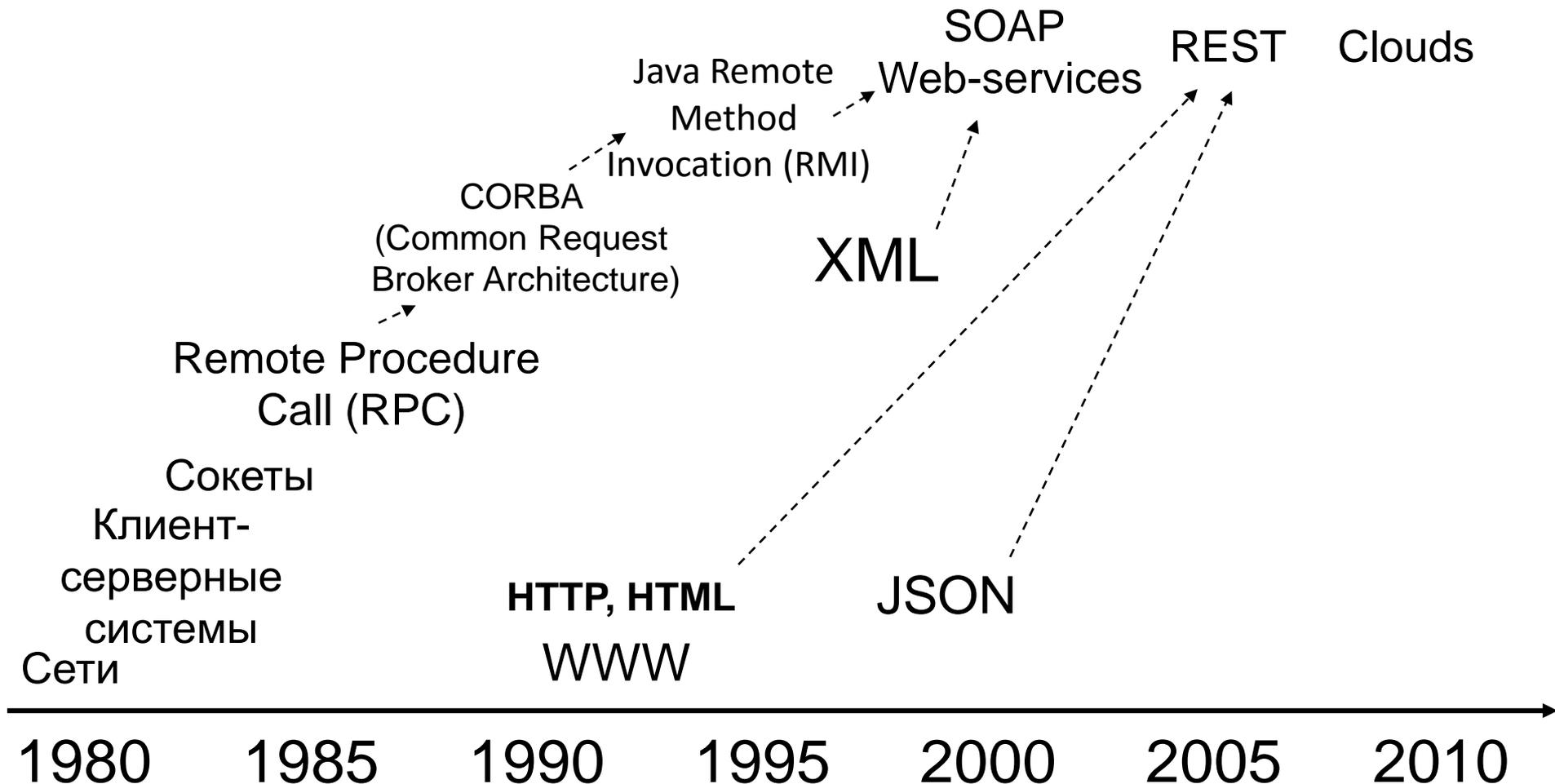


# РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Сервис-ориентированная архитектура

# СТАНДАРТЫ, ПРОТОКОЛЫ, АРХИТЕКТУРЫ РВС

2



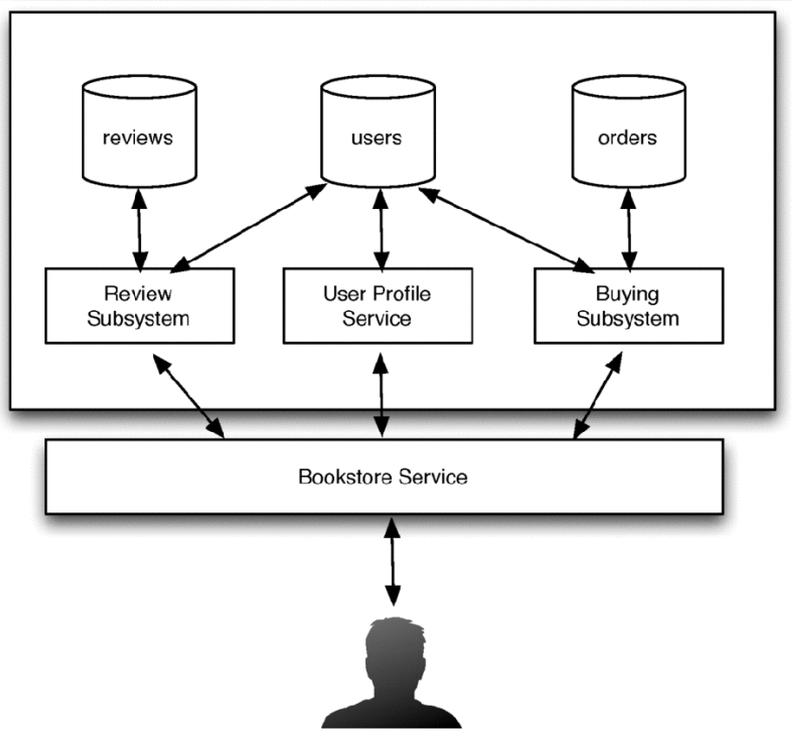
# ВВЕДЕНИЕ: ПИСЬМО ДЖЕФФА БЕЗОСА (2002)

- ⦿ Все команды с настоящего момента обязаны предоставлять данные и функциональность через сервисные интерфейсы.
- ⦿ Команды должны взаимодействовать друг с другом посредством этих интерфейсов
- ⦿ Другие методы меж-процессной коммуникации запрещены: никакого прямого доступа к чужой оперативной памяти, никакого прямого доступа к чужим хранилищам данных и др. Всё взаимодействие – только через сервисные интерфейсы по сети.
- ⦿ Не важно, какую технологию вы используете (HTTP, Corba...)
- ⦿ Все интерфейсы сервисов должны быть спроектированы таким образом, чтобы быть расширяемыми. Команды должны разрабатывать интерфейсы так, чтобы сервисы можно было предоставить потребителям из внешнего мира. Без исключений.
- ⦿ Любой, кто не следует этим правилам будет уволен.
- ⦿ Спасибо, приятного дня!

