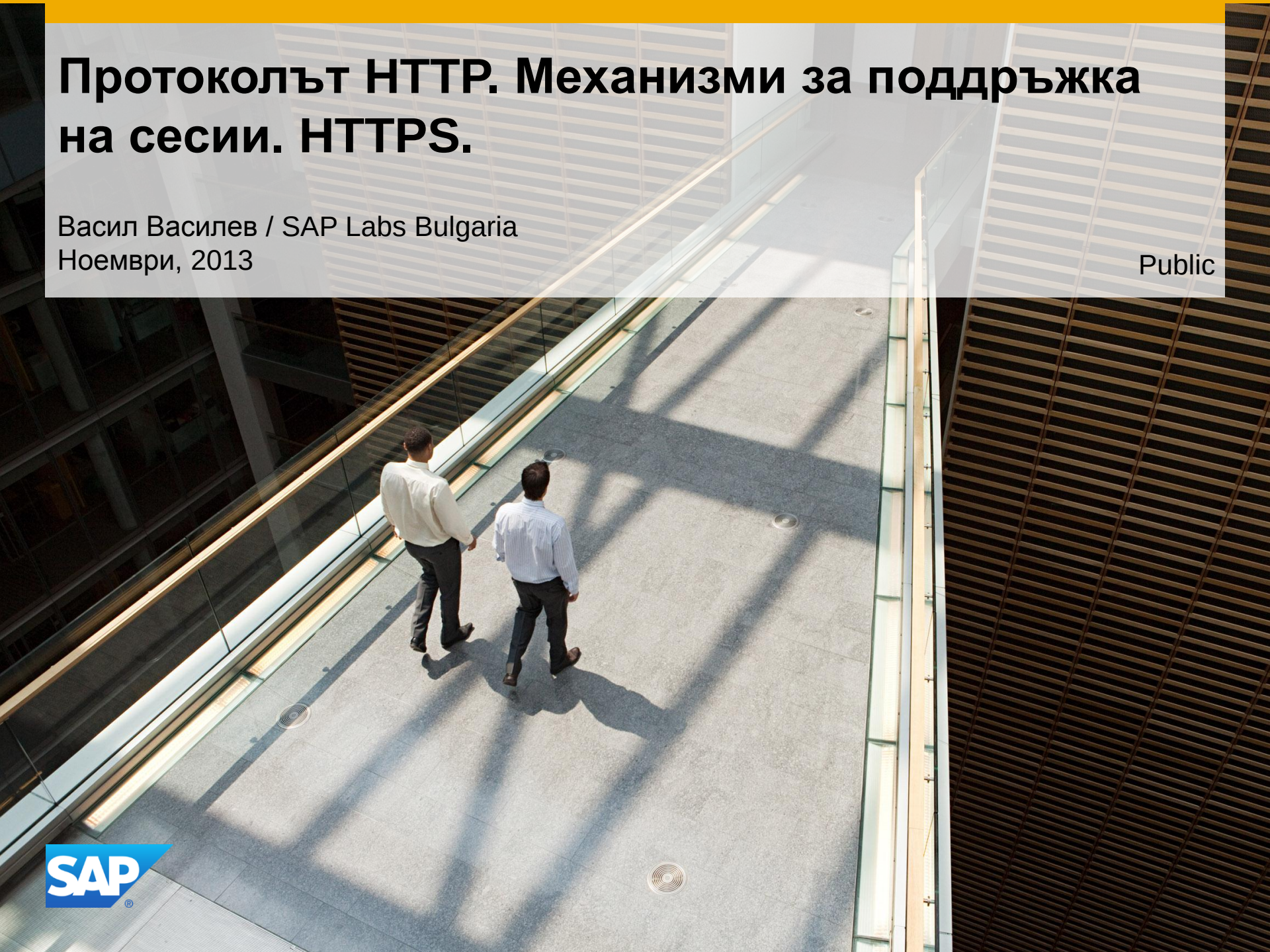
Протоколът HTTP. Механизми за поддръжка на сесии. HTTPS.

Васил Василев / SAP Labs Bulgaria
Ноември, 2013

Public



Съдържание

Протоколът HTTP.

- HTTP Общи сведения
  - Характеристики
  - Съобщения
  - Ресурси
- HTTP Заявка
- HTTP Отговор

Механизми за поддръжка на сесии.

