Web Services, SOA and REST

Services, Clients, SOAP, WSDL, XML, HTTP, REST, RESTful Services, JSON

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

SOFTWARE UNIVERSITY FOUNDATION

Distributed Apps, Web Services and SOA

Distributed Applications

- Most modern applications are distributed
  - Several components interact with each other
- Distributed application models
  - "Client-Server" model – persistent socket / WebSocket connection
  - "Distributed Objects" model – client an server objects
    - DCOM, CORBA, Java RMI, .NET Remoting, …
    - Outdated, not used in modern apps
  - "Web Services" / "RESTful Web Services" model
    - RESTful (HTTP, REST, JSON) and heavy services (SOAP, WSDL, XML)

Services: Real World and Software

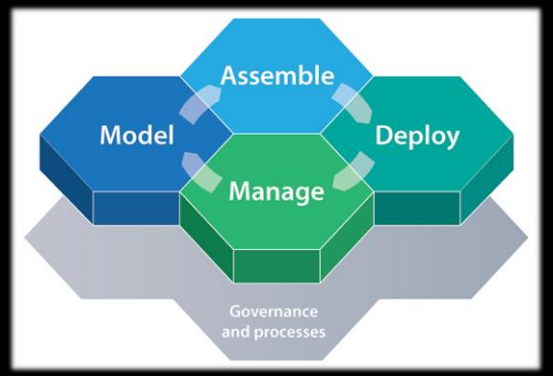
- In the real world a "**service**" is:
  - A piece of work performed by a service provider
  - Takes some input and produces some desired results
  - E.g. a supermarket: pay some money and get some food
  - Has quality characteristics (price, execution time, constraints, etc.)
- In the software world a "**service**"
  - Takes some input, performs some work, produces some output
  - Request-response model: client requests, server responses

Web Services and Clients

- A "**web service**" is:
  - Software service that communicates over standard Web protocols
  - Classical (heavyweight) services use SOAP, WSDL, XML, WS-*
  - Lightweight (RESTful) services use HTTP, REST and JSON
- A "**client**" (consumer) uses the services
  - Requests something to be performed
  - Gets the desired result
  - Or gets an error

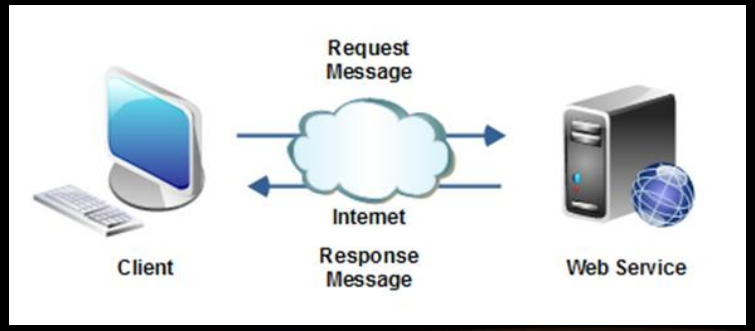
What is Service-Oriented Architecture (SOA)?

- **SOA** (**S**ervice-**O**riented **A**rchitecture) is an architectural concept for development of software systems
  - Using reusable building blocks (components) called "services"
  - SOA == decouple the monolithic software to reusable services
- Services in SOA are:
  - Autonomous, stateless business functions
  - Accept requests and return responses
  - Use well-defined, standard interface (standard protocols)