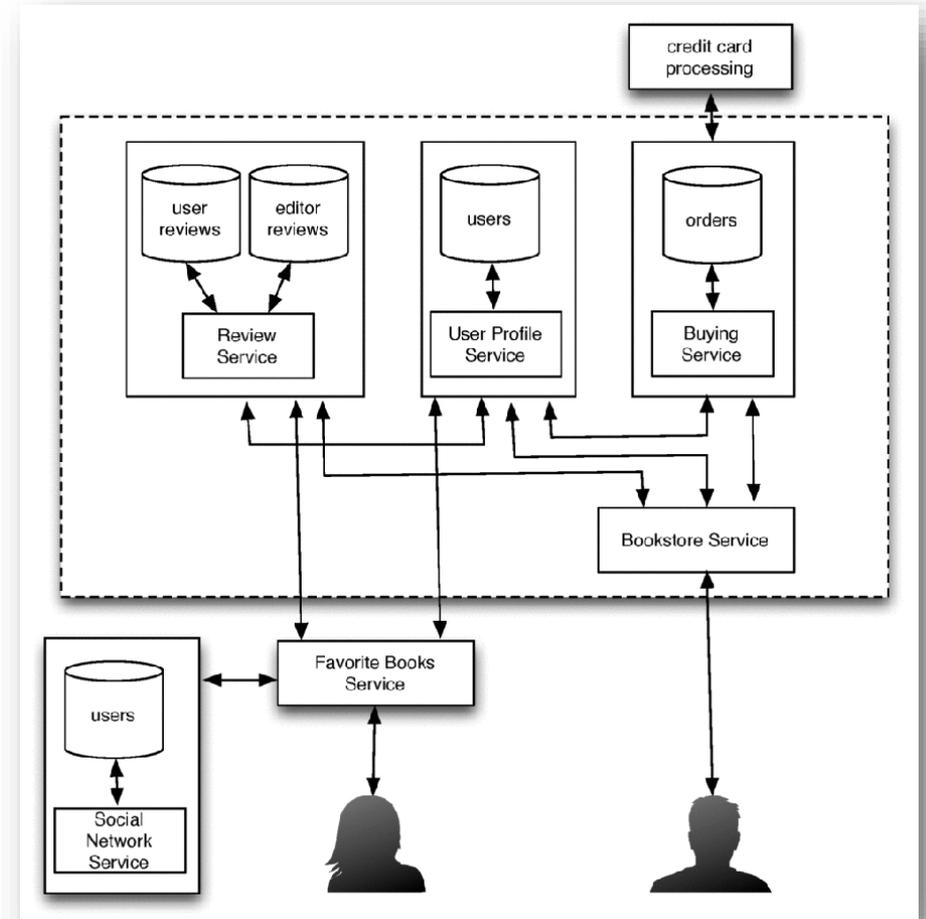
# FACEBOOK (2007) И GOOGLE+

- ⦿ Аналогичный переход сделал Facebook в 2007 году
- ⦿ Была запущена Facebook Platform, позволив сторонним программистам разрабатывать приложения, которые могли получать доступ к социальному графу и другим данным пользователей в Facebook.
- ⦿ Google+ очень долго ругали за отсутствие API, но когда он был выпущен в сентябре 2011 года он оказался «монолитным» и абсолютно не «SOA-like», обеспечивая лишь простейший доступ к профилю пользователя и всему, что он видит.

# ХРАНИЛИЩЕ VS SOA



«Хранилище»



SOA

# СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА РАСПРЕДЕЛЕННЫХ СИСТЕМ

# СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА (SOA)

- ◎ **Сервис ориентированная архитектура** – это парадигма организации и использования распределенных функциональных возможностей, которые могут предоставляться различными владельцами
  - ◎ Определение организации OASIS (Organization for the Advancement of Structured Information Standards)
- ◎ По существу, SOA означает, что *компоненты приложения* представляют собой взаимодействующие *сервисы*, которые могут быть использованы *независимо* друг от друга, или же вообще могут быть *отделены* для реализации других приложений.

