Сървър комуникация в iOS

05.01.2012

Съдържание

- Видове уеб услуги
- Видове формати за предаване на информация
- Комуникация със сървър в IOS
- Предаване на потоци
- Push notifications

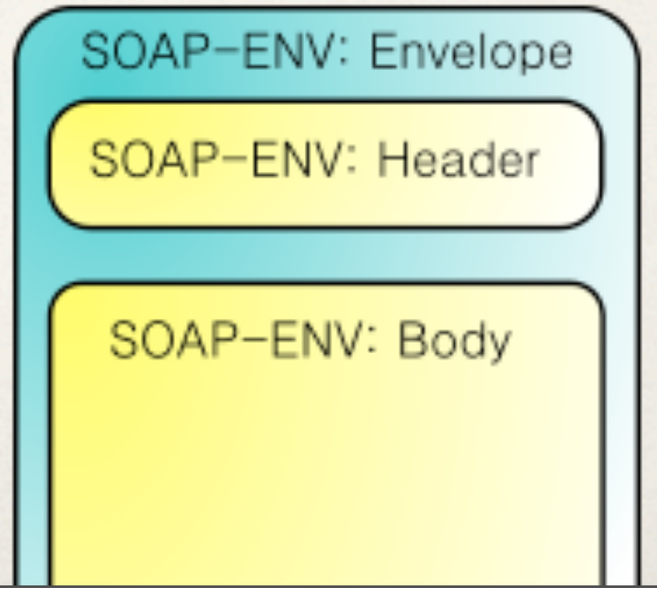
Видове уеб услуги

- Какво е уеб услуга?
- Кога една уеб услуга е RESTful(Representional state transfer)?
- Съществуване на ресурси, всеки представен чрез уникален идентификатор (URI)
  пример: http://iosRulez.com/iPhone-scrolls
- Интернет медия тип(MIME type), поддържан от уеб услугата,
  пример: text/html; charset=UTF-8
- поддържане на набор от операции - HTTP GET, PUT, POST, DELETE
- Stateless комуникация - сървърът не знае нищо за данните от клиента, цялата необходима информация е в подадената заявка

Видове уеб услуги

- Какво е SOAP?
  - Simple Object Access Protocol
  - “премества” XML съобщение от точка А до точка Б
  - може да се използва през HTTP, SMTP, TCP...
  - дефинирана XML схема (WSDL спецификация)

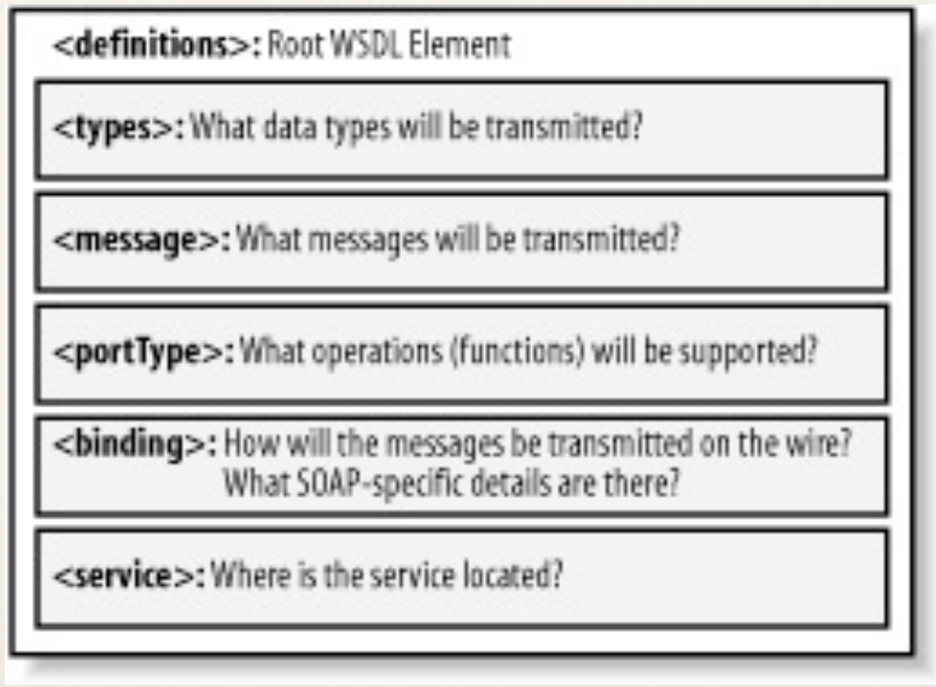
```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice> ...
  </soap:Body>
</soap:Envelope>
```



