Лекция 12. Протоколы передачи данных и типы интеграции ПО с внешними системами

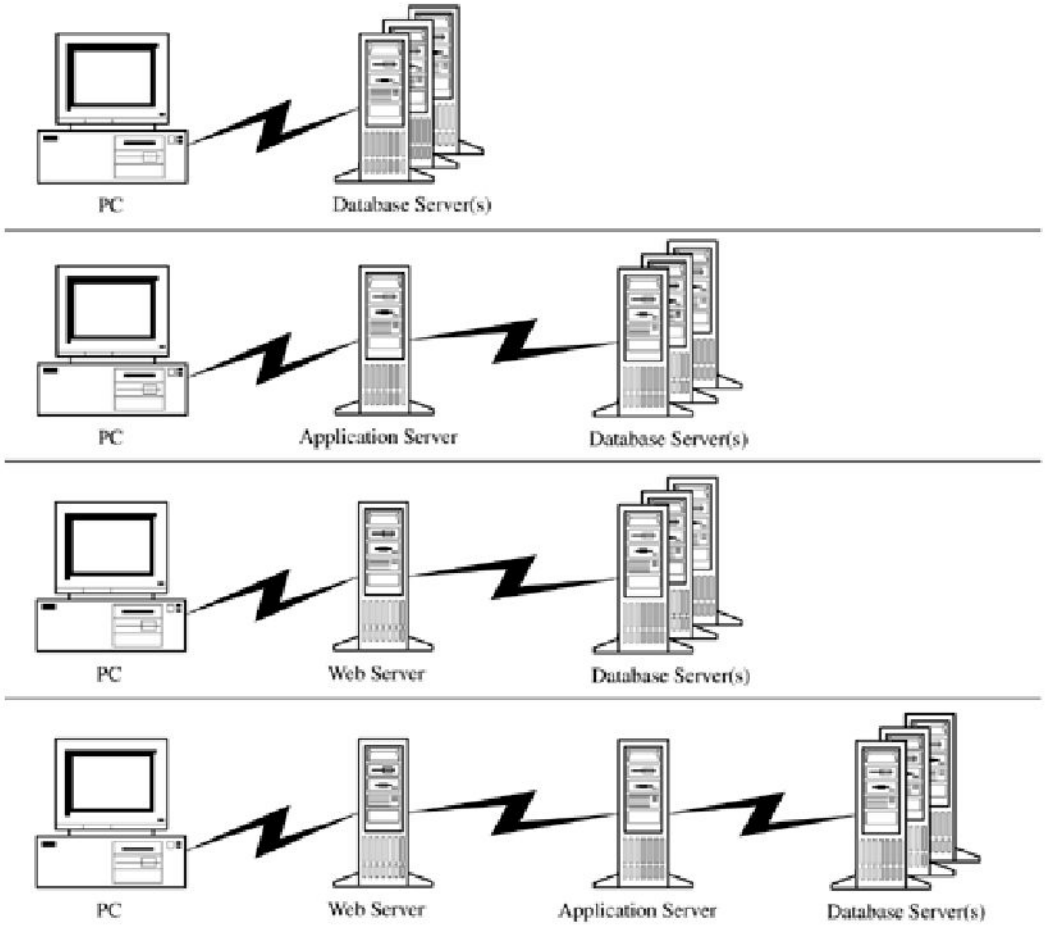
NetCracker®

Agenda

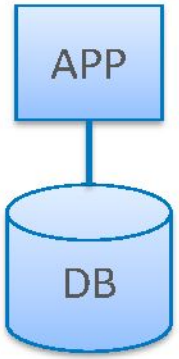
- Клиент-серверная архитектура ПО
- Зачем нужна интеграция?
- Модель обмена информацией OSI
- Протоколы передачи данных
- Основные протоколы прикладного, транспортного и сетевого уровней
- Интеграция через DB
- Интеграция через файловый обмен
- Интеграция через RPC
- Интеграция через web-сервисы
- Основы XML
- Основы работы протокола HTTP

Клиент-серверная архитектура ПО

- Client side vs. Server side
- Server side = {Database, Application, Web server}