# Стили API ВЕБ-СЕРВИСОВ

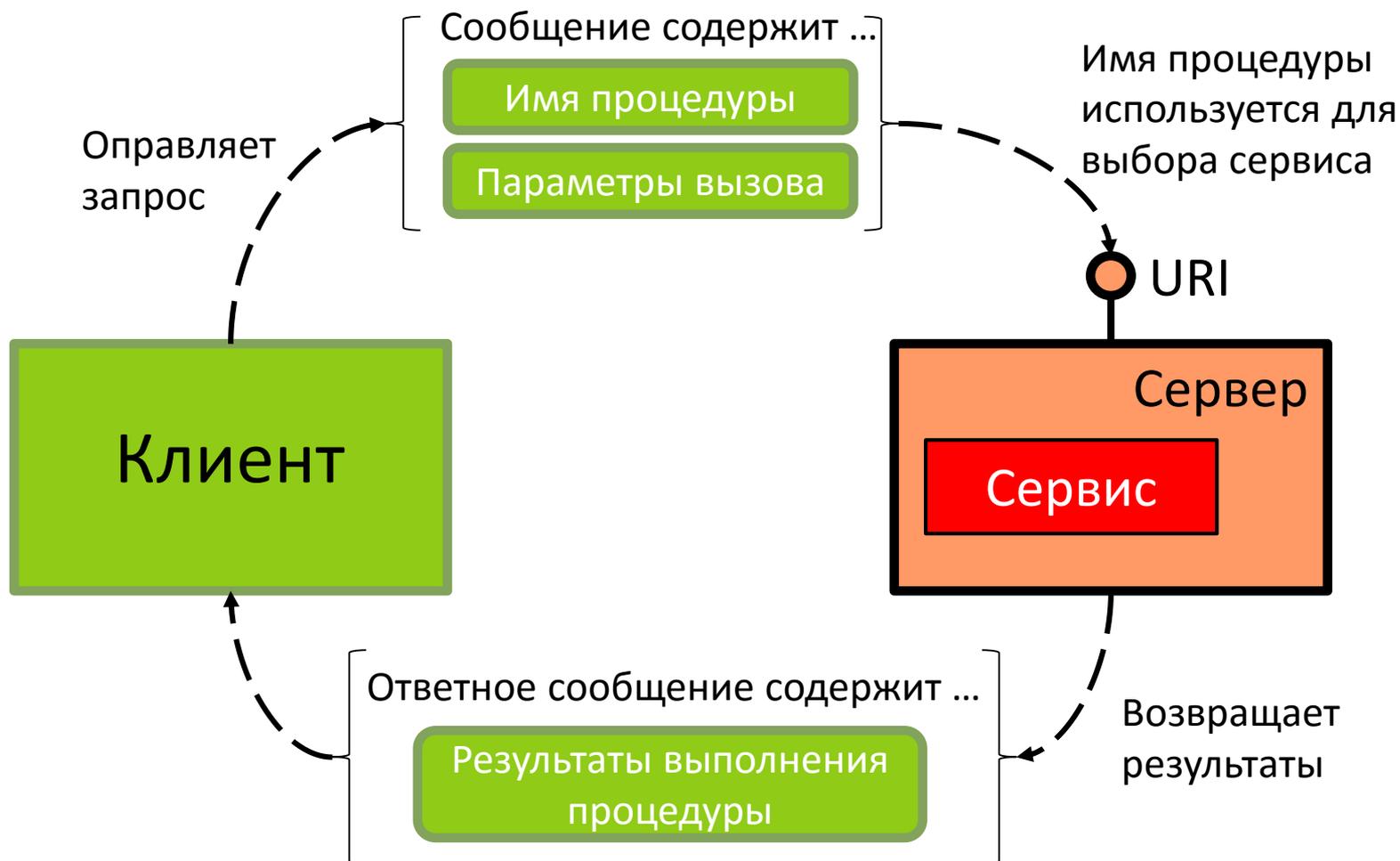
Стиль Веб-сервиса	Проблема
RPC API	Как клиенты могут выполнить удаленные процедуры посредством HTTP?
API сообщений	Как клиенты могут отправлять команды, нотификации или другую информацию удаленной системе по HTTP, но не привязывая себя к удаленным процедурам?
API ресурсов	Как клиент может манипулировать данными, предоставляемыми удаленной системой по HTTP, без связывания себя с удаленными процедурами, а также без необходимости создания специального проблемно-ориентированного API?

# RPC API

# Стили API ВЕБ-СЕРВИСОВ: RPC API

- ⊙ Необходимо определить сообщения, которые бы описывали:
  - ⊙ *удаленные процедуры,*
  - ⊙ *а также набор элементов, которые формируют набор параметров для удаленной процедуры.*
- ⊙ Клиент должен отправить сообщение с этой информацией по указанному URI для выполнения процедуры.

# Стили API ВЕБ-СЕРВИСОВ: RPC API



# ОБМЕН СООБЩЕНИЯМИ

- ⊙ Использование низкоуровневого транспортного протокола для обмена сообщениями (BSD Socket API) приводило к необходимости сложной трансформации данных, чтобы типы данных, определенные у клиента можно было использовать на сервере.
  - ⊙ *На разных платформах могут использоваться различные методы кодировки данных (ASCII , EBCDIC , UTF-8, UTF-16, Endianness ) для представления и хранения данных.*
- ⊙ Технологии CORBA и DCOM решили предложили решения для данной проблемы, посредством собственных абстракций над низкоуровневыми типами данных.
- ⊙ Но появились проблемы при связи между различными инфраструктурами (CORBA и .NET) + обмен сообщениями через Интернет (т.к. часто на брандмауэре закрыты все порты, кроме 80-го).

