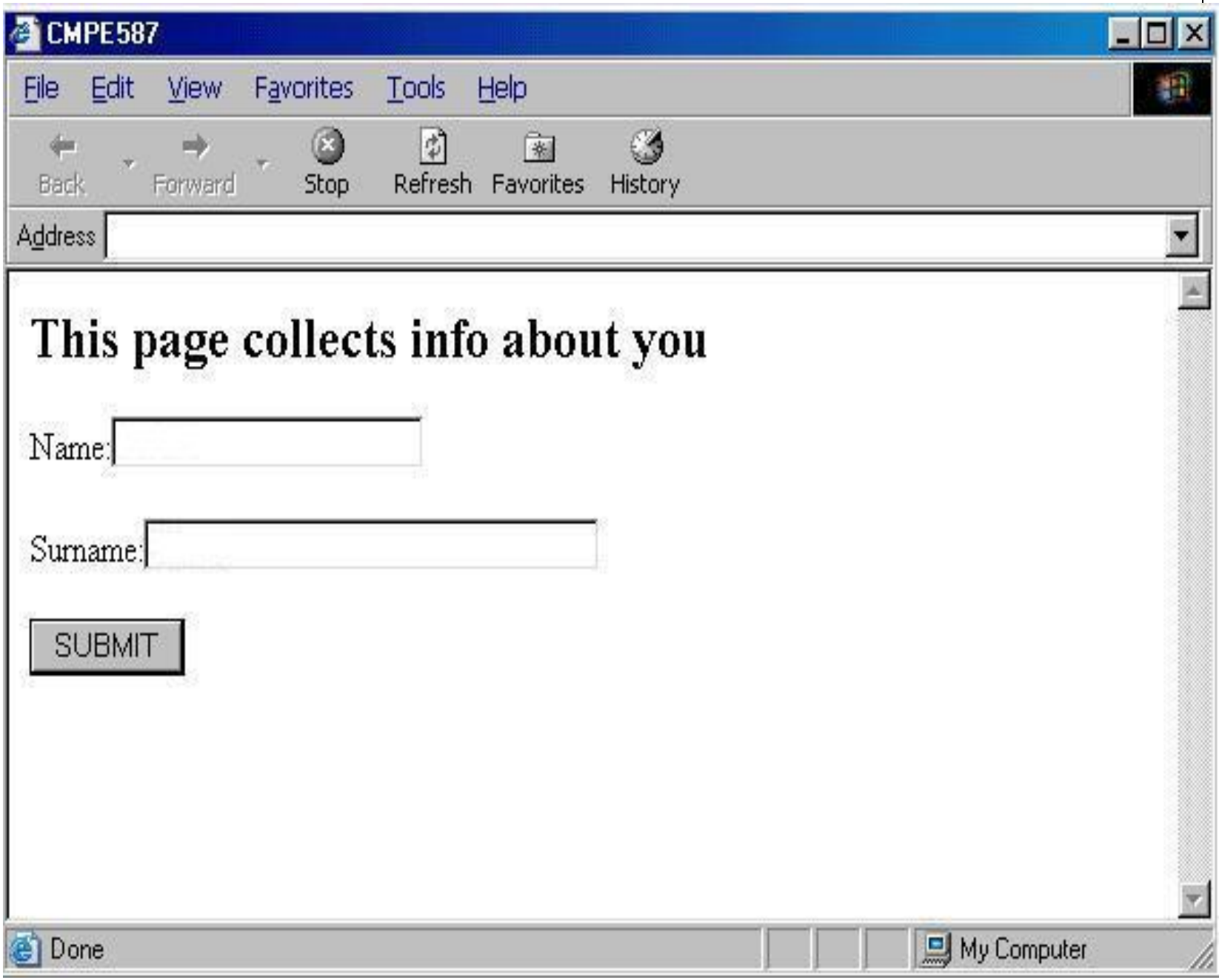
http://my_server.bg/somthig-cgi/

cgi-script?текст1+текст2+текст3

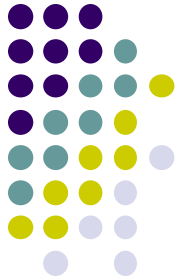
Заявка от тип form-urlencoded:
http://my_server.bg/somthig-cgi/
cgi-script?field1=word1&field2=word2