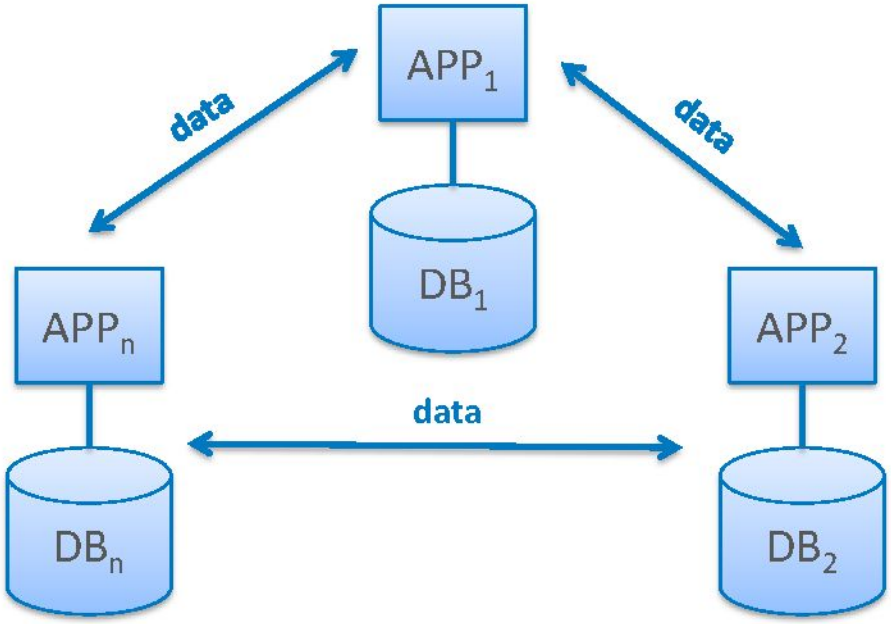
БД и приложение – главные компоненты серверной части ПО, реализующие его бизнес-логику



Зачем нужна интеграция?

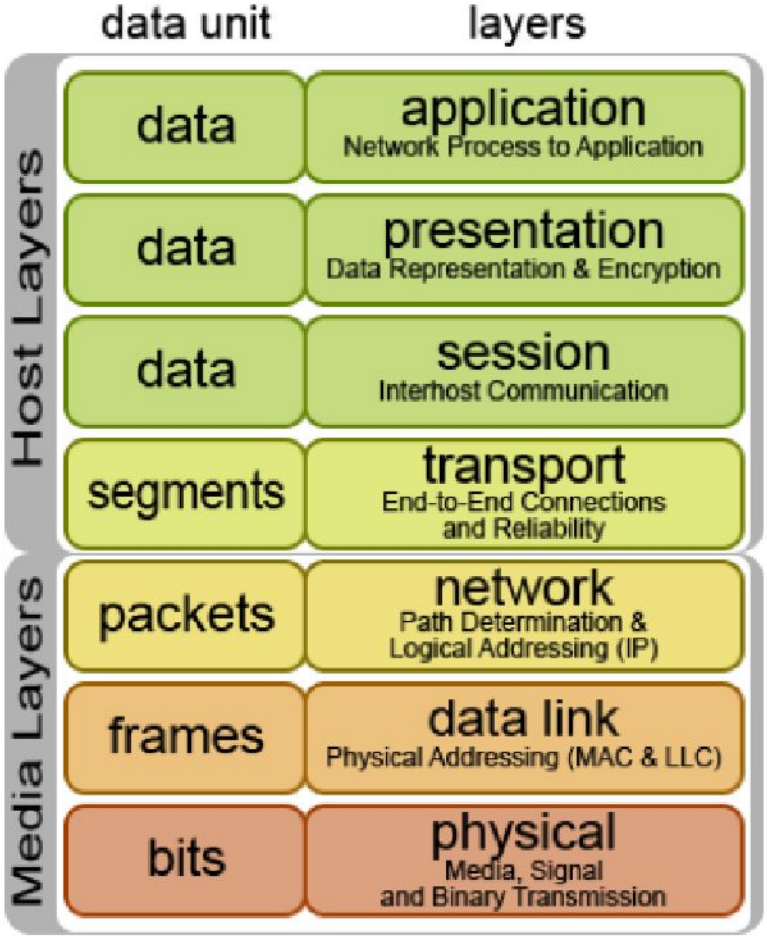
- Реализовано – единожды, использовано – многократно
- Актуальность данных в разных системах
- Необходимость взаимодействия между системами