- ◎ Применение протокола HTTP решает многие из этих проблем, т.к. обеспечивает возможность взаимодействия клиентов и серверов, базирующихся на различных платформах через Интернет.
- ◎ Для организации RPC посредством HTTP существует множество реализаций на разных языках:
  - ◎ JAX-WS (Java)
  - ◎ WCF (Microsoft .NET)
- ◎ Они позволяют создавать простые веб-процедуры и клиентов к ним без необходимости понимания форматов данных (XML, JSON), методов кодирования (UTF-8 и др.) или описания структуры сервисов (WSDL).

# МЕТОДЫ ОРГАНИЗАЦИИ RPC

- ◎ Есть 2 основных подхода для описания контрактов таких сервисов:
  - ◎ *XML Веб-сервисы*: наиболее стандартный подход, использующий язык WSDL (Web Services Description Language) и спецификации WS-Policy и WS-Security для определения различных методов авторизации, шифрования и др.
  - ◎ Не-XML Веб-сервисы (RPC): это целый спектр различных протоколов RPC, начиная со стандарта JSON-RPC и заканчивая множеством проприетарных или открытых стандартов от различных поставщиков.

- ◎ JSON-RPC (JavaScript Object Notation Remote Procedure Call — JSON-вызов удаленных процедур) — протокол удаленного вызова процедур, использующий JSON для кодирования сообщений.
- ◎ JSON-RPC работает отсылая запросы к серверу, реализующему протокол. Клиентом обычно является программа, которой нужно вызвать метод на удаленной системе. Все передаваемые данные — простые объекты, сериализованные в JSON

# JSON-RPC - ЗАПРОС И ОТВЕТ

- ⊙ **Запрос** должен содержать три обязательных свойства:
  - ⊙ *method* — Строка с именем вызываемого метода.
  - ⊙ *params* — Массив объектов, которые должны быть переданы методу, как параметры.
  - ⊙ *id* — Значение любого типа, которое используется для установки соответствия между запросом и ответом.
  
- ⊙ **Ответ** должен содержать следующие свойства:
  - ⊙ *result* — Данные, которые вернул метод. Если произошла ошибка во время выполнения метода, это свойство должно быть установлено в null.
  - ⊙ *error* — Код ошибки, если произошла ошибка во время выполнения метода, иначе null.
  - ⊙ *id* — То же значение, что и в запросе, к которому относится данный ответ.

# JSON-RPC - ПРИМЕР

--> обозначает данные, отправленные серверу (запрос)

<-- обозначает ответ

```
...
--> {"method": "postMessage", "params": ["Hello all!"], "id": 99}
<-- {"result": 1, "error": null, "id": 99}
<-- {"method": "handleMessage", "params": ["user1", "Ok!"], "id": null}
<-- {"method": "handleMessage", "params": ["user3", "gotta go"], "id": null}
--> {"method": "postMessage", "params": ["I have a question:"], "id": 101}
<-- {"method": "userLeft", "params": ["user3"], "id": null}
<-- {"result": 1, "error": null, "id": 101}
...
```