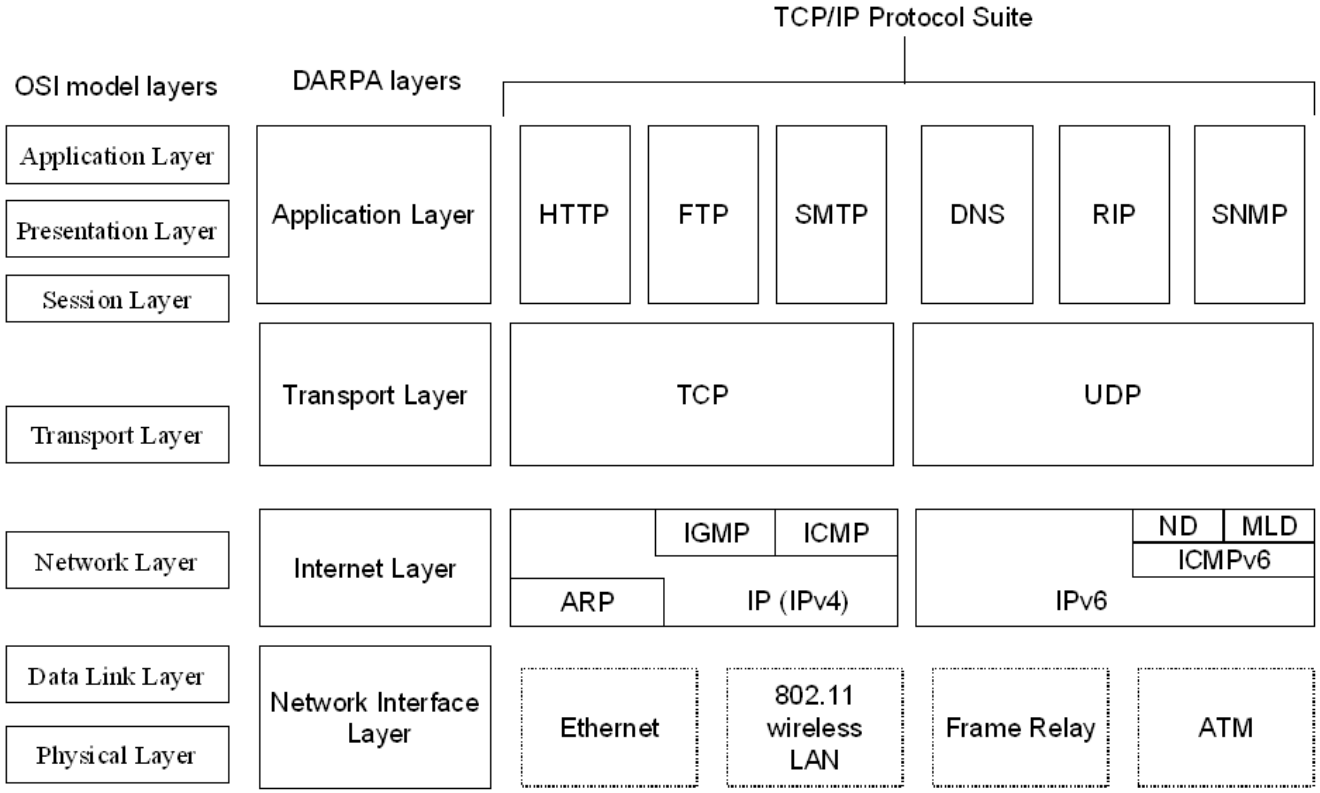
- HTTP Сесия
- Cookies
- Session failover

HTTPS

Що е протокол?

„communications protocol is a system of digital rules for message exchange within or between computers. When messages are exchanged through a computer network, the rules system is called a **network protocol**.“

http://en.wikipedia.org/wiki/Communications_protocol



Протоколът HTTP

- Създаден е специално за посрещане на нуждите на World Wide Web, от създателя на WWW Тим Бърнърс-Лий. В първата версия клиентите са могли да правят само GET заявки, а сървърът е връщал единствено HTML документи като отговор.
- Абревиатурата означава - HyperText Transfer Protocol (протокол за трансфер на хипервръзки). Хипервръзка е автоматична препратка (указател) към определен елемент в документа или извън него. Установява унифицирани правила за комуникация в Клиент-Сървър архитектура между софтуерни клиенти наречени User Agent (Потребителски агенти) и Уеб сървъри
- Развитието му се координира от IETF и W3C, текущата версия е HTTP 1.1 – стандартизирана с RFC 2616 (1999г)