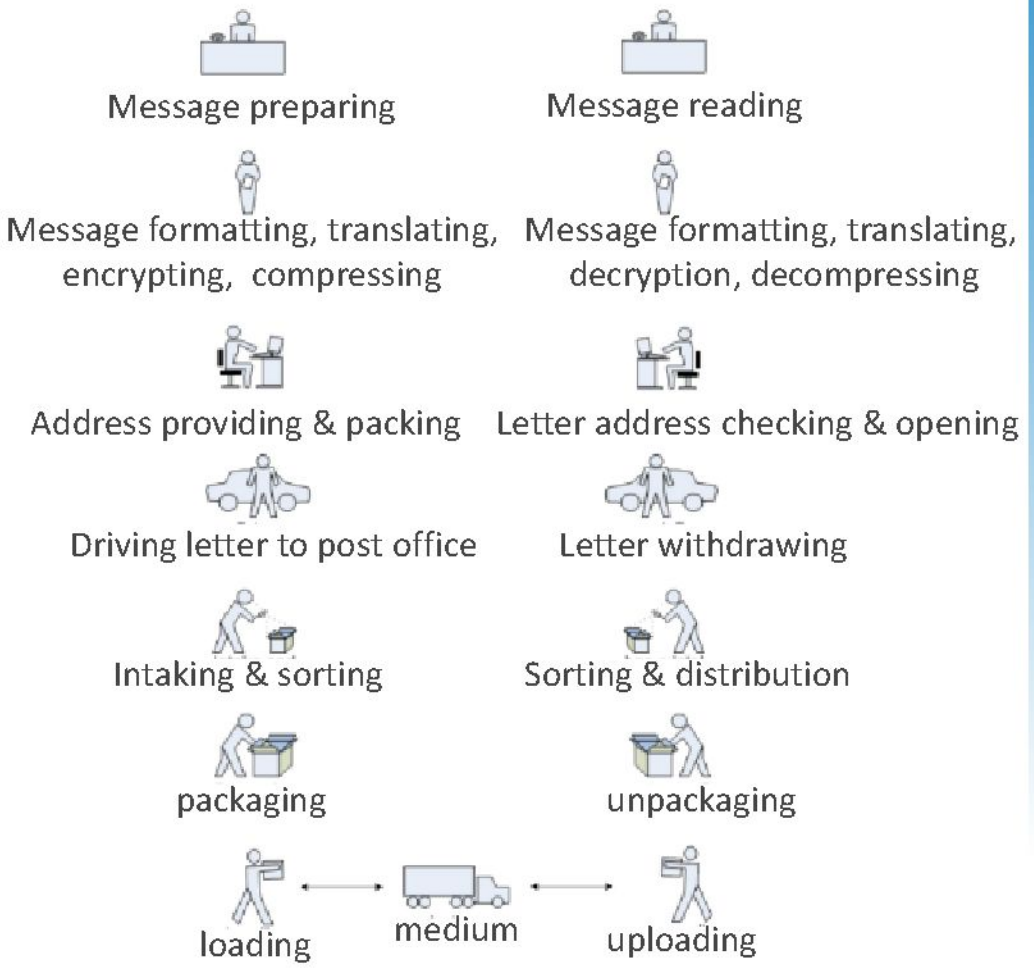
Интеграция – процесс установления способности систем обмениваться информацией