Пример с метода GET

```
<h2>This page collects info about you</h2>

<form
   action:“http://www.students.su.edu.bg/~is/cgi-
   bin/info.cgi”  method=“GET”>

Name:<input type=“text” name=“Name” size=20>

<p>

Surname:<input type=“text” name=“Surname”
   size=30><p>

<input type=“submit” value=“SUBMIT”>

Finish
```


CMPE587
File Edit View Favorites Tools Help
Back Forward Stop Refresh Favorites History
Address
This page collects info about you
Name:
Surname:
SUBMIT
Done
My Computer

Механизми за приемане на данни от скрипта



Скриптът може да приема данни от сървъра по три начина:

- чрез променливите на обкръжението
- чрез аргументите на командния ред
- чрез потока на стандартния вход

Променливи на обкръжението

В CGI спецификацията са определени 22 променливи на средата на обкръжението (мета променливи)

SERVER_SOFTWARE - определя името и версията на сървъра

SERVER_NAME - определя доменното име на сървъра

GATEWAY_INTERFACE - определя версията на интерфейса

SERVER_PROTOCOL - протокол на сървъра. CGI се е разработвал не само за използване в World Wide Web с протокола HTTP, но и за други протоколи, но широко приложение е получил само в WWW

SERVER_PORT- определя порта на TCP, по който се осъществява взаимодействието. За работа с HTTP се използва 80 порт, но той може да бъде преназначен при конфигурирането на сървъра

REQUEST_METHOD - определя метода на достъп до информационния ресурс. Това е важна променлива в CGI. Различните методи на достъп използват различни механизми за предаване на данни. Тази променлива може да приема стойности GET, POST, HEAD и др.

PATH_INFO - предава на програмата пътя, част от URL спецификацията, така както е зададена от клиента. Това означава, че се предава пътя (адреса на скрипта) във вида, посочен в HTML документа

пример: "/cgi-bin/search?nuclear+isotop"

PATH_TRANSLATED - същото както PATH_INFO, но едва след замяната от сървъра на определените в конфигурацията му съответствия. При конфигурирането на сървъра някои елементи (клони) на дървото на файловата система могат да се задават със синоними

пример: "/usr/local/etc/httpd/cgi-bin/test"

SCRIPT_NAME - определя адреса на скрипта така, както той е зададен от клиента. Ако не са посочени параметрите, тогава стойността на тази променлива ще съвпада с PATH_INFO, но ако променливите са посочени, тогава всичко, което следва след знака "?" ще бъде отхвърлено

пример: "/cgi-bin/search"

QUERY_STRING– променливата определя съдържанието на заявката към скрипта. Важна е при използването на метода за достъп GET. В QUERY_STRING се записва всичко, което е записано след символа "?"

пример: "nuclear+isotop“

REMOTE_HOST- доменен адрес на машината, от която се осъществява заявката

REMOTE_ADDR - IP адрес на запитващата машина

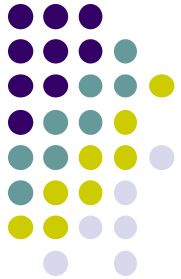
REMOTE_USER - използва се за идентификация на потребителя

CONTENT_TYPE - определя MIME типа данни, предавани на скрипта. Използвайки тази променлива може с един скрипт да се обработват различни формати данни

CONTENT_LENGTH- определя размера на данните в байтове, които се предават на скрипта. Тази променлива е важна при обмен на данни по метода POST, понеже няма друг механизъм за определянето на размера на данните, които трябва да се прочетат от стандартния вход