# РЕАЛИЗАЦИЯ JSON-RPC

Название	версия JSON-RPC	Описание	Язык(и), Платформы
<a href="#">JSON-RPC.NET</a>	2.0	Быстрый JSON-RPC сервер. Поддерживает сокет, именованные сокеты и HTTP с помощью ASP.NET требует Mono или .NET Framework 4.0.	.NET
<a href="#">Jayrock</a>	1.0	Серверная реализация JSON-RPC 1.0 для версий 1.1 и 2.0 Microsoft .NET Framework.	.NET
<a href="#">jsonrpc-c</a>	2.0	Реализация JSON-RPC через TCP сокеты (только сервер).	C
<a href="#">libjson-rpc-cpp</a>	2.0	C++ JSON-RPC фреймворк, поддерживающий клиентскую и серверную стороны через HTTP.	C++
<a href="#">Phobos</a>	2.0	Реализация для <b>Q/C++</b> . Абстрагирует уровень передачи данных (готовые к использованию классы для TCP и HTTP).	C++
<a href="#">qjsonrpc</a>	2.0	Реализация для <b>Q/C++</b> . Поддерживает соединения между сообщениями и QObject слотами (как QDBus, qxtRpc) использует новые JSON классы, включенные в Qt5.	C++
<a href="#">JSON Toolkit</a>	2.0	Реализация на Delphi	Delphi
<a href="#">go/net/rpc</a>	?	Реализация JSON-RPC стандартной библиотеки Go	Go
<a href="#">jsonrpc4j</a>	2.0	Java реализация JSON-RPC 2.0 поддерживает как сокеты, так и HTTP соединение.	Java
<a href="#">json-rpc</a>	1.0	Базовая Java/JavaScript реализация, которая хорошо интегрируется в Android/Servlets/Standalone Java/JavaScript/App-Engine приложения.	Java / JavaScript
<a href="#">jproxy</a>	2.0	Простая Java реализация JSON-RPC созданная для упрощения реализации доступа к POJOs через сырой RPC фреймворк.	Java
<a href="#">JSON Service</a>	2.0	JSON-RPC серверная реализация поддержки Service Mapping Description. Хорошо интегрируется с <b>Dojo Toolkit</b> и <b>Spring Framework</b> .	Java
<a href="#">JSON-RPC 2.0</a>	2.0	Легкая Java library для разбора и сериализации JSON-RPC 2.0 сообщений (open source). Несколько реализация на сайте. (Base, Client, Shell, ...)	Java
<a href="#">java-json-rpc</a>	2.0	Реализация для J2EE серверов.	Java
<a href="#">lib-json-rpc</a>	2.0	Реализация servlet, client, JavaScript	Java
<a href="#">simplejsonrpc</a>	2.0	Простой JSON-RPC 2.0 Servlet, обслуживающий методы класса.	Java
<a href="#">gson-rmi</a>	2.0	Легковесный, независимый от способа передачи RMI фреймворк разработанный для распределенных вычислений.	Java
<a href="#">jsonrpcjs</a>	2.0	JavaScript клиентская библиотека для JSON-RPC 2.0. Не имеет зависимостей.	JavaScript
<a href="#">easyXDM</a>	2.0	Библиотека для cross-domain соединений с поддержкой RPC. Поддерживает все браузеры: postMessage, nix, frameElement, window.name, и FIM, очень проста в использовании.	JavaScript
<a href="#">Dojo Toolkit</a>	1.0+	Предоставляет поддержку JSON-RPC	JavaScript
<a href="#">Pmrpc</a>	2.0	JavaScript библиотека для использования в HTML5 браузерах. Реализация JSON-RPC, используя HTML5 postMessage API для передачи сообщений.	JavaScript
<a href="#">qooxdoo</a>	2.0	Имеет JSON-RPC реализацию с опциональными бэк-эндами на Java, PHP, Perl и Python.	JavaScript, Java, PHP, PERL, & Python
<a href="#">JSON-RPC Реализация на JavaScript</a>	2.0	Поддерживает JSON-RPC через HTTP и TCP/IP.	JavaScript
<a href="#">jabsorb</a>	2.0	Легковесный Ajax/Web 2.0 JSON-RPC Java фреймворк, расширяющий JSON-RPC протокол дополнительной <b>ORB</b> функциональностью, такой как поддержка циклических зависимостей.	JavaScript, Java
<a href="#">The Wakanda platform</a>	2.0	Поддерживает JSON-RPC 2.0 клиент внутри Ajax Framework и JSON-RPC 2.0 сервис в серверном JavaScript	JavaScript