Модель обмена информацией OSI

Open Systems Interconnection (OSI) Model

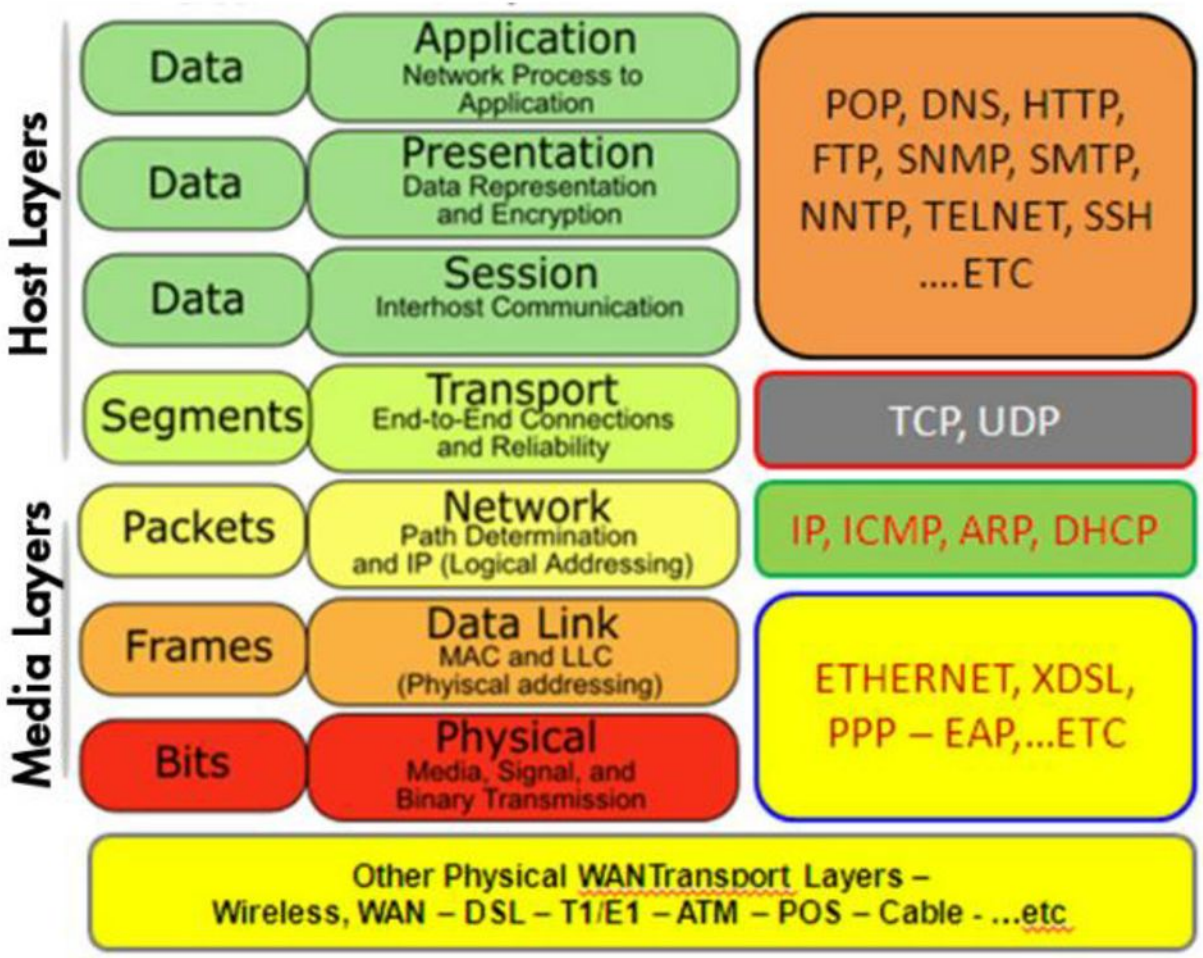




Протоколы передачи данных

Open Systems Interconnection (OSI) Model

Протокол – набор правил и соглашений, позволяющих провести обмен информацией между разнородными системами



Основные протоколы прикладного уровня

Application Layer Protocols

HTTP = Hypertext Transfer Protocol

- обмен контентом между web-сервером и web-клиентом

FTP = File Transfer Protocol

- обмен файлами между конечными сетевыми устройствами

SMTP = Simple Mail Transfer Protocol

- отправка сообщений клиентами электронной почты на сервер электронной почты

POP = Post Office Protocol, IMAP = Internet Message Access Protocol

- извлечение сообщений клиентами электронной почты с сервера электронной почты

HTTPS = Secure HTTP

- безопасный обмен контентом между web-сервером и web-клиентом

SFTP = SSH FTP

- безопасный обмен файлами между компьютерами

SSH = Secure Shell Protocol

- удалённое управление компьютером и обмен любой информацией с использованием алгоритмов шифрования

Основные протоколы транспортного и сетевого уровня

Transport Layer Protocols

TCP = Transmission Control Protocol

- управление передачей данных на уровне приложений
- подразумевает проверку соединения
- проверка факта доставки и порядка доставки данных
- широко используется в системах, не чувствительных ко времени

UDP = User Datagram Protocol

- управление передачей данных на уровне приложения
- не подразумевает проверку соединения
- нет проверки факта доставки и порядка доставки данных
- широко используется в системах реального времени

Network Layer Protocols

IP = Internet Protocol

- управление передачей данных на уроне сетевых устройств
- управляет передачей данных на основе адресов устройств, участвующих в передаче и приёме данных

Способы интеграции ПО

Интеграция через DB

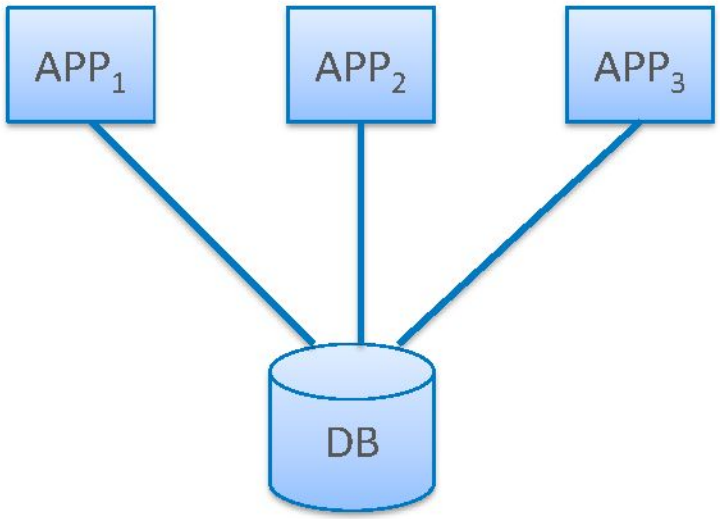
Файловый обмен

Remote Procedure Call (RPC)

Web-сервисы

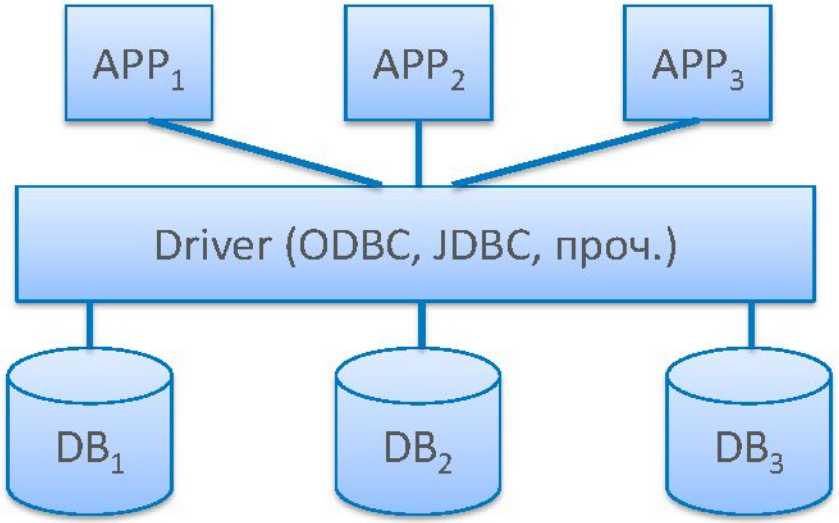
Способы интеграции ПО: интеграция через DB