Видове уеб услуги

- Какво е WSDL(Web Service Definition Language)?
  - Спецификация за описание чрез XML на функционалности, предложени от някоя уеб услуга
  - платформено и езиково независима

- използва се(не само) за описание на SOAP уеб услуги



Предаване на информация

- Plain text
  - MIME тип: text/plain
- JSON
  - MIME тип: “application/json”
  - опростен формат за обмяна на данни
  - състои се от две структури :
    - колекция от двойки име/стойност (хеш таблица)
    - подреден списък от стойности (масив, списък)

{“christmasPresent” : “iPhone”}

{“christmasPresents” : [{“mother”: “iPhone”}, {“father”:”iPad”}]

Предаване на информация

- JSON : IOS
  - json-framework : http://code.google.com/p/json-framework/downloads/list
  - embedded library (папката се копира в проекта)
  - `#import JSON.h`

```
SBJSON *parser = [[SBJSON alloc] init];
NSDictionary *object = [parser objectWithString:json_String
                                          error:someErrorObject];
```

  - повече информация : http://mobile.tutsplus.com/tutorials/iphone/iphone-json-twitter-api/

Предаване на информация

- XML
  - MIME type – application/xml, text/xml
- Protocol Buffers
- MIME type – application/x-protobuf
  - представяне на данни в дефиниран от Google формат
  - препращане на двоичния код на .proto файлове между сървър и клиент
  - възможност да се конвертира до XML и JSON

```
message Person {
    required string name = 1;
    required string email = 2;
}   optional PhoneNumber number = 3;
```

Предаване на информация

- Protocol buffers
  - .proto файлът се компилира и генерира код на Objective-C (или Java или Python)
  - генериране на съобщение

```
Person* person = [[[[Person builder] setId:123]
                  setName:@"Bob"] setEmail@"bob@example.com"]
build];
NSData* data = [person data];
```

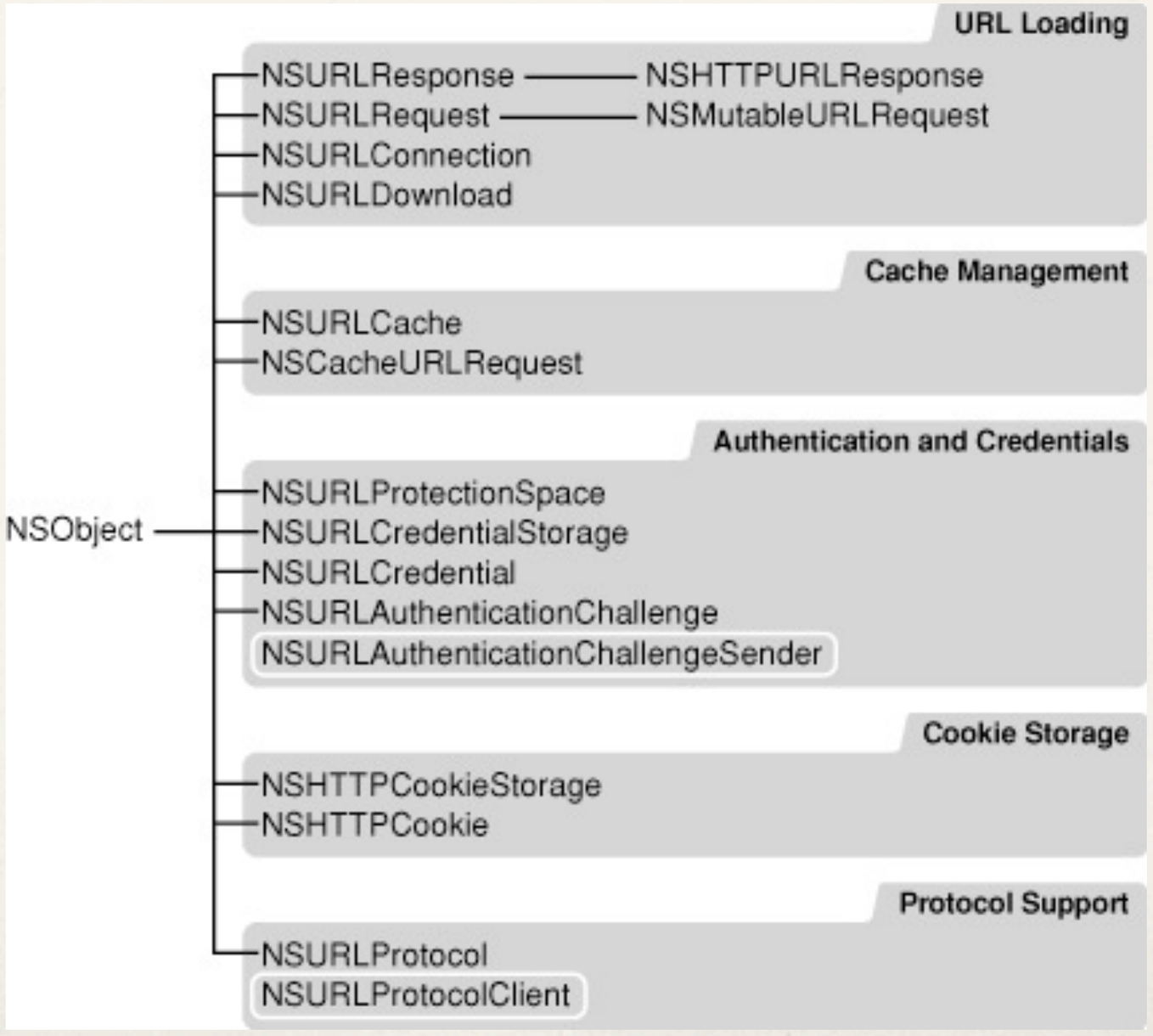
  - четене на съобщение

```
NSData *rawData = ...;
Person* person = [Person parseFromData:rawData];
```