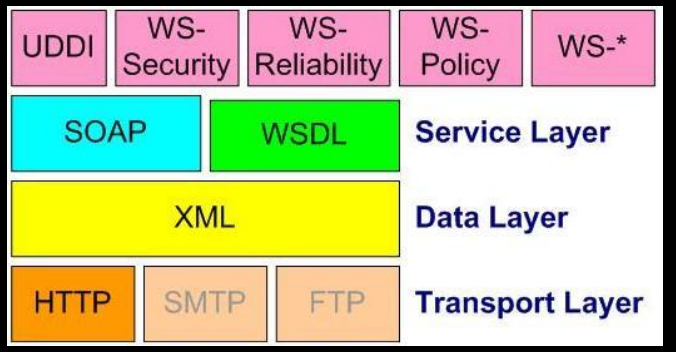
SOA Services

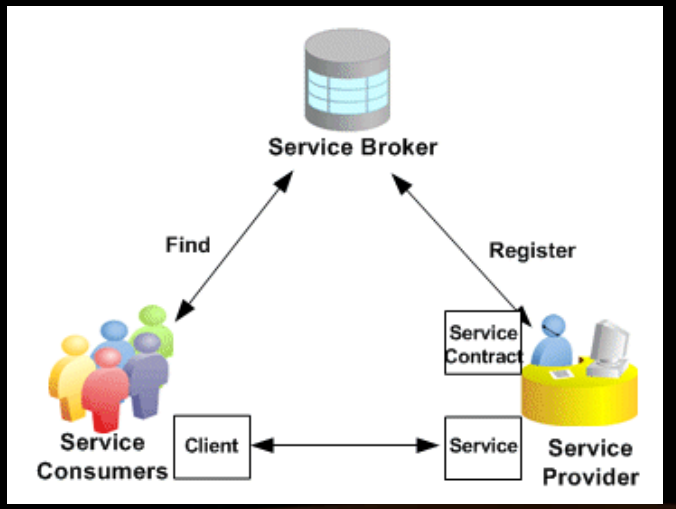
- Autonomous
  - Each service operates autonomously
  - Without any awareness that other services exist
- Stateless
  - Do not remember a durable state between requests
    - Can store state in a database and reference it by ID
  - Easy to scale → just add more nodes
- Request-response model
  - Client asks, server returns an answer
  - Server never sends requests to the client



SOA Services (2)

- Communication through standard protocols
  - HTTP, FTP, SMTP, RPC, MSMQ, ...
  - JSON, XML, SOAP, RSS, WS-*, ...
- Platform independent
  - Independent of OS, platforms, languages, frameworks, …
- Discoverable
  - Service registries and brokers



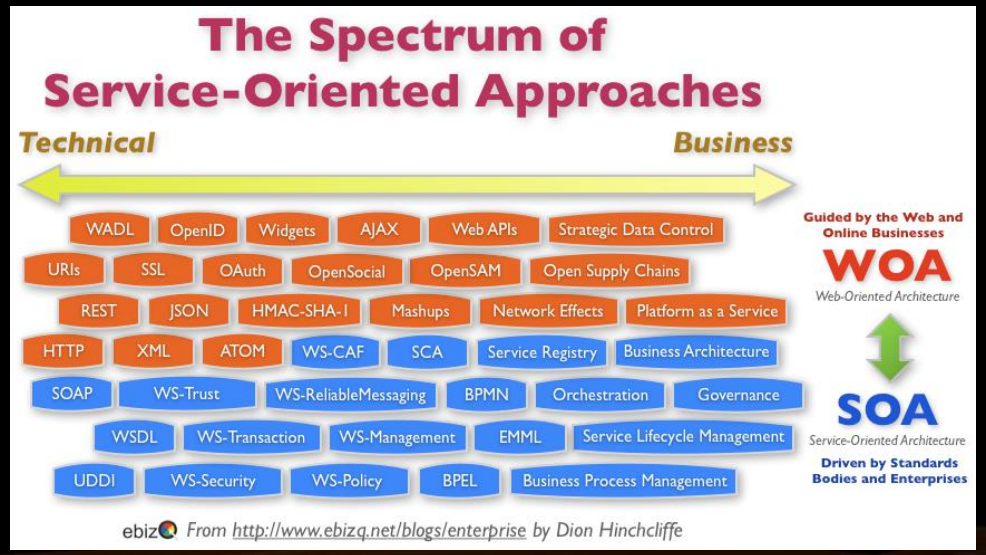


Lightweight SOA (SOA in Internet)

- Internet companies implement lightweight SOA in Internet
  - Also called WOA (Web-Oriented Architecture)
  - Examples: Google, Amazon, Facebook, Twitter, Parse.com, ...
  - Based on lightweight Web standards:
    - AJAX and Rich Internet Applications (RIA)
    - REST, JSON, RSS, XML, proprietary APIs
- RESTful Web services == lightweight Web services
  - Use simple HTTP requests and simple JSON responses

Heavyweight SOA (SOA in Enterprises)