Общая база данных



- является физическим объединение существующих баз данных
- позволяет иметь единое хранилище данных
- согласованность формата данных
- лёгкость в масштабировании системы
- отсутствие гибкости в администрировании приложений
- высокая стоимость реализации для уже существующих крупных систем
- отсутствие распределения нагрузки на DB сервер

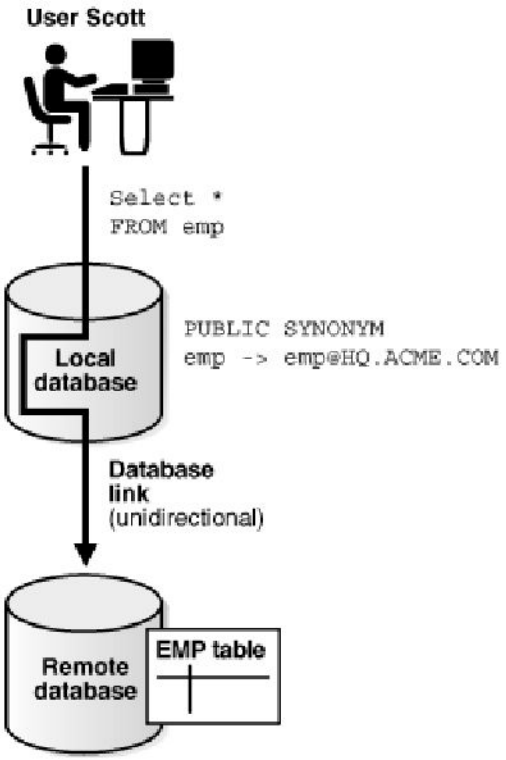
Общий интерфейс доступа

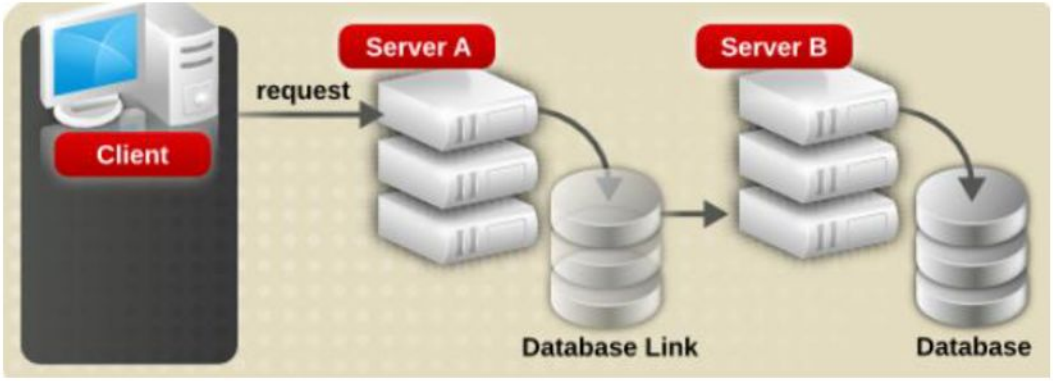


- является виртуальным объединением существующих баз данных
- гибкость в администрировании приложений
- позволяет иметь единый интерфейс доступа к данным
- необходимость реализации механизма real time трансформации данных
- обеспечивает лишь частичную интеграцию баз данных (на уровне единого средства доступа к ним)

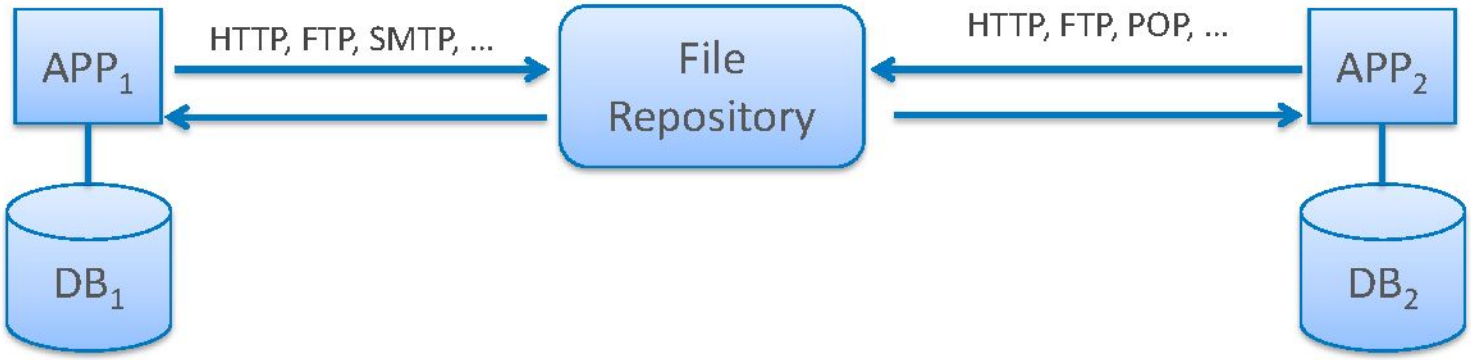
Способы интеграции ПО: интеграция через DB