Комуникация със сървър в IOS

- Какво ни е необходимо, за да получим/върнем данни от/към сървър?
  - Осъществена връзка към сървъра
  - (не задължително) Употреба на протокол за комуникация (HTTP, SMTP, SHTTP...). Протоколът има дефинирани правила, по които да се осъществява комуникацията, които трябва да спазваме.
  - Начин, по който да съобщим на сървъра, че сме наши хора (cookies, user credentials)
  - Заявка, казваща на сървъра, че искаме/изпращаме нещо.
  - Начин по който да получим данните и да ги обработим.
  - Начин, по който да обработим получените грешки.
  - Асинхронна обработка на получените отговори!

Комуникация със сървър в IOS



Комуникация със сървър в IOS

Начин по който да обработим получените данни:

- NSData/NSMutableData
  - съдържа буфер от байтове
  - използва се за пазене/извличане на информация

```
const char *utfString = [@”Santa” UTF8String];
NSData *data1 = [NSData dataWithBytes:utfString
 length:strlen(utfString)];
NSData *data2 = [NSData dataWithContentsOfURL:(NSURL object)];
int len = [data2 length];
Byte byteData[len];
byteData = [data2 bytes];
NSString *santaString = [NSString stringWithFormat:@”%@”, data1];
```

Комуникация със сървър в IOS

заявка, казваща на сървъра, че искаме/изпращаме нещо:

- NSURLRequest/NSMutableURLRequest
  - заявка към дадено URL, без значение от протокола и схемата на URL

```
NSURLRequest *req = [NSURLRequest requestWithUrl:url

cachePolicy:NSURLRequestReloadIgnoringCacheData
timeoutInterval:30.0];
```

- Какво е кеширане?
  - малка памет, която пази вече поискана информация. При повторно поискване извличането става много по-бързо.
  - локално кеширане - в случая на самото устройство (NSURLCache)

Комуникация със сървър в IOS

- CachePolicy варианти

```
NSURLRequestUseProtocolCachePolicy
NSURLRequestReloadIgnoringLocalCacheData
NSURLRequestReturnCacheDataElseLoad
NSURLRequestReturnCacheDataDontLoad
NSURLRequestReloadIgnoringLocalAndRemoteCacheData
NSURLRequestReloadRevalidatingCacheData
```

  Та-дааа

- NSURLCache
  - предоставя методи за конфигуриране на локалния кеш (размер, локация)
  - колекция от NSCachedURLResponse

Комуникация със сървър в IOS

Заявка, казваща на сървъра, че искаме/изпращаме нещо:

- NSHTTPURLRequest
  - NSURLRequest с добавени категории, например за връщане на HTTP хедър полета, тяло и т.н.