<a href="#">Deimos</a>	1.0+2.0	Серверная реализация для Node.js/JavaScript.	JavaScript
<a href="#">Barracuda Web Server</a>	2.0	Barracuda Web Server's интегрированный	JavaScript
<a href="#">DeferredKit</a>	1.0	Поддерживает JSON-RPC 1.0 клиент.	Lua
<a href="#">Demiurgic</a>	2.0	JSON-RPC 2.0 клиент для Objective-C	Objective-C
<a href="#">Oxen iPhone Commons JSON components</a>	1.0	JSON-RPC 1.0 клиент для Objective-C	Objective-C
<a href="#">objc-JSONRpc</a>	2.0	Objective-c JSON RPC клиент. Поддерживает уведомления, простые вызовы и множественные вызовы.	Objective-C
<a href="#">JSON-RPC</a>	2.0	Реализация сервера JSON RPC 2.0	Perl
<a href="#">json-rpc-perl6</a>	2.0	Клиент и сервер.	Perl 6
<a href="#">php-json-rpc</a>	2.0	Простая PHP реализация JSON-RPC 2.0 через HTTP клиента.	PHP
<a href="#">jQuery JSON-RPC Server</a>	2.0	JSON-RPC сервер, специально сделанный для работы с Zend Framework JSON RPC Server.	PHP, JavaScript
<a href="#">jsonrpc2php</a>	2.0	PHP JSON-RPC 2.0 базовый класс и пример сервера	PHP
<a href="#">tivoka</a>	1.0 + 2.0	Универсальный клиент/серверная JSON-RPC библиотека для PHP 5+.	PHP
<a href="#">junior</a>	2.0	Client/server библиотека для JSON-RPC 2.0	PHP
<a href="#">json-rpc-php</a>	2.0	Client/server библиотека для JSON-RPC 2.0	PHP
<a href="#">JSONrpc2</a>	2.0	Реализация с «dot magic» для PHP (= поддержка группировки методов и разделения точками)	PHP
<a href="#">GetResponse jsonRPCClient</a>	2.0	Объектно-ориентированная реализация клиента	PHP
<a href="#">zoServices</a>	2.0	PHP, Node.js и JavaScript реализация JSON-RPC 2.0	PHP, JavaScript, Node.js
<a href="#">json-rpc2php</a>	2.0	Серверная и клиентская реализация для PHP. Содержит JavaScript клиент, использующий <a href="#">jQuery</a>	PHP, JavaScript
<a href="#">jsonrpcphp</a>	2.0	JSON-RPC реализация для PHP	PHP
<a href="#">php-json-rpc</a>	2.0	Реализация JSON-RPC 2.0.	PHP
<a href="#">Django JSON-RPC 2.0</a>	2.0	JSON-RPC сервер для Django	Python
<a href="#">Pyjamas</a>		JSON-RPC клиентская реализация.	Python
<a href="#">Zope 3</a>	1.1	JSON RPC реализация для Zope 3	Python
<a href="#">jsonrpclib</a>	2.0	JSON-RPC клиентский модуль для Python.	Python
<a href="#">tornadoRPC</a>	2.0	Поддерживает JSON-RPC требует <b>Tornado web server</b> .	Python
<a href="#">tinyrpc</a>	2.0	Поддерживает JSON-RPC через TCP, WSIG, ZeroMQ и др. Разделяет передачу данных от обработки сообщений, может работать без пересылки сообщений.	Python
<a href="#">jsonrpc</a>	2.0	JSON-RPC 2.0 для Python + Twisted.	Python
<a href="#">bjsonrpc</a>	1.0+	Реализация через TCP/IP (асинхронная, двунаправленная)	Python
<a href="#">Barrister RPC</a>	2.0	JSON-RPC реализация клиента и сервера	Python, Ruby, JavaScript (Node.js + web browser), PHP, Java
<a href="#">pyramid_rpc</a>	2.0	Гибкая JSON-RPC реализация интегрированная в Pyramid web application. Работает с Pyramid's системой авторизации.	Python
<a href="#">rj</a>	2.0	JSON-RPC через TCP/UDP, HTTP, WebSockets, AMQP, и прочие.	Ruby (EventMachine) сервер с Ruby и JavaScript клиентами.
<a href="#">jimson</a>	2.0	Клиент и сервер для Ruby	Ruby
<a href="#">JSON-RPC Objects</a>	1.0+	Реализация только объектов (без клиента и сервера).	Ruby
<a href="#">JSON-RPC RT</a>	2.0	Полная поддержка JSON-RPC 2.0 через TCP.	Windows Runtime (WinRT)
<a href="#">XINS</a>	2.0	С Версии 2.0, поддерживает JSON и JSON-RPC.	XML