Database link





Способы интеграции ПО: файловый обмен



- файлы – универсальный механизм хранения данных, встроенный в любую ОС и поддерживающийся любым языком программирования
- требует выбора единого формата файлов
- позволяет скрыть внутреннюю реализацию интегрируемых приложений (принцип инкапсуляции)
- гибкость в выборе протокола передачи файлов
- необходимость дополнительного сервера – файлового репозитория
- слабая связанность интегрируемых приложений
- риск рассинхронизации интегрируемых приложений из-за низкой частоты обмена информацией
- риск перехвата данных, находящихся в файле

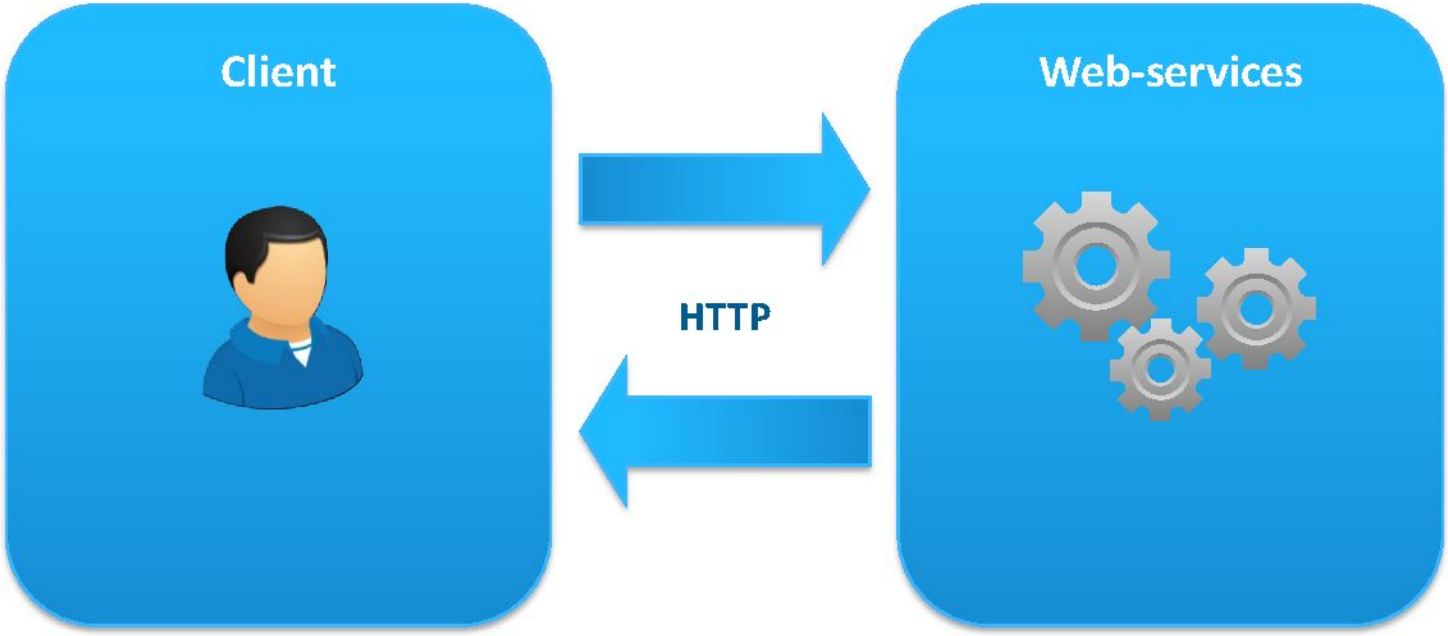
Способы интеграции ПО: RPC

RPC = Remote Procedure Call – первый высокоуровневый механизм взаимодействия приложений