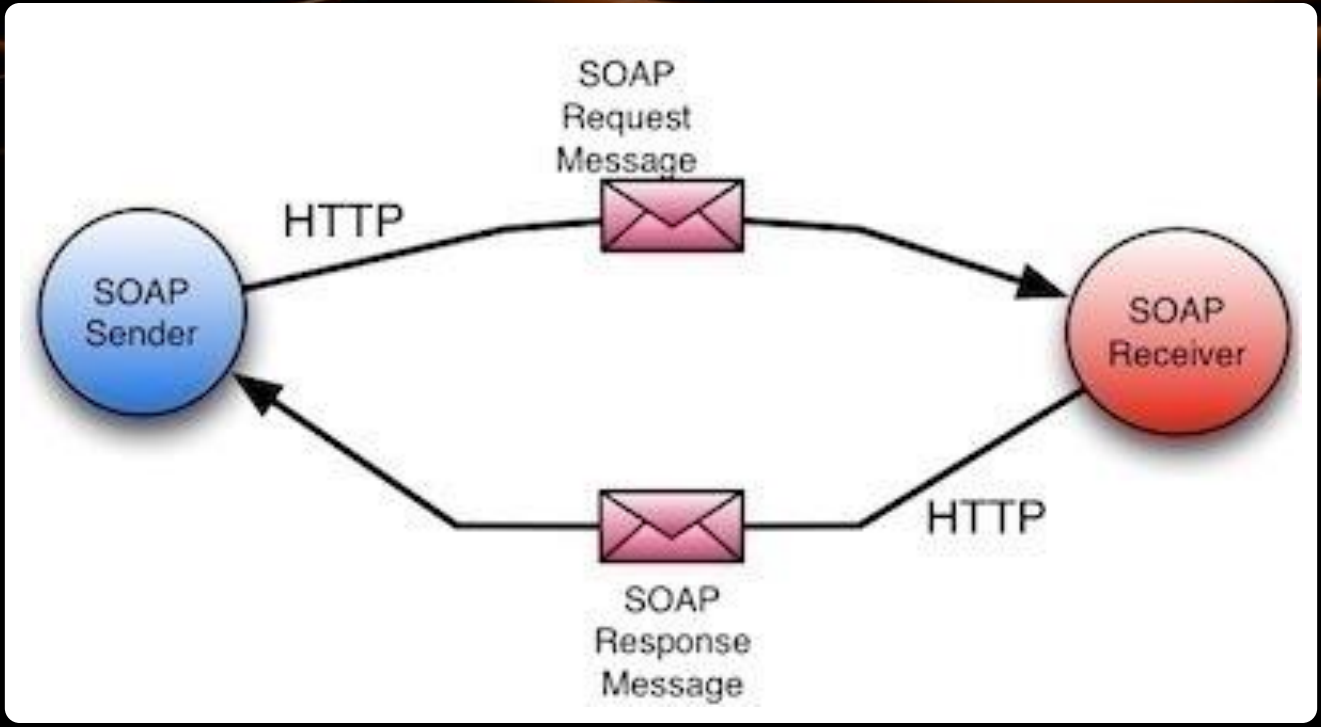
- **Heavyweight SOA** stacks
  - Driven by business processes: BPM, BPMN, BPEL, ...
  - Enterprise application integration (EAI)
  - B2B integration and SOA based portals
  - Unified Frameworks: SCA and WCF
  - Enterprise Service Bus (ESB)
  - SOA governance (control)
  - Many public standards like **WS-***



Web Service Standards: WS-*

https://www.oasis-open.org/standards

- Service Discovery Standards
  - UDDI, RDDL, XRI, XRDS
- Service Messaging Standards
  - SOAP, SOAP over JMS, MTOM, WS-Addressing
- Service Meta-Data Standards
  - WSBPEL, WSDL, WADL, WSFL, WS-Policy, WS-PolicyAssertions, WS-PolicyAttachment, WS-MetadataExchange (WS-MEX)
- Web Service Security Standards
  - XML-Signature, WS-SecurityPolicy, WS-Security, WS-Trust, WS-SecureConversation
- Quality of Service Standards
  - WS-ReliableMessaging (WS-RM), WS-Coordination, WS-AtomicTransactions, WS-TX

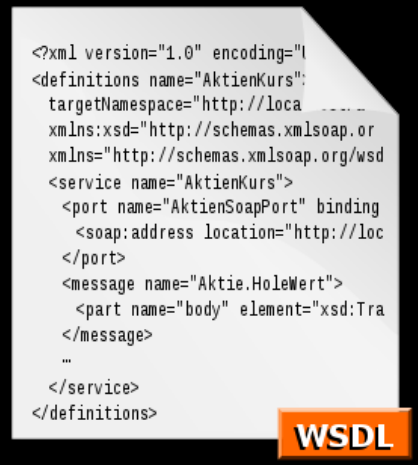


Enterprise Web Service Infrastructure

SOAP / WSDL / XML / HTTP

Heavyweight Web Services Infrastructure

- Heavyweight (classical) Web service infrastructure components:
  - Description
    - WSDL (Web Service Definition Language)
  - Metadata
    - WS-MetadataExchange (WS-MEX), DISCO
  - Wire format
    - SOAP, XML, XSD
    - HTTP



WSDL Service Description (WSDL)