и други

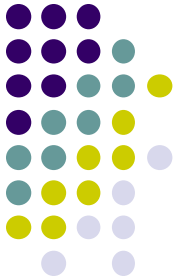
Аргументи на командния ред



Аргументите на командния ред се появяват само в заявките от тип ISINDEX

Получаването на достъп до съдържимото на командния ред на скрипта може да стане с помощта на същите функции, които се ползват при извикването му от интерактивната конструкция

Поток за стандартния вход stdin (С функции fread или scanf)



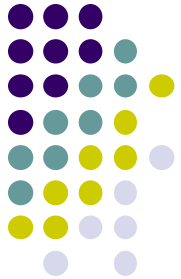
Въвеждането на данни в скрипта чрез потока за стандартен вход се реализира само при използването на метода за достъп до ресурса (скрипта) POST.

Пример: HTML формата трябва да се предаде заявка от типа: a=b&b=c

CONTENT_LENGTH = 7,

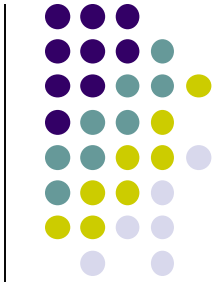
CONTENT_TYPE = application/x-www-form-urlencoded

POST:



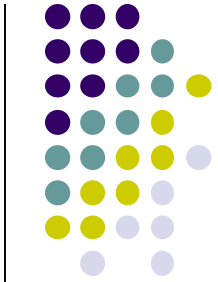
- Данните се изпращат като част от тялото на съобщението
- Данните не са видими в URL
- Може да се изпраща голям обем данни към сървъра
- Данните се предават чрез стандартния входящ поток

Механизъм за генериране на отговора от скрипта



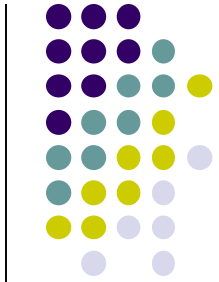
Скриптът реализира своя изход в стандартен поток за изход stdout

Този изход може да представлява документ, генериран от скрипта или инструкции към сървъра, къде да получи необходимия документ



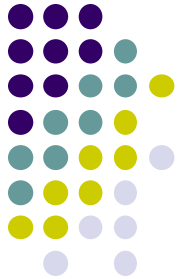
Обикновено в скриптовете се задават само три полета на HTTP заглавието:

- Content-type
- Location
- Status



Основното достоинство на технологията CGI е това, че се появява възможност динамично да се създават HTML документи.
Друго достоинство е, че тази технология се поддържа от повечето уеб сървъри.

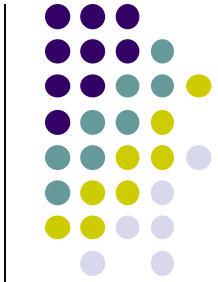
Основен недостатък на CGI е проблемът с увеличаването на натоварването на уеб сървъра. Всяка копие на CGI модул се зарежда на сървъра в отделен процес. Така при едновременно постъпване на заявки от 1000 потребителя ще бъдат стартирани едновременно 1000 копия на CGI програмата.



С цел да се реши този проблем е създаден стандартът FastCGI, който позволява да се избегне стартирането на излишни процеси, но той не е намерил широко приложение.

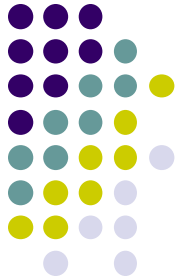
Също така за платформата MS Windows е разработен стандартът Windows CGI (WinCGI). Основната разлика от CGI е в това, че вместо променливи на обкръжението се използва INI файл.

Python използва технологията WSGI (Web Server Gateway Interface) – разновидност на FastCGI.

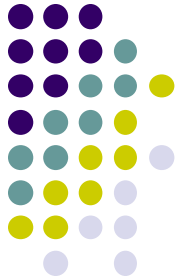


FastCGI сваля множество от ограниченията на CGI програмите.