- как и в случае файлового обмена, позволяет скрыть внутреннюю реализацию интегрируемых приложений (принцип инкапсуляции)
- в отличие от файлового обмена, позволяет реализовать быструю реакцию на обновление данных
- каждое приложение самостоятельно обеспечивает целостность данных и может изменять их формат
- поддерживается такими технологиями: CORBA, .NET Remoting, Java RMI, др.
- недостатком является необходимость работоспособности всех приложений в момент взаимодействия
- файерволы и прокси-сервера обычно блокирует такой траффик, поэтому RPC требует дополнительной настройки политики сетевой безопасности

Способы интеграции: web-service

- **Web-сервис – удалённый программный модуль, предоставляющий свой функционал средствами XML и протокола HTTP**



XML

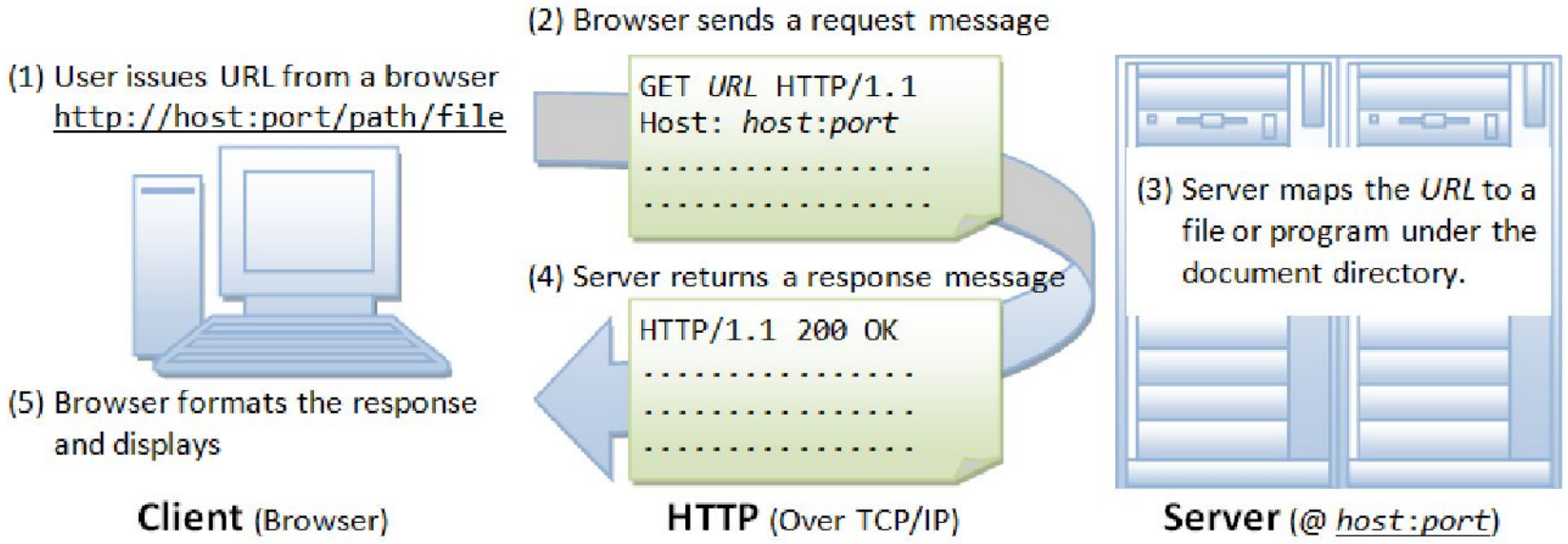
XML = Extensible Markup Language

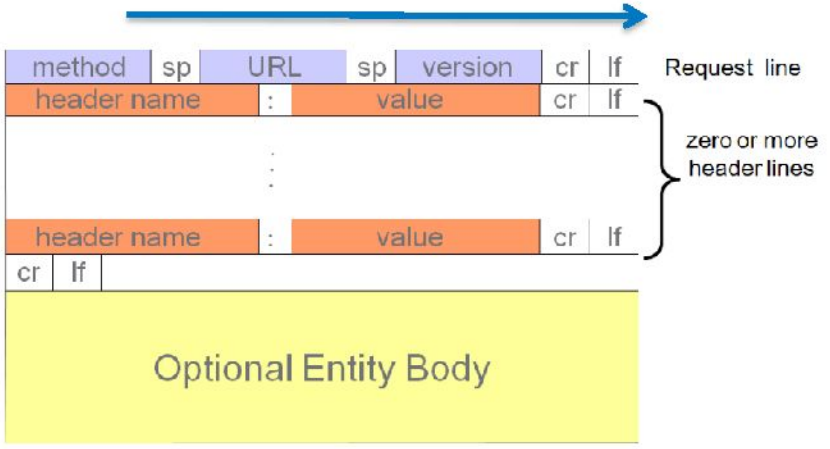
- XML – язык разметки
- XML отделяет хранение данных от их представления (HTML)
- XML хранит данные как plain text
- XML упрощает обмен данными
- На базе XML легко создаются новые языки
- XML-документ имеет древовидную структуру (root element, child element)
- элемент XML состоит из:
  - названия элемента;
  - контента;
  - возможно, атрибута элемента;
  - возможно, значения атрибута элемента