Характеристики на HTTP

- **Приложен протокол** – като транспортен протокол, почти винаги се ползва TCP/IP, в редки случай и UDP. По подразбиране слуша на порт 80.
- **Модел „Заявка-Отговор“ (“Request-Response”) -** служи за комуникационен канал в „Клиент-Сървър“ архитектура, като следва строги правила за ред и формат на съобщенията между участниците.
- **Не пази състояние (Stateless)** – всяка клиентска заявка е независима сама по себе си. Сървърът не обвързва логически серия заявки от определен клиент. Това води до липса на вграден в протокола механизъм за поддържане на сесии.
- **HTTP Транзакция** - /опростен модел – без преизползване на конекцията/
  1. Клиентът отваря комуникационен канал (TCP сокет)
  2. Изпращане на заявка от клиента към сървъра
  3. Сървърът връща отговор на клиента
  4. Затваряне на сокет-а от сървъра.

HTTP Съобщения

- **Заявка** – инициатор е клиентът – подава информация на сървъра, достъп до кой ресурс иска да получи и каква операция иска да извърши с него (и евентуални входни параметри). **Клиент** (условно наречен User-Agent в HTTP) може да бъде всяко софтуерно приложение, спазващо правилата на протокола на комуникация.

- **Отговор** – изпраща се от уеб сървъра, като резултат от изпълнението на клиентска заявка. Под **уеб сървър** разбираме софтуерно приложение, служещо като доставчик на дадени услуги върху определени негови ресурси.

- **Структура на HTTP Съобщенията**

1. Начален ред – задължителен, специфичен за отговора и заявката
2. Заглавия /Headers/ - опционални, служат за задаване на параметри на HTTP транзакцията или за допълнителни (мета-данни) за съобщението
3. Празен ред
4. Данни /Тяло/ – опционален

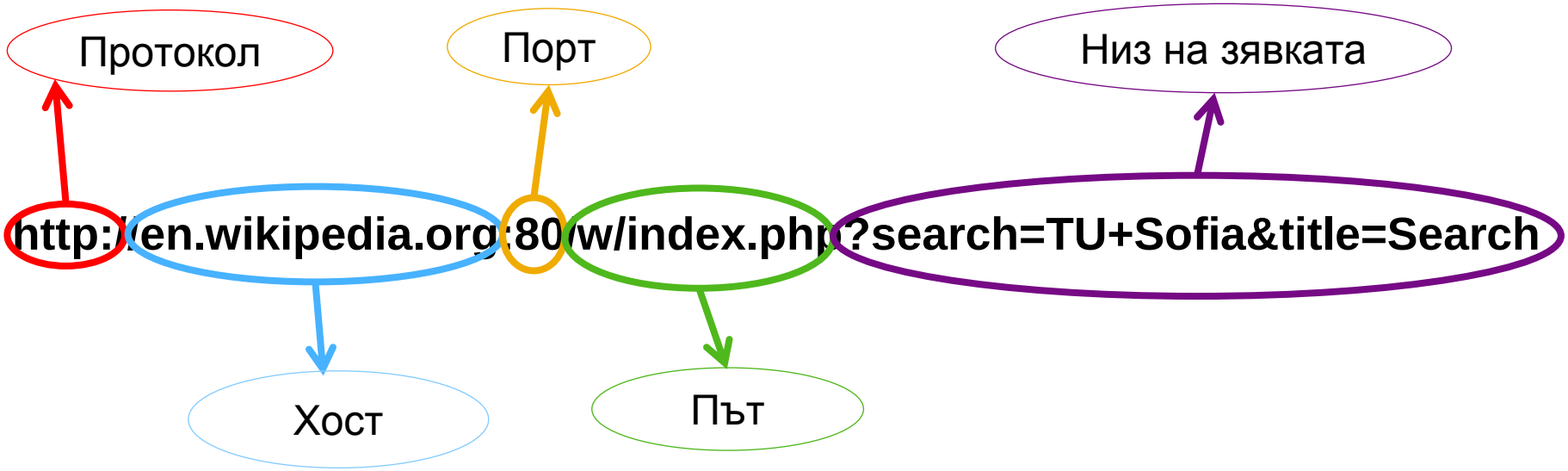
HTTP Ресурси

1. Унифициран локатор на ресурси /URL/ - стандартизиран адрес на даден мрежов ресурс /документ или страница/. Всяка уеб страница е идентифицирана уникално чрез URL