Недостатъкът на CGI програмите е този, че те трябва да се презареждат на уеб сървъра за всяка заявка, което намалява производителността на системата.
FastCGI вместо да създава нови процеси за всяка нова заявка, той използва постоянно заредени процеси за обработка на множество от заявки.
Това позволява да се използва по-ефективно времето.

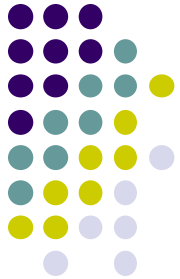


- CGI програмите взаимодействат със сървъра чрез STDIN и STDOUT на стартирания CGI процес, FastCGI процесите използват Unix Domain Sockets или TCP/IP за връзка със сървъра
- Това дава следните предимства пред обикновените CGI програми: FastCGI програмите могат да бъдат стартирани не само на този сървър, но също така и там, където е наложително в мрежата
- Също така е възможна обработка на заявките от няколко FastCGI процеса, работещи паралелно



Ако уеб сървърът е създаден на базата на Microsoft Internet Information Server, вместо CGI програми могат да се използват ISAPI приложения

- Въпреки, че ISAPI е отворена спецификация, и поддръжката на технологията е реализирана например в модула mod_isapi.dll за Apache for Win32, не съществува възможност да се прилага ISAPI на платформи, различни от MS Windows.