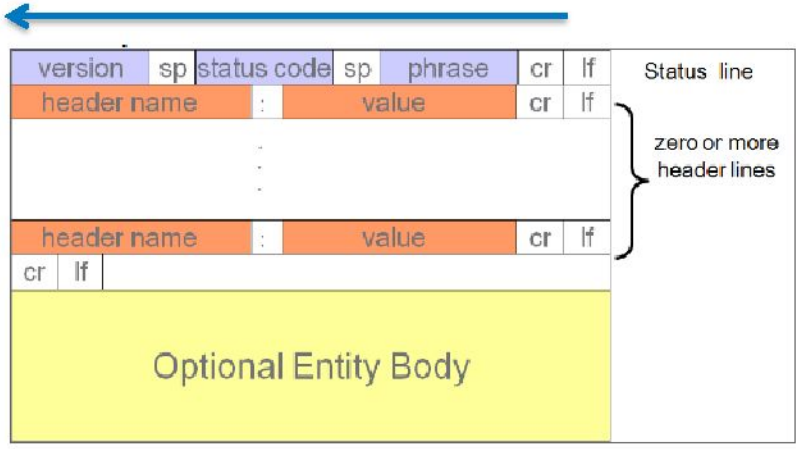
```
<?xml version = “1.0”>
<!-- Это пример использования XML
для хранения данных DB -->
<item number="00001">
 <name>
  <first>Jane</first>
  <middle>Q</middle>
  <last>Public</last>
 </name>
 <phone type="voice">
  <areacode>407</areacode>
  <number>555-1212</number>
 </phone>
 <phone type="fax">
  <areacode>407</areacode>
  <number>555-1213</number>
 </phone>
 <email>jpublic@gmail.com</email>
</item>
```

HTTP

Принцип работы протокола HTTP



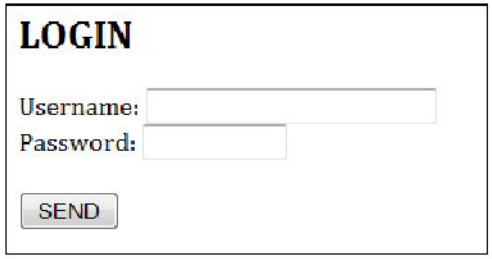




HTTP-запросы

HTTP Request

Web-страница



HTML-код web-страницы

```
<html>
<head><title>Login</title></head>
<body>
  <h2>LOGIN</h2>
  <form method="    " action="/bin/login">
    Username: <input type="text" name="user" size="25" /><br />
    Password: <input type="password" name="pw" size="10" /><br /><br />
    <input type="hidden" name="action" value="login" />
    <input type="submit" value="SEND" />
  </form>
</body>
</html>
```

post или get?

HTTP GET request

```
GET /bin/login?user=Peter+Lee&pw=123456&action=login HTTP/1.1
Accept: image/gif, image/jpeg, */*
Referer: http://127.0.0.1:8000/login.html
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Host: 127.0.0.1:8000
Connection: Keep-Alive
```

HTTP POST request

```
POST /bin/login HTTP/1.1
Host: 127.0.0.1:8000
Accept: image/gif, image/jpeg, */*
Referer: http://127.0.0.1:8000/login.html
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Content-Length: 37
Connection: Keep-Alive
Cache-Control: no-cache

User=Peter+Lee&pw=123456&action=login
```

HTTP-ответы

HTTP Response

```
GET /index.html HTTP/1.1
Host: 127.0.0.1
(blank line)
```

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 12:10:12 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>
```

```
GET /index.html HTTP/1.1
(blank line)
```

```
HTTP/1.1 400 Bad Request
Date: Sun, 18 Oct 2009 12:13:46 GMT
Server: Apache/2.2.14 (Win32)
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>
```

Status codes

Status code	Definition	Example
1xx	indicates an informational message	101 Switching Protocols
2XX	indicates success of some kind	200 OK
3xx	redirects the client to another URL	301 Moved permanently 302 Moved temporarily
4xx	indicates an error on the client's part	400 Bad request (bad syntax mostly) 401 Unauthorized (e.g. wrong user/pass) 403 Forbidden (e.g. not allowed client) 404 Not found
5xx	indicates an error on the server's part	500 Internal server error

URL

URL = Unified Resource Locator

URL pattern:

```
protocol://domain:port/requestURI#anchor?parameters&parameter
```

protocol	in our case HTTP
domain	server location (e.g www.NBA.com)
port	the port that the web server listen to (default-80)
request_URI	web server resource (e.g. index.html)
#anchor	This is used in order to define a link that takes the user to the middle of a page.
?parameter	These are extra values that are passed along with the path (e.g. username=JohnnyCash).
&	is used to concatenate several parameters (e.g. arg1=value1&arg2=value2)

Example:

Q&A