- NSMutableHTTPRequest – ако искаме да променяме инициализираната заяка, например като и добавим хедър поле

```
[request setHTTPMethod:@"POST"];
[request setValue:@"application/json"
                  forHTTPHeaderField:@"Content-Type"];
[request setHTTPBody:(some NSData)];
```

Комуникация със сървър в IOS

Начин, по който да обработим получените грешки:(и не само)

- NSURLResponse/NSHTTPURLResponse
  - съдържа върнатия от сървъра отговор на изпратена по-рано заявка
  - [response statusCode] – помага ни да открием върната от HTTP протокола грешка
  - дава ни достъп до върнатите хедър полета

```
NSDictionary *fields = [HTTPResponse allHeaderFields];
[NSString *cookie = [fields valueForKey:”Set-Cookie”];
```

  - Въпрос: Как/Кога получаваме response?

Комуникация със сървър в IOS

Осъществена връзка към сървъра:

- NSURLConnection
  - Реализира връзка до специфицирано от NSURLRequest място и сваля предоставените данни.

```
NSURLConnection *theConnection=[[NSURLConnection alloc]
initWithRequest:theRequest delegate:self];
```

  - към делегата се изпраща периодично съобщение `connection:didReceiveData:`, докато не получи данните.
  - делегатът е отговорен за запазване на върнатите данни.
  - Грешките се обработват в `connection:didFailWithError:`
  - Ако връзката е успяла – `connection:connectionDidFinishLoading:`

Комуникация със сървър в IOS

Осъществена връзка към сървъра:

- NSURLConnection
  - получаване на Response (`connection:didReceiveResponse`)
  - възможност за кеширане на Response
  - възможност за проверяване на прогреса на изпратени данни от клиента
  - възможност за изпращане на синхронна заявка

    `sendSynchronousRequest:returningResponse:error:`.
    (не се препоръчва, защо?)
  - автоматично запазване на сесията (получаване и изпращане на бисквитки)

Комуникация със сървър в IOS

Начин, по който да съобщим на сървъра, че сме наши хора

- NSURLCredentialStorage
  - ако сървърът, към който правим заявка, изисква user credentials, те трябва вече да са заредени в този клас
  - NSProtectionSpace – представя място от сървър, което изисква аутентикация(протокол, хост, порт)
  - NSURLCredential – съдържа username, pass, certificates и т.н.

```
[[NSURLCredentialStorage sharedCredentialStorage]
                        setCredential:credential
forProtectionSpace:protectionSpace];
```

- NSURLConnection
  - -(void)connection:didReceiveAuthenticationChallenge:

Комуникация със сървър в IOS

DEMO

Комуникация със сървър в IOS

- МНОГО полезни библиотеки:
  - http://allseeing-i.com/ASIHTTPRequest/
  - http://restkit.org/
  - http://wsclient.neurospeech.com/
  - https://github.com/akosma/iPhoneWebServicesClient

Мрежова комуникация

- Как да четем и пишем на TCP ниво?
  - Инициализираме NSHost обект, който съдържа адрес за достъп
  - Инициализираме входен и изходен поток от тип NSStream
  - свързваме стриймовете с отдалечения адрес

```
[NSStream getStreamsToHost:host port:80
                inputStream:&iStream
               outputStream:&oStream];
```