- **WSDL** (**W**eb **S**ervices **D**escription **L**anguage)
  - Describes what a Web service can do
    - Names of the available methods (messages)
    - Input and output parameters, returned value
    - Data types used for parameters or result
    - Endpoints: ports and bindings
  - WSDL is an XML based, open standard from W3C

WSDL – Example

```
<?xml version="1.0" encoding="utf-8"?>
<definitions
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:s="http://www.w3.org/2001/XMLSchema"
 xmlns:s0="http://www.devbg.org/ws/MathService"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 targetNamespace="http://www.devbg.org/ws/MathService"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

 <types> … </types>
 <message name="AddSoapIn"> … </message>
 <portType name="MathServiceSoap"> … </portType>
 <binding name="MathServiceSoap" … > … </binding>
 <service name="MathService"> … </service>
</definitions>
```

Discovery of Web Service

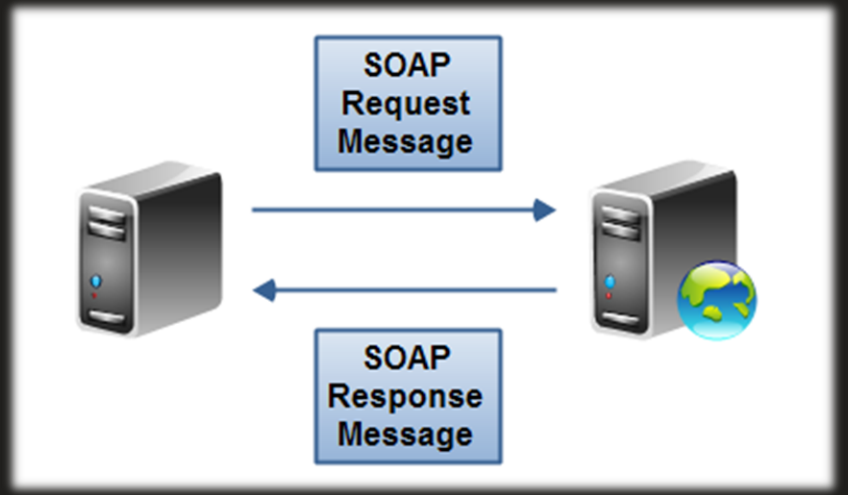
- The process of getting the service metadata (description)
- Usually a **URL** is interrogated to retrieve the metadata
- Two protocols for interrogation
  - **WS-MetadataExchange** (WS-MEX)
    - Standardized protocol developed by Microsoft, Sun, SAP, …
  - **DISCO**
    - Old Microsoft protocol to use with the UDDI registries

SOAP – Request / Result Format

- **SOAP** (**S**imple **O**bject **A**ccess **P**rotocol)
  - Open XML based format for sending messages
  - Open standard from W3C
- A **SOAP `message`** consists of:
  - **SOAP `header`** – describes the message parameters (metadata)
  - **SOAP `body`** – the message data (request or response body)
- Typically **SOAP** messages are sent over **HTTP**
  - Optionally TCP / message queue / other channels can be used