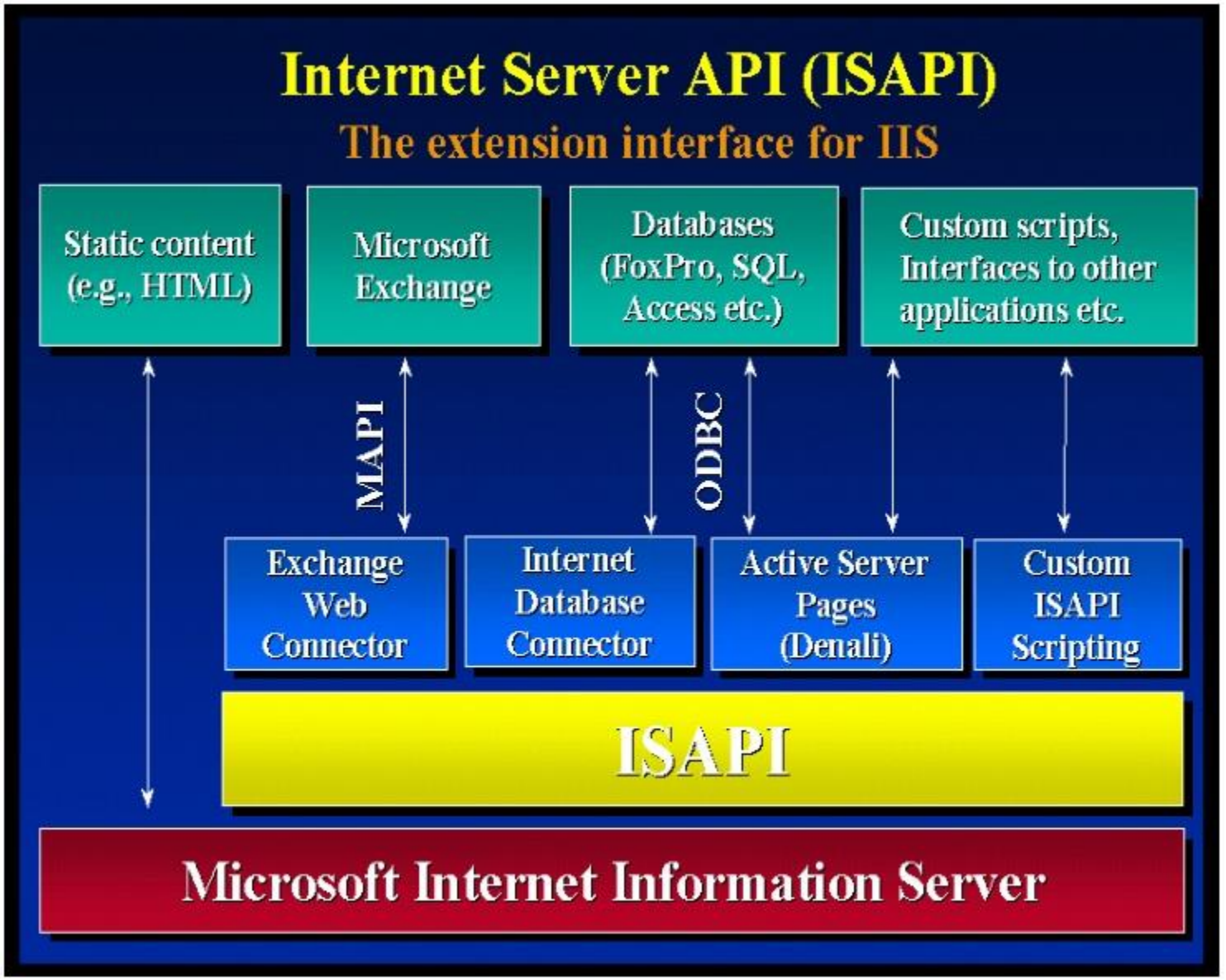


Цялото програмно осигуряване
на уеб приложенията на Microsoft пряко или косвено използва технологията ISAPI.

Технологиите Microsoft Application Server Pages (ASP) и .NET Framework са построени като приложения на ISAPI.

ISAPI е съвкупност от интерфейси, предоставяни от MS IIS уеб сървъра за създаване на приложения, взаимодействащи с този сървър и разширяващи възможностите му.

ISAPI приложенията представляват динамично свързвани библиотеки (Dynamic Link Library, DLL), директно взаимодействащи с API IIS.


Internet Server API (ISAPI)
The extension interface for IIS
Static content (e.g., HTML)
Microsoft Exchange
Databases (FoxPro, SQL, Access etc.)
Custom scripts, Interfaces to other applications etc.
MAPI
ODBC
Exchange Web Connector
Internet Database Connector
Active Server Pages (Denali)
Custom ISAPI Scripting
ISAPI
Microsoft Internet Information Server

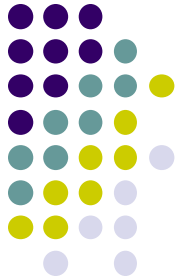
Internet Server Application Programming Interface

ISAPI приложенията се делят на:

- ISAPI разширения
- ISAPI филтри

Така цялото многообразие от разработвани ISAPI приложения се свежда само до тези два типа.

И филтрите и разширенията се компилират в DLL файлове, динамично зареждани от уеб сървъра.



Обръщението към тази библиотека се изпълнява в HTML документите, аналогично на обръщението към CGI програмите, от техните форми или препратки, създадени с таговете <FORM> и <A>

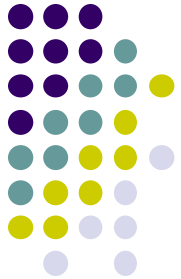
ISAPI разширението може да бъде извикано както явно:

- чрез заявка от вида http://<URL/path>/isapiext.dll?paramstring;

така и неявно:

- чрез карта на разширенията, в която са посочени обработчиците за регистрираните типове файлове (mapping).