2. Структура на URL

- Схема – Указва се протокола, следван от „://“
- Хост – IP адреса или домейн името на сървъра, от който искаме да заявим даден ресурс
- Порт – портът, на който „слуша“ съответния протокол на сървъра, при HTTP по подразбиране е 80.
- Път – конкретния сървски ресурс, който искаме да достъпим
- Низ на заявката(Query string) – съдържа парамери, които клиентът иска да изпрати до сървъра за обработка
- Фрагмент идентификатор – указател към специфична част в документа

URL Структура



Структура на HTTP Заявката

- **Начален ред**

HTTP Метод (Глагол) – указва типът операция, която клиентът иска да извърши със заявеният ресурс.

URL – уникален локатор на заявеният ресурс

Версия на HTTP – версията на протокола, която ще се позлва за комуникация

```
GET en.wikipedia.org/w/index.php HTTP/1.1
```

- **Хедъри** - опционални (HTTP 1.1 задължава специфицирането на хедър HOST в заявката). Възможно е да дефинира множество хедъри, като всеки от тях заема точно един ред и следва форматът: “Име на хедър: Стойност на хедър”

```
Connection: Keep-Alive
Host: en.wikipedia.org
```

- **Данни/Тяло/** – опционални, може да съдържат множество редове, включително и празни

Видове HTTP Методи

- **GET** – за зареждане на ресурс от сървъра
- **POST** - изпраща данни (от HTML форма) за обработка от сървъра. Данните се съдържат в тялото на заявката
- **HEAD** - идентичен с GET, с разликата, че отговорът няма да върне тяло, а само хедъри
- **PUT** – ъплоудва специфичен ресурс
- **DELETE** – трие специфичен ресурс
- **TRACE** – указва на сървъра да върне низа на заявката в тялото на отговора
- **OPTIONS** – казва на сървъра да му върне всички позволени методи за даден ресурс
- **CONNECT** – за работа с проксита
- **PATCH** – за подмяна на части от ресурса

HTTP Заявка - Пример

```
GET /w/index.php?search=tu+sofia&title=Special%3ASearch HTTP/1.1
```

```
Accept: application/x-ms-application, image/jpeg,
application/xaml+xml, image/gif, image/pjpeg, application/x-ms-
xbap, */*
Referer: http://en.wikipedia.org/wiki/Main_Page
Accept-Language: en-US
User-Agent: Mozilla/4.0
Accept-Encoding: gzip, deflate
Host: en.wikipedia.org
Connection: Keep-Alive
```

```
<Празен ред>
```

```
<GET заявките нямат тяло!>
```

Структура на HTTP Отговор

- **Начален ред –** съдържа 3 елемента, разделени с празно пространство помежду си:

  Версия на HTTP

  Статус код – обяснява резултата на изпълнието на заявката

  Причина – кратко обяснение на статус-кода

HTTP/1.1 200 OK

- **Хедъри**

Date: Sat, 17 Nov 2012 15:08:15 GMT

Server: Apache

- **Данни/Тяло/** – отговорите обикновено връщат данни, като тук най-често се съдържа **HTML документът**, получен на базата на клиентската заявка.

HTTP Отговор - Пример

```
HTTP/1.1 200 OK
```

```
Date: Sat, 17 Nov 2012 15:08:15 GMT
Server: Apache
X-Content-Type-Options: nosniff
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: en
Vary: Accept-Encoding,Cookie
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Encoding: gzip
Content-Length: 8582
```

```
<Празен ред>
```

```
<HTML>Пропускаме документа за простота!</HTML>
```

HTTP Статус кодове (1)

- **Трицифрени кодове, идентифициращи какъв е резултът от обработката на клиентската заявка**
- **Групирани са в 5 категории, на базата на цифрата на стотиците**

1. Група 100 – те са чисто информационни, не дават индикация дали заявката е била успешна или не. Служат за „временни“ кодове, т.е. заявката е пристигнала, но сървърът не е готов с резултата все още:

100 Continue
101 Switching protocols

2. Група 200 – сървърът е обработил успешно клиентската заявка

200 OK
206 Partial content

HTTP Статус кодове (2)