SOAP Request – Example

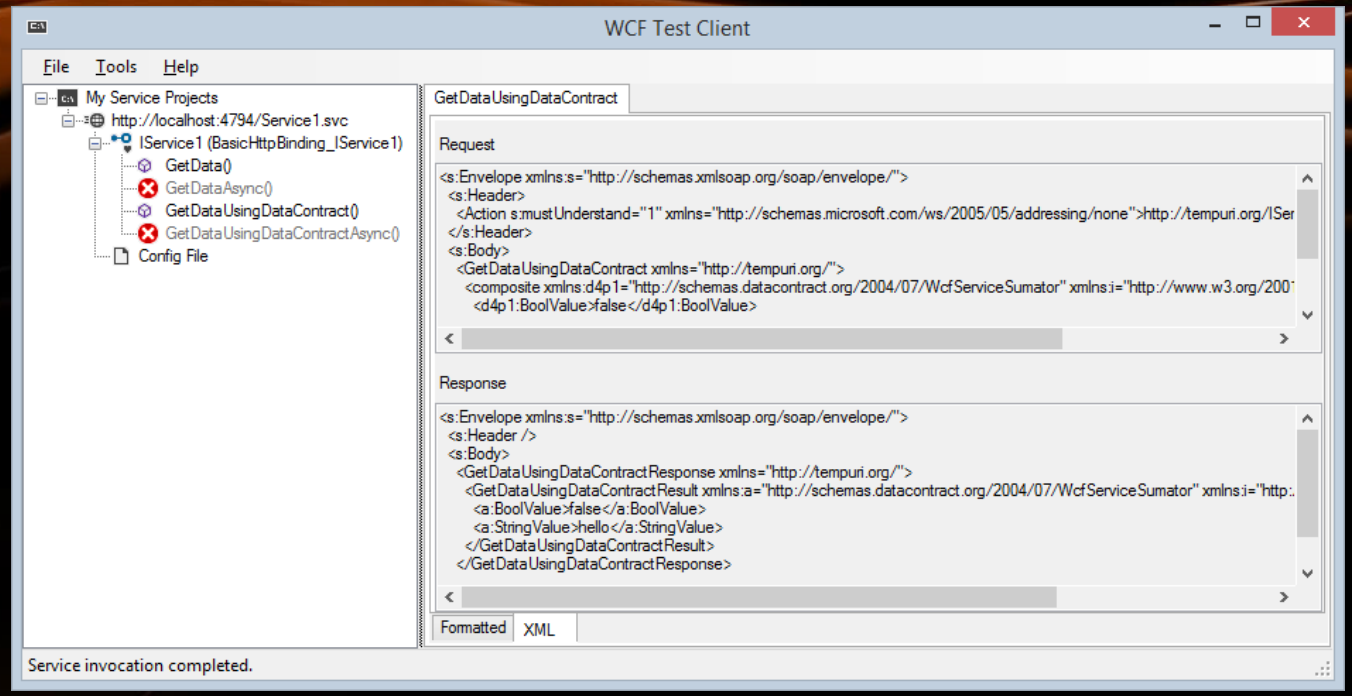
```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CalcDistance xmlns="http://www.devbg.org/Calc">
      <startPoint>
        <x>4</x> <y>5</y>
      </startPoint>
      <endPoint>
        <x>7</x> <y>-3</y>
      </endPoint>
    </CalcDistance>
  </soap:Body>
</soap:Envelope>
```