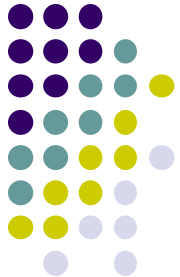
ISAPI приложения



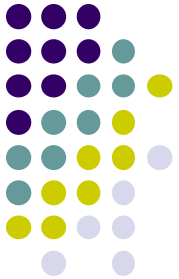
- ISAPI приложенията могат да се разработват с помощта на различни езици, поддържащи експорт на стандартните C - функции, например C, C++, Delphi

- За целта са налични ограничен брой библиотеки за разработването на ISAPI приложения, например Intraweb - компонентите Delphi Pascal, специални MFC - класове, специална C++ библиотека за сървърни технологии ATL

ISAPI разширения



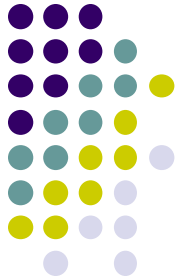
- ISAPI разширенията имат достъп до всички функционални възможности на IIS
- Реализират се във вид на DLL модули, зареждат се в пространството на процеса, контролиран от IIS
- Клиентите могат да се обръщат към ISAPI разширенията по същия начин както и към статичните HTML страници
- ISAPI разширенията могат да бъдат асоциирани с отделни разширения на файлове, с цели каталози или сайтове



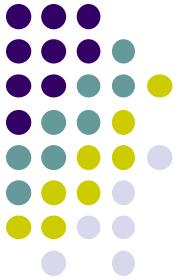
Съществуват реализации във вид на ISAPI разширения за такива инструментални средства като:

- ASP (Active Server Pages)
- ASP.NET
- ColdFusion
- Perl ISAPI (Perliist)
- PHP

ISAPI филтри

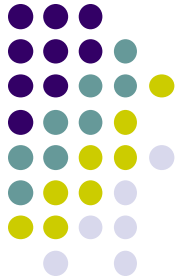


- ISAPI филтрите са необходими за изменение или усъвършенстване на функционалността на IIS
- Те обикновено работят с IIS сървъра и филтрират всяка заявка
- Филтрите се използват за анализ и модификация на входящия и изходящия поток от данни
- Филтрите също така се реализират във вид на DLL файлове



ISAPI филтрите обикновено се използват за решаването на следните задачи:

- Промяна в данните на заявката от клиента (URL или заглавието)
- Управление на отразяването на URL във физически файлове
- Управление на имената и паролите на потребителите при анонимна или базова автентификация
- Анализ и модификация на заявките след приключването на автентификацията
- Модификация на отговора на уеб сървъра
- Водене на журнали и анализ на трафика
- Реализация на собствена автентификация
- Управление на шифрирането и компресията

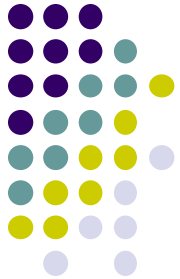


- Противоположно на CGI – ISAPI приложението се зарежда в същото адресно пространство, в което се зарежда и IIS уеб сървъра
- Това позволява да се увеличи производителността на приложенията
- Срив в ISAPI приложението може да доведе до нестабилност в работата на уеб сървъра
- В 6-тата версия на IIS вече има възможност за зареждане на приложенията в рамките на отделен процес

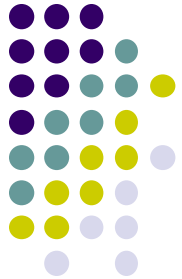
Основните недостатъци на разработването на уеб приложенията с помощта на ISAPI са:

- увеличаване на сроковете за разработване в сравнение със скриптовите езици, като например PHP и ASP
- недостатъчна поддръжка на такива стандартни особености на уеб приложенията като управление на сесиите
- понеже ISAPI разширенията се създават с използването на ненадеждни C-подобни езици също така расте рискът от препълване на буфера и за други уязвимости

Сървлети (Servlets) – решение за динамично генериране на HTML



- Сървлетите са написани на Java програми, които работят върху уеб сървър и комуникират използвайки уеб посредством HTTP и HTML
- Сървлетите са съвместими с всички уеб браузъри поради комуникация само посредством HTML и HTTP
- Сървлетите се програмират лесно поради това, че се използва стандартен Java
- По-голямата част от комуникацията се извършва от Servlet class (не е необходимо да работим със сокети, TCP/IP или Java serialisation).