3. Група 300 – ресурсът е наличен, но е разположен на друго място

301 Moved permanently	**304 Not Modified**
307 Temporary redirect	

4. Група 400 - клиентска грешка

400 Bad Request	**404 Not Found**
401 Not Authorized	**408 Request Timeout**

5. Група 500 - сървърна грешка

500 Internal Server Error	**501 Not Implemented**
503 Service Unavailable	

HTTP Хедъри (1)

- **Основни (General headers)** – могат да се ползват едновременно и в заявки, и в отговори. Съдържат информация (мета-данни) за самото съобщение или за метода на комуникация

```
Connection: keep-alive
Date: Sat, 17 Nov 2012 15:08:15 GMT
```

- **Заявка (Request headers)** – специфични са само за заявките и могат да съдържат данни за самата заявка или за клиента

```
Accept: text/html
Accept-Charset: utf-8
Accept-Language: en-US
User-Agent: Mozilla/4.0
```

HTTP Хедъри (2)

- **Отговор (Response headers) -** съдържат информация (мета-данни) за сървъра и формата на съобщението

```
Server: Apache
Allow: GET, HEAD
```

- **Същински (Entity headers)** – информация за самото съдържание на данни (тяло) и/или за ресурса, заявен от клиента:

```
Content-Language: en
Content-Encoding: gzip
Content-Length: 8582
Last-Modified: Tue, 15 Nov 2012 12:45:26 GMT
```

Кеширане

- **Браузър кеш** – механизъм за временно съхранение на ресурси, с цел по-бързият им достъп. Предотвратява излишното натоварване на мрежата със заявки за зареждане на ресурси, които не са се променяли. Първият достъп до даден ресурс отива директно на сървъра и документът се зарежда в браузър кеша. Следващата заявка може да бъде обслужена или от уеб сървъра, или от браузър-кеша. При зареждане от кеша е възможно потребителят да получи некоректни (стари) данни. Има два основни метода как да се установи това*:*

- **Срок на годност (“expiration period”)** – целта е да се елиминират HTTP заявки, които биха получили еднакъв документ като отвор. За целта браузър кешът трябва да знае за колко дълго може да се “довери” на кеширания документ.

  Cache-Control: max-age=3600

  Expires: Thu, 01 Dec 2012 16:00:00 GMT

- **Актуалност на данните (“validation”) –** сървърът предоставя възможност на клиента да провери дали кешираните му ресурси са били променяни.

  Last-Modified: 01 Dec 2012 16:00:00

  If-Modified-Since: 01 Dec 2012 16:00:00

Добра статия по въпроса: http://blog.httpwatch.com/2007/12/10/two-simple-rules-for-http-caching/

Примерен Java User-Agent

```
// Отваряме физическа(TCP/IP) връзка към уикипедия
Socket httpSocket = new Socket("wikipedia.org", 80);

// Инициираме стрийм обекти за писане/четене по мрежата
BufferedWriter outputStream = new BufferedWriter(new
        OutputStreamWriter(httpSocket.getOutputStream()));
BufferedReader inputStream = new BufferedReader(new
        InputStreamReader(httpSocket.getInputStream()));

// Композираме началния ред на HEAD заявката
outputStream.write("HEAD " + "http://en.wikipedia.org/wiki/Main_Page" + “
        HTTP/1.1\r\n");
outputStream.write("\r\n");

// Добяваме хедъри
outputStream.write("Host: wikipedia.org\n");
outputStream.write("User-Agent: My custom HTTP communicator");
outputStream.write("\r\n");

// Изпращаме заявката
outputStream.flush();
```

HTTP Сесия

Сесията е концепция, която позволява да се поддържа връзка (състояние) между 2 или повече http requests, изпратени към даден сървър в Internet.

Например, ако пазаруваме online от сайт – какво би се случило с кошницата ни за пазаруване, ако нямаме сесия?

Механизми за поддръжка на сесия (1)

- **Hidden fields forms**

HTML страницата трябва да съдържа скрита (hidden) форма:
<INPUT TYPE="HIDDEN" NAME="jsessionid" VALUE="...">

Когато формата е submit-ната, специфицираните име и стойност ще бъдат включени в GET или POST параметрите. Но, този механизъм за поддръжка на сесии би работил само в случай, че всички страници са конструирани динамично всеки път. Защо е необходимо да се генерират всеки път? Какво би се объркало?