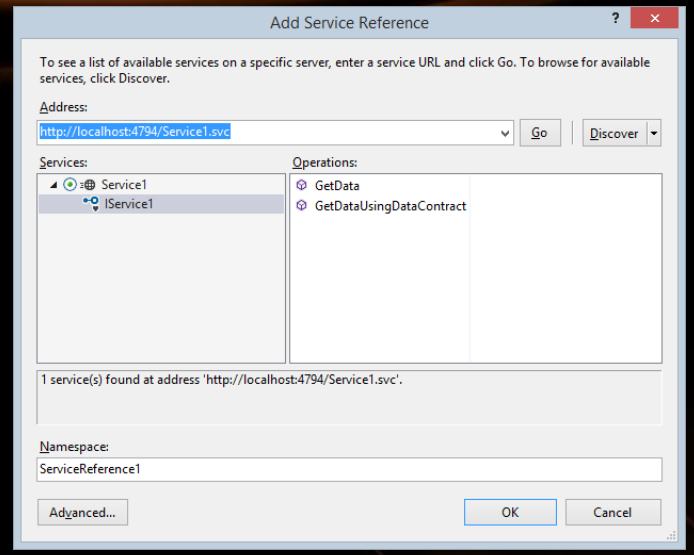


SOAP Response – Example

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CalcDistanceResponse
     xmlns="http://www.devbg.org/Calc/">
      <CalcDistanceResult>8.54400374531753</CalcDistanceResult>
    </CalcDistanceResponse>
  </soap:Body>
</soap:Envelope>
```

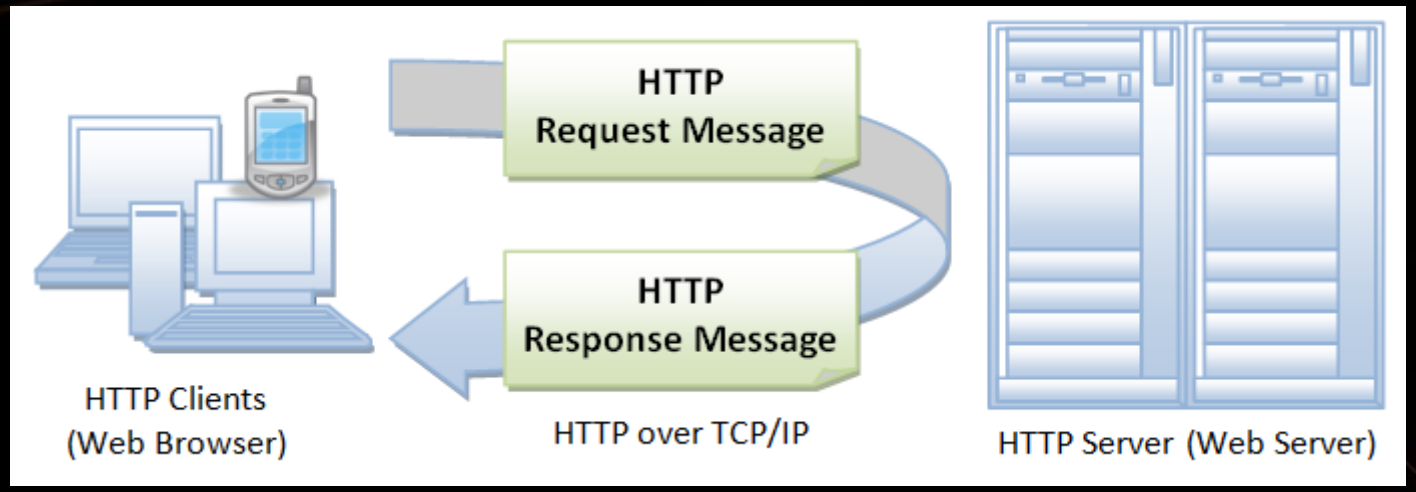
SOFTWARE UNIVERSITY FOUNDATION





Heavyweight Web Services (Based on SOAP and WSDL)

Live Demo



The HTTP Protocol

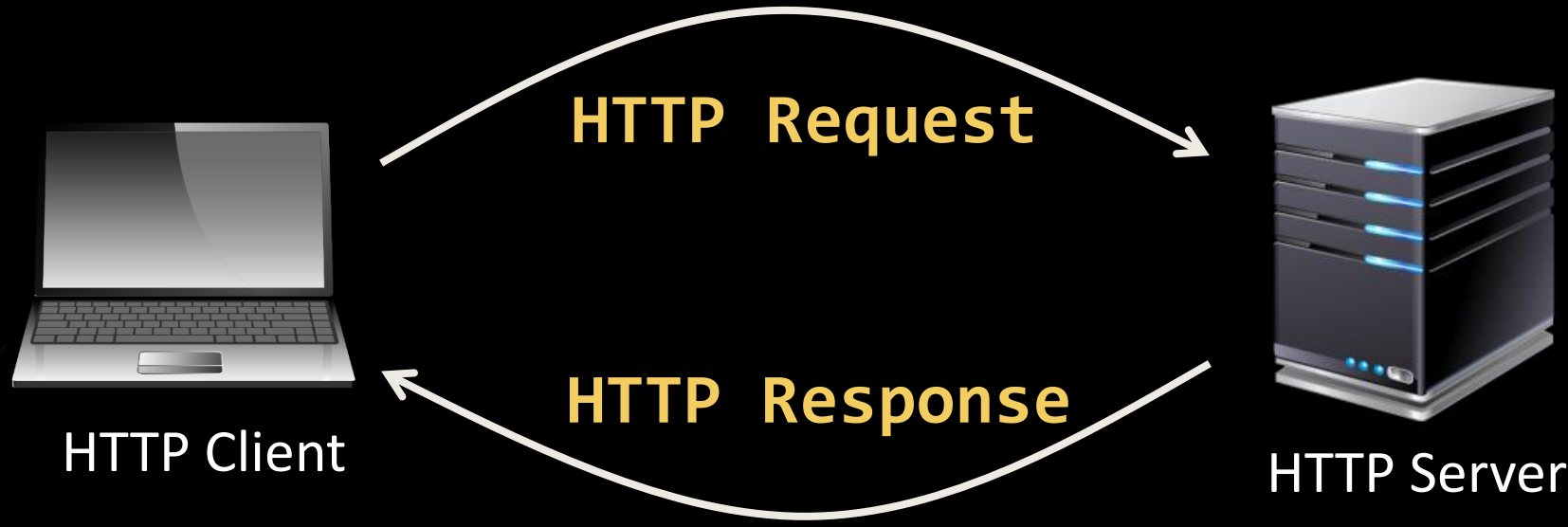
How HTTP Works?

HTTP

- HTTP == Hyper Text Transfer Protocol
  - Client-server protocol for transferring Web resources (HTML files, images, styles, scripts, data, etc.)
  - The widespread protocol for Internet communication today
  - Request-response model (client requests, server answers)
  - Text-based format (human readable)
  - Relies on unique resource URLs
  - Provides resource metadata (e.g. encoding)
  - Stateless (cookies and Web storages can overcome this)

HTTP: Request-Response Protocol

- Client program
  - Running at end host
  - Requests a resource
- Server program
  - Running at the server
  - Provides resources



HTTP Conversation: Example

- HTTP request:

```
GET /courses/javascript HTTP/1.1
Host: www.softuni.bg
User-Agent: Mozilla/5.0
<CRLF>
```

The empty line denotes the end of the request header

- HTTP response:

```
HTTP/1.1 200 OK
Date: Mon, 5 Jul 2010 13:09:03 GMT
Server: Microsoft-HTTPAPI/2.0
Last-Modified: Mon, 12 Jul 2014 15:33:23 GMT
Content-Length: 54
<CRLF>
<html><title>Hello</title>
Welcome to our site</html>
```

The empty line denotes the end of the response header

HTTP Request Methods

- HTTP defines request methods
  - Specify the action to be performed on the identified resource

Method	Description
GET	Retrieve a resource (execute query)
POST	Creates a resource
PUT	Modifies a resource
DELETE	Remove (delete) a resource
HEAD	Retrieve the resource's headers
OPTIONS	Requests communication options

HTTP Request Message

HTTP Request Message

- Request message sent by a client consists of:
  - HTTP request line
    - Request method (GET / POST / PUT / DELETE / …)
    - Resource URI (URL)
    - Protocol version
  - HTTP request headers
    - Additional parameters
  - HTTP body – optional data, e.g. posted form fields

```
<method> <resource> HTTP/<version>
<headers>
(empty line)
<body>
```

HTTP GET Request – Example

- Example of HTTP **GET** request:

```
GET /courses/javascript HTTP/1.1
Host: www.softuni.bg
Accept: */*
Accept-Language: bg
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0(compatible;MSIE 6.0;Windows NT 5.0)
Connection: Keep-Alive
Cache-Control: no-cache
<CRLF>
```

HTTP request line

HTTP request headers

The request body is empty

HTTP POST Request – Example

- Example of HTTP **POST** request:

```
POST /webmail/login.phtml HTTP/1.1
Host: www.abv.bg
Accept: */*
Accept-Language: bg
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0(compatible;MSIE 6.0; Windows NT 5.0)
Connection: Keep-Alive
Cache-Control: no-cache
Content-Length: 59
<CRLF>
username=mente&password=top*secret!
<CRLF>
```

HTTP request line

HTTP request headers

The request body holds the submitted form data

Conditional HTTP GET – Example

- Example of HTTP conditional GET request:

```
GET /apply HTTP/1.1
Host: www.softuni.bg
User-Agent: Gecko/20100115 Firefox/3.6
If-Modified-Since: Tue, 9 Mar 2015 11:12:23 GMT
<CRLF>
```

- Fetches the resource only if it has been changed at the server
  - Server replies with "304 Not Modified" if the resource has not been changed
  - Or "200 OK" with the latest version otherwise

HTTP Response Message

HTTP Response Message

- The response message sent by the HTTP server consists of:
  - HTTP response status line
    - Protocol version
    - Status code
    - Status phrase
  - Response headers
    - Provide meta data about the returned resource
  - Response body
    - The content of the HTTP response (data)

```
HTTP/<version> <status code> <status text>
<headers>
<CRLF>
<response body – the requested resource>
```

HTTP Response – Example

- Example of HTTP response from the Web server:

```
HTTP/1.1 200 OK
Date: Fri, 17 Jul 2010 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Accept-Ranges: bytes
Content-Length: 84
Content-Type: text/html
<CRLF>
<html>
  <head><title>Test</title></head>
  <body>Test HTML page.</body>
</html>
```

HTTP response status line

HTTP response headers

The HTTP response body

HTTP Response – Example

- Example of HTTP response with error result:

```
HTTP/1.1 404 Not Found
Date: Fri, 17 Nov 2014 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Connection: close
Content-Type: text/html
<CRLF>
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /img/logo.gif was not found on this server.<P>
<HR><ADDRESS>Apache/2.2.14 Server at Port 80</ADDRESS>
</BODY></HTML>
```

HTTP response status line

HTTP response headers

The HTTP response body

HTTP Response Codes

- HTTP response code classes
  - 1xx: informational (e.g., "100 Continue")
  - 2xx: successful (e.g., "200 OK", "201 Created")
  - 3xx: redirection (e.g., "304 Not Modified", "301 Moved Permanently", "302 Found")
  - 4xx: client error (e.g., "400 Bad Request", "404 Not Found", "401 Unauthorized", "409 Conflict")
  - 5xx: server error (e.g., "500 Internal Server Error", "503 Service Unavailable")

Content-Type and Content-Disposition

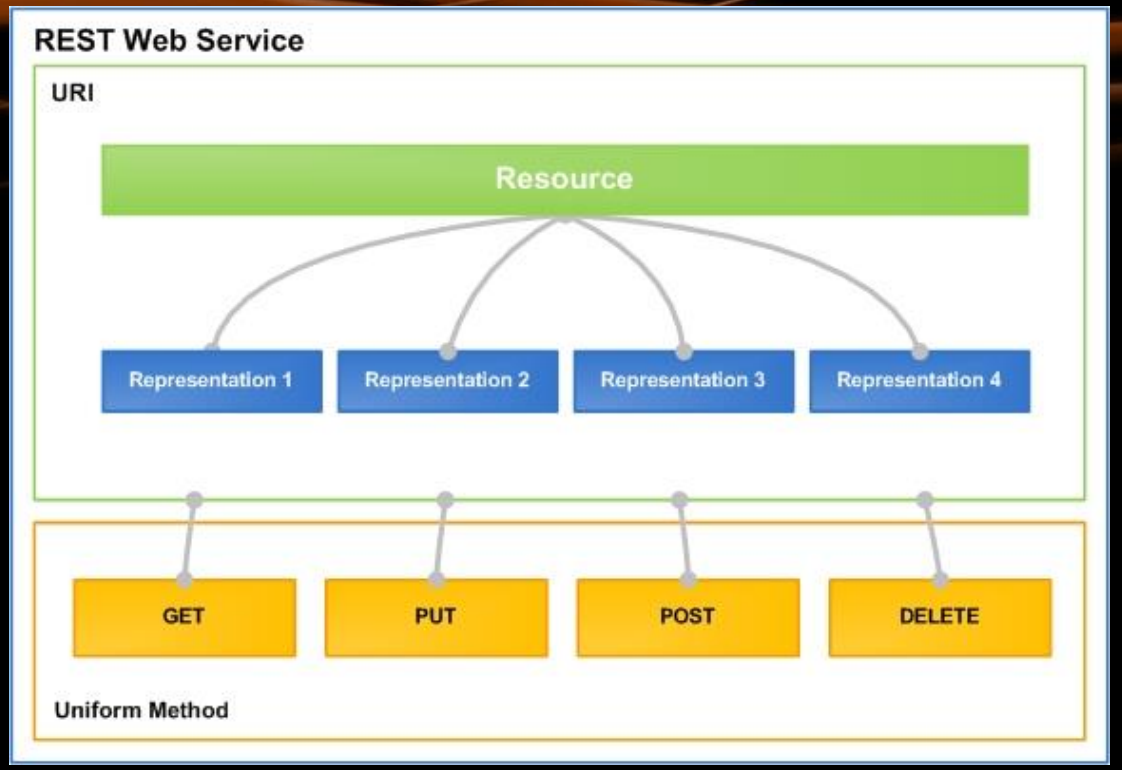
- The **`Content-Type`** response header the server specifies how the output should be processed
- Examples:

UTF-8 encoded HTML page; will be shown in the browser

```
Content-Type: text/html; charset=utf-8
```

```
Content-Type: application/pdf
Content-Disposition: attachment; filename="Report-April-2015.pdf"
```

This will download a PDF file named **`Financial-Report-April-2015.pdf`**



RESTful Web Services

Lightweight Architecture for Web Services

What is REST?

SOFTWARE UNIVERSITY FOUNDATION

"Representational State Transfer (REST) is a software architecture style consisting of guidelines and best practices for creating scalable Web services." http://en.wikipedia.org/wiki/Representational_State_Transfer

- Application state and functionality are resources
  - Every resource is associated with unique URI
  - Each resource supports standard operations (CRUD)
- This natively maps to the HTTP protocol
  - HTTP methods: **GET**, **POST**, **PUT**, **DELETE**, **PATCH**, **OPTIONS**, …

CRUD Operations in REST APIs

URL	HTTP Verb	POST Body	Result
http://yourdomain.com/api/entries	GET	empty	Returns all entries
http://yourdomain.com/api/entries	POST	JSON String	New entry Created
http://yourdomain.com/api/entries/:id	GET	empty	Returns single entry
http://yourdomain.com/api/entries/:id	PUT	JSON string	Updates an existing entry
http://yourdomain.com/api/entries/:id	DELETE	empty	Deletes existing entry

RESTful Web Services and HTTP Methods

- One URI per resource
  - Multiple operations per URI
- Get all resources / single resource by ID
  - GET http://myservice.com/api/Books
  - GET http://myservice.com/api/Books/3
- Add a new resource
  - POST http://myservice.com/api/Books
- Modify (update) a resource
  - PUT http://myservice.com/api/Books/3