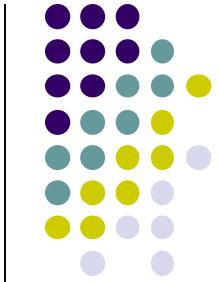


Servlet API доставя един общ модел на сървър (от там идва и наименованието им - малки, специализирани сървъри).

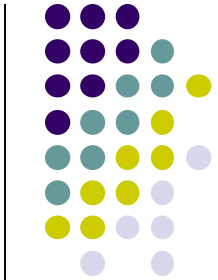
Обикновено всеки сървлет е предназначен за извършване на една услуга.

Servlet API представлява по-проста и по-дружелюбна среда за разработване.

Задачи на сървлета



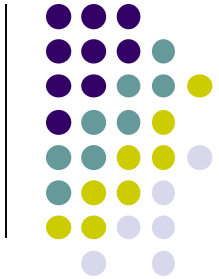
1. Четене на данните от формата, предавани явно от клиента
2. Четене на неявните данни – заглавията на HTTP протокола
3. Генериране на резултата
4. Изпращане на клиента на явни данни във вид на HTML
5. Изпращане на неявни данни, като статуси и заглавия на HTTP протокол



- Java сървлетите са:
  - Технология за генериране на динамични уеб страници (както PHP, ASP, ASP.NET, ...)
  - Независими от платформата и протокола сървърни компоненти, написани на Java, които разширяват стандартните уеб сървъри
  - Java програми, обработващи HTTP заявки
- Класът **`HttpServlet`**
  - Предоставя динамично генериране на съдържание (HTML, XML, …)

- Сървлети
  - Предоставят общ framework за услуги базирани на модела заявка-отговор
  - Преносими на всеки Java приложен сървър
  - Имат достъп до цялата фамилия от Java и Java EE API интерфейси
    - JDBC, Persistence, EJB, JMS, JAX-WS, JTA, JTS, RMI, JNDI, JAXP, ...
  - Основна част от всички Java web application технологии (JSP, JSF, ...)

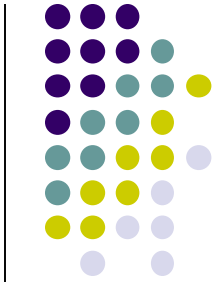
Извеждане на прост HTTP отговор



```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```

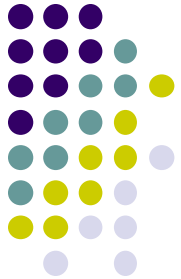
URL assumes you have deployed from an Eclipse project named "intro". Code was in src/HelloWorld.java

HTTP отговор във вид на HTML



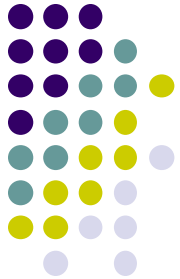
```
public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 "+
      "Transitional//EN\">\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Hello</TITLE></HEAD>\n"+
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>Hello</H1>\n" +
                "</BODY></HTML>");
  }
}
```

FormServlet2.java (using POST)

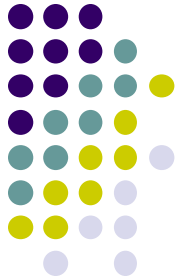