  - Определяме делегат на стриймовете

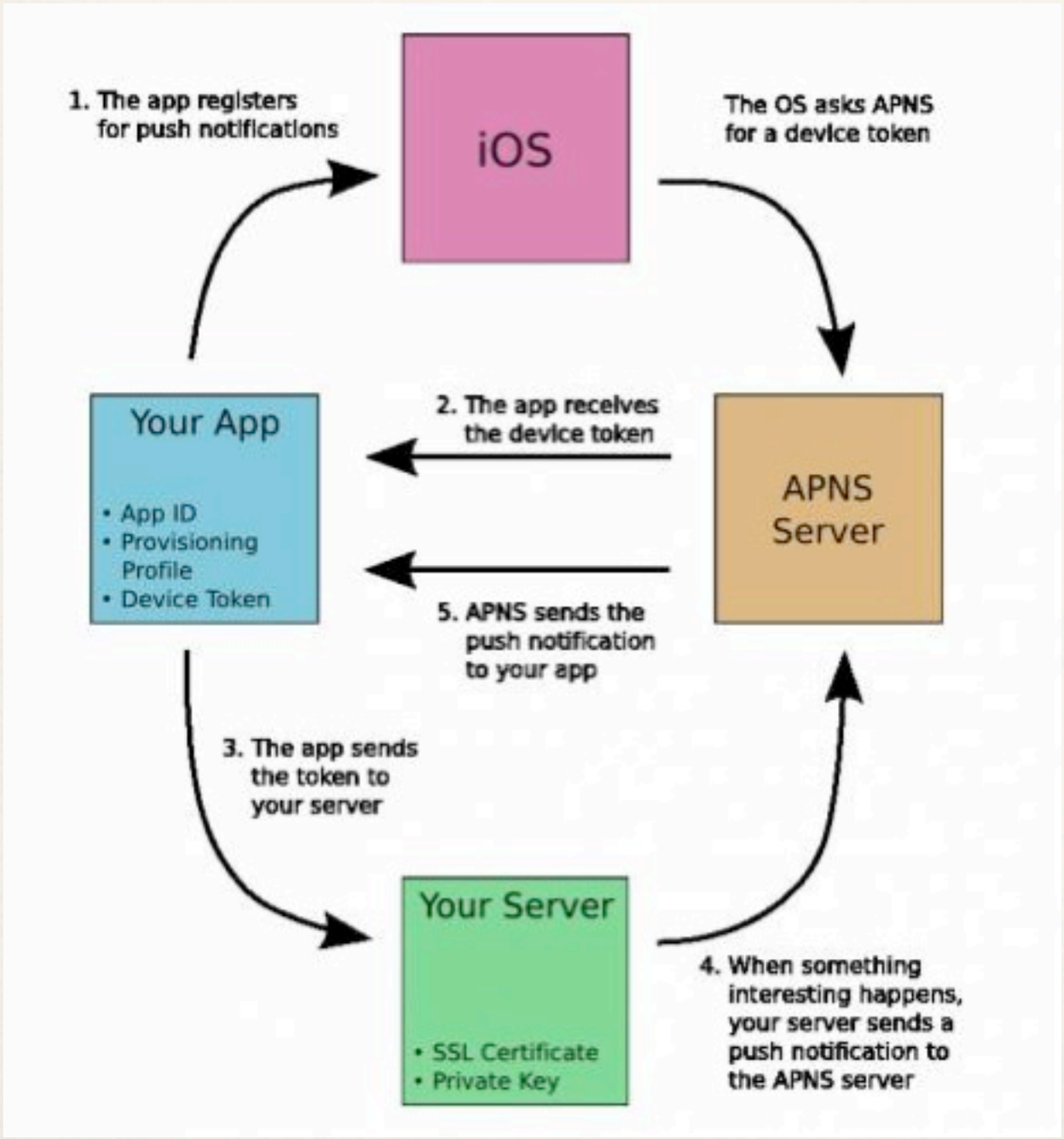
```
[isStream setDelegate:self] //[osStream setDelegate:self]
```

  - делегатът получава съобщение stream:handleEvent, когато информацията е във входния поток
  - ако е възникнала грешка, потоците остават nil стойности

Push Notifications

- Apple Push Notification Services
- постоянно отворена връзка между сървър и клиент
- сървърът известява клиента си за възникнало събитие чрез звуци или текстови съобщения
- спестява се нуждата от “слушащи” процеси на клиента, когато приложението е в background режим (защо това е хубаво?)
- ако приложението не е във “foreground” режим, клиентът получава Alert, че нещо се е случило/ще се случи.
- нотификациите са с малък размер (256 байта), затова и са ефикасни при комуникация с мобилни телефони (защо?)

Мрежова комуникация



Повече информация тук

Push Notifications

- Съобщенията се подават от сървъра в JSON формат
- Сертификати, профили и какво ли още не за регистрация към APN
- Клиент
  - регистриране за нотификации

```
[UIApplication sharedApplication]
registerForRemoteNotificationTypes:
(UIRemoteNotificationTypeBadge |
UIRemoteNotificationTypeSound
| UIRemoteNotificationTypeAlert)]
```

  - проверка дали се е регистрирал коректно (методи в делегата) и получаване на данните от сървъра

```
application:didRegisterForRemoteNotificatonsWithDeviceToken:
application:didFailToRegisterForRemoteNotificationsWithError
application:didReceiveRemoteNotification
```

Благодаря за вниманието!

ВЪПРОСИ ?

Използвана литература

- http://www.iphonedevsdk.com/forum/iphone-sdk-tutorials/76730-webservice-how.html
- Programming IOS 4 на O'REILLY
- http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/URLLoadingSystem/Tasks/UsingNSURLConnection.html
- http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Streams/Articles/NetworkStreams.html
- http://www.slideshare.net/akosma/accessing-rest-web-services-from-ios-applications