Механизми за поддръжка на сесия (2)

- **URL Rewriting (презаписване)**

  Може да добавите в края на всяко URL данни, които да унифицират сесията, за да може сървъра да прочете тези данни и да намери вашата сесия. Това е особено добър начин за поддръжка на сесия, в случай че браузърът, който ползваме, не поддържа cookie-та или просто те са disabled. Но, разбира се, този подход също има своите недостатъци, защото трябва да подсигурим, че всяко URL, което бъде генерирано, ще съдържа правилните данни. Какво би станало ако bookmark-нем някакво URL в такъв случай?

  http://<my_java_mighty_site>?jsessionid=I_am_unique_session_identifier

Механизми за поддръжка на сесия (3)

- **Cookies (Бисквитки)**

Cookie-тата са малки текстови файлове генерирани от сървъра и изпратени на клиента в header-ите.

Как работят бисквитките ☺

1. Клиентът изпраща request към сървъра.

2. Сървърът отговоря и в header-ите на response-а праща към клиента cookie-тата, които ще се ползват за проследяване на сесията

Примерен отговор (response) на apache tomcat web container-а съдържа header:

Set-Cookie: JSESSIONID=ACFF1B473DAB71CD27AA16049D61265E; Path=/SessionTest

3. Всеки следващ request, изпратен от клиента, трябва да съдържа Cookie header-а, за да може сървърът да намери сесията на клиента

Cookie: JSESSIONID=ACFF1B473DAB71CD27AA16049D61265E

Атрибути на cookie-та

- Cookie-тата са дефинирани в RFC 2109.

- **Comment** - обикновено се използва от програмистите, за да обсноват нуждата от ползването на cookie-то.
- **Domain** – определя домейна, за който cookie-то е валидно и ще бъде изпращано.
- **Max-age** – задава lifetime-а на cookie-то в секунди. След като изтече валидността на cookie-то, клиентът не трябва да го праща повече.
- **Path** – специфицира subset от URL-та, където cookie-то може да бъда изпращано.
- **Secure** – този атрибут указва, че cookie-то може да бъде трансферирано само https протокола.
- **HttpOnly** – когато този атрибут е добавен, cookie-то не може да бъде четено или променяно от java script
- **Version** – цяло число, което определя на коя версия на RFC-то отговоря cookie-то.

Cookies

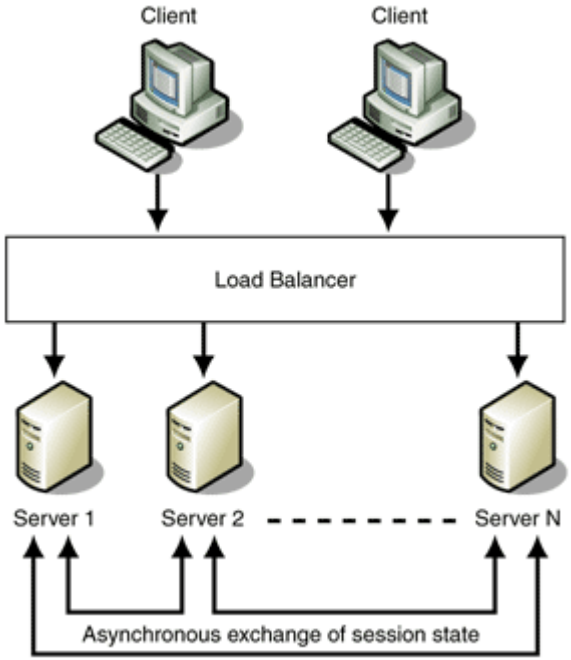
Къде би се пратило cookie със следните:
Domain = myhost.com, Path = /TU