```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FormServlet2 extends HttpServlet
{
    public void doPost(HttpServletRequest req, HttpServletResponse res)
       throws ServletException, IOException
    {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<HTML><HEAD><TITLE>Form
    Example</TITLE></HEAD>");
            out.println("<BODY><H1>Hello " + req.getParameter("firstname") +
    " "
                        + req.getParameter("surname") + "</H1>");
            out.println("You are " + req.getParameter("age") + " years old!");
            out.println("</BODY></HTML>");
            out.close();
    }
}
```

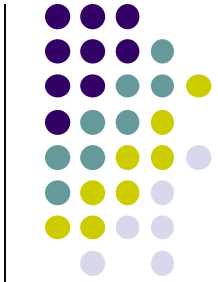
Архитектура на сървлетите



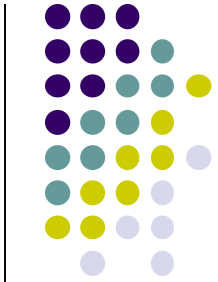
- Класът **HttpServlet**
  - Обслужва HTTP заявки от клиента
- За всеки от HTTP методите GET, POST и други е дефиниран съответен метод:
  - **doGet(…)** – обработва HTTP GET заявки
  - **doPost(…)** – обработва HTTP POST заявки
  - **doPut(…), doHead(…), doDelete(…), doTrace(…), doOptions(…)**
- Сървлетът обикновено предефинира един от първите два метода или метода **service(…)**



- Обектът **HttpServletRequest**
  - Съдържа данните от заявката на клиента
    - Заглавните части на HTTP заявката
    - Данни от форми и параметри на заявката
    - Други данни от клиента (бисквитки, път и т.н.)
- Обектът **HttpServletResponse**
  - Капсулира данни изпращани обратно на клиента
    - Заглавни части на HTTP отговора (content type, бисквитки и други)
    - Тяло на отговора (като **OutputStream**)

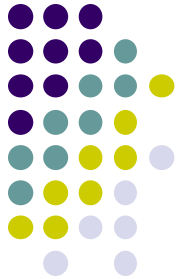


- HTTP GET методът се използва когато:
  - Обработката на заявката не променя състоянието на сървъра
  - Количеството данни от формата е малко
  - Искаме да позволим добавяне на заявката в bookmarks
- HTTP POST методът се използва когато:
  - Обработката на заявката променя състоянието на сървъра, например запис на данни в база от данни
  - Количеството данни от формата е голямо
  - Съдържанието на данните не трябва да е видимо в URL (например пароли)

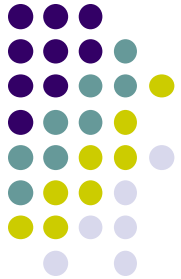


- init
  - изпълнява се еднократно при зареждането на сървлета (не се извиква за всяка заявка).
- service
  - извиква се от сървъра за всяка заявка от нов поток, като на свой ред предава извикването към doGet, doPost и други;
  - този метод не следва да се преопределя.
- doGet, doPost, doXxx
  - обработва HTTP заявки GET, POST и други;
  - следва тези методи да се преопределят с цел задаване на функционалност на сървлета.
- destroy
  - извиква се при унищожаване на екземпляра на класа на сървлета;
  - не се извиква след всяка заявка.

doGet и *doPost* имат идентични сигнатури и получават от service метода два параметъра

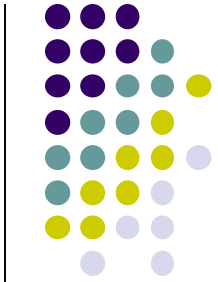


- ***public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException***
- ***public void doPut(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException***



HttpServletRequest е обект, съдържащ информация за заявка от уеб браузър, която може да бъде:

- login на потребителя и някакви стойности от заглавната част, изпратена от уеб браузъра.
- Някакви параметри, изпратени като част на заявката
- Някакви информационни параметри, поставени от deployment descriptor
- Доставя сесии (session) и бисквитки (cookies)



HttpServletResponse е обект, който позволява

- Добавяне на cookies
- Добавяне на Header стойности
- Пренасочване към нови URL
- Създаване на HTTP за уеб браузър
- Изпращане на съобщения за грешки

Обработка на данните от HTML формата

```
<FORM ACTION="./servlet/coreservlets.ThreeParams">
  First Parameter:  <INPUT TYPE="TEXT" NAME="param1"><BR>
  Second Parameter: <INPUT TYPE="TEXT" NAME="param2"><BR>
  Third Parameter:  <INPUT TYPE="TEXT" NAME="param3"><BR>
  <CENTER><INPUT TYPE="SUBMIT"></CENTER>
</FORM>
```