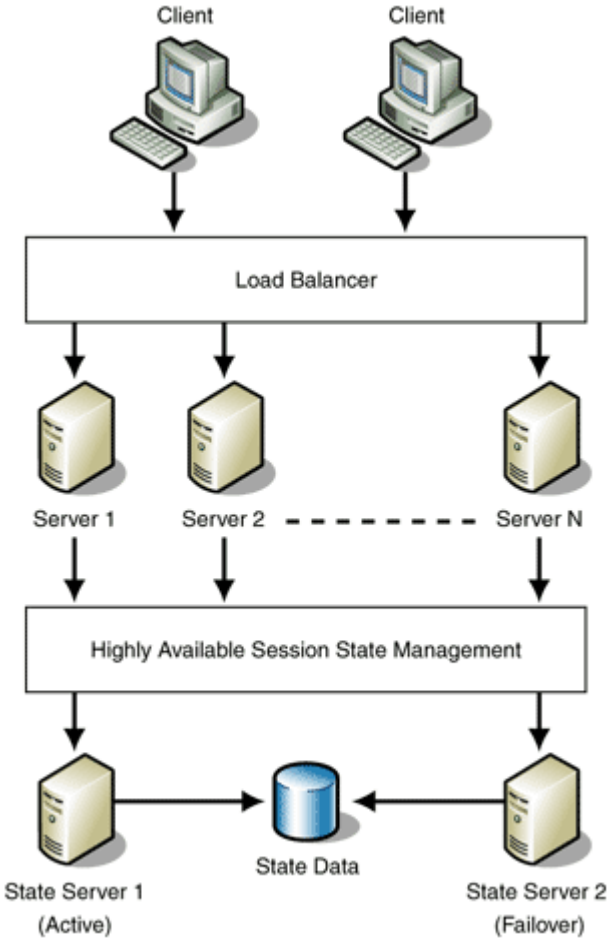
http://myhost.com/TU/Resources

http://myhost.com/TUTest/Resources

Подръжка на сесии в клъстър среда и session failover

За да отговорят на нуждите на бизнеса, където хиляди потребители работят с голямо множество от приложения, е необходимо сървърите да гарантират, че клиентските сесии няма да бъдат „изгубени“, в случай че някой от сървърите в клъстъра „падне геройски“, поради някакъв проблем в хардуера или софтуера.





HTTPS протокол

Https(Hipertext Transfer Protocol Secure) е “secure” вариантът на http протокола. Всички правила за http протокола важат и за https протокола, със съществената разлика, че се ползва върху TLS/SSL протокола. Това позволява данните да бъдат трансферирани по мрежата криптирани.

Https протоколът е особено широко използван от сайтовете на банки, online магазини и т.н. Когато клиентът (браузърът) осъществи връзка с даден сайт по https, сайтът криптира връзката с дигитален сертификат, който е издаден от сайта. Типичен премер за такъв сертификат е online банкирането, при което банката издава сертификат, който потребителите трябва да инсталират в техните браузъри.

Как протича комуникацията по HTTPS протокола

Клиентът изпраща съобщение ClientHello

1

Сървърът отговаря с ServerHello съобщение

2

Сървърът може да прати съобщение Certificates, при което праща своя сертификат към клиента. След което клиентът ще провери дали сървърският сертификат е доверен (trusted).

3

Сървърът изпраща CertificateRequest съобщение, с което изисква от клиента да изпрати клиентския сертификат (от примера с online банкирането – това е сертификатът, който банката ни е издала и ние сме инсталирали в нашия браузър).

4

Клиентът отговаря с Certificate съобщение, което съдържа клиентския сертификат и сървърът го проверява.

5

Клиентът изпраща ClientKeyExchange съобщение, което може да съдържа публичния ключ (key) на клиентския сертификат или може съобщението да е празно.

6

Клиентът изпраща CertificateVerify съобщение, което съдържа private key-а на сертификата, за да може сървърът да е сигурен, че клиентът разполага и с двата key-а.

7

Клиентаът изпраща ChangeChiperSpec съобщение, последвано от Finished съобщение.

8

Сървърът отговаря със същите две съобщения, след което се осъществява secure връзка.

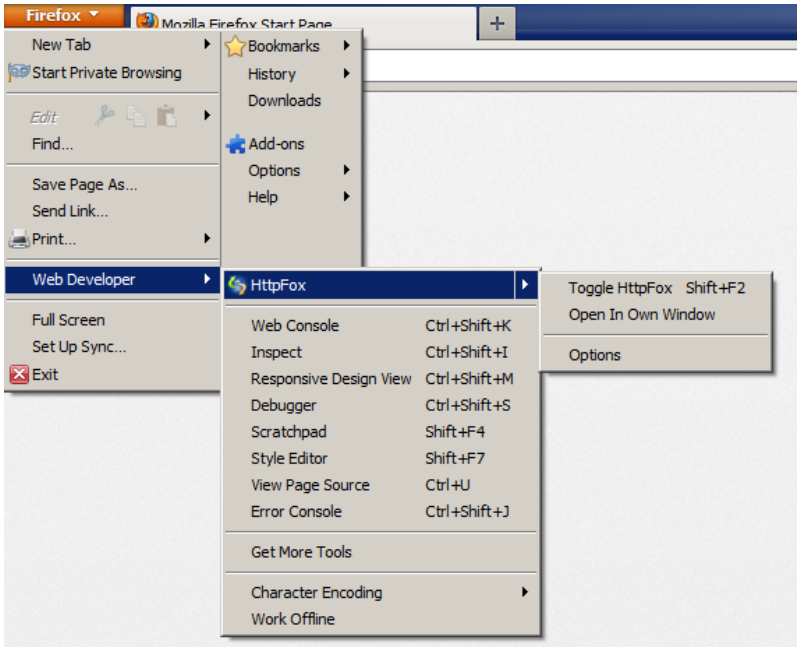
Благодаря за вниманието!

За контакти:

vasil.vasilev@sap.com (лектор)
nikolay.samardzhiev@sap.com
radoslav.smilyanov@sap.com
SAP Labs Bulgaria
София 1618
бул. Цар Борис III, 136А
тел: 02 91 57 690

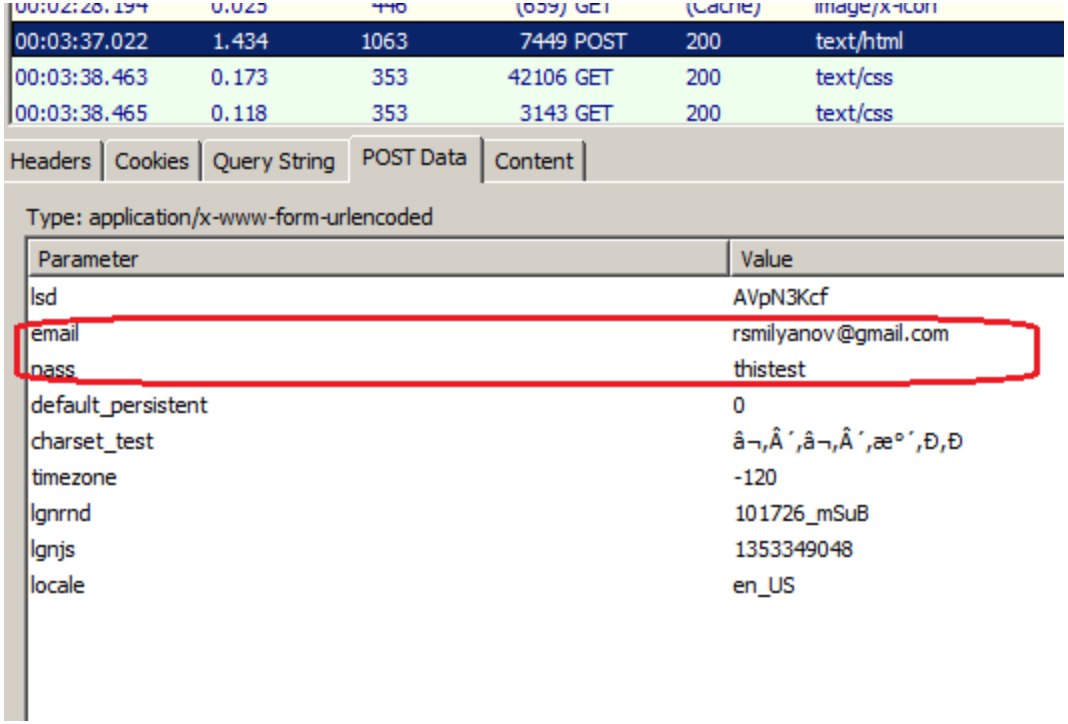
Как да проследяваме HTTP трафика на нашия браузър

В Mozilla Firefox може да се инсталира HttpFox add-on. След като го инсталирате, може да стартирате програмата, както е показнано на картинката



Как да проследяваме HTTP трафика на нашия браузър

Ако успеете да накарате някой да се аутентицира в неговия facebook, ползвайки HttpFox, може да видите неговите username и парола ☺



ИЗПОЛЗВАНА ЛИТЕРАТУРА

- http://www.w3.org/Protocols/rfc2616/rfc2616.html
- http://www.f5.com/flash/wbt/http-basics-i/player.html
- http://www.f5.com/flash/wbt/http-basics-ii/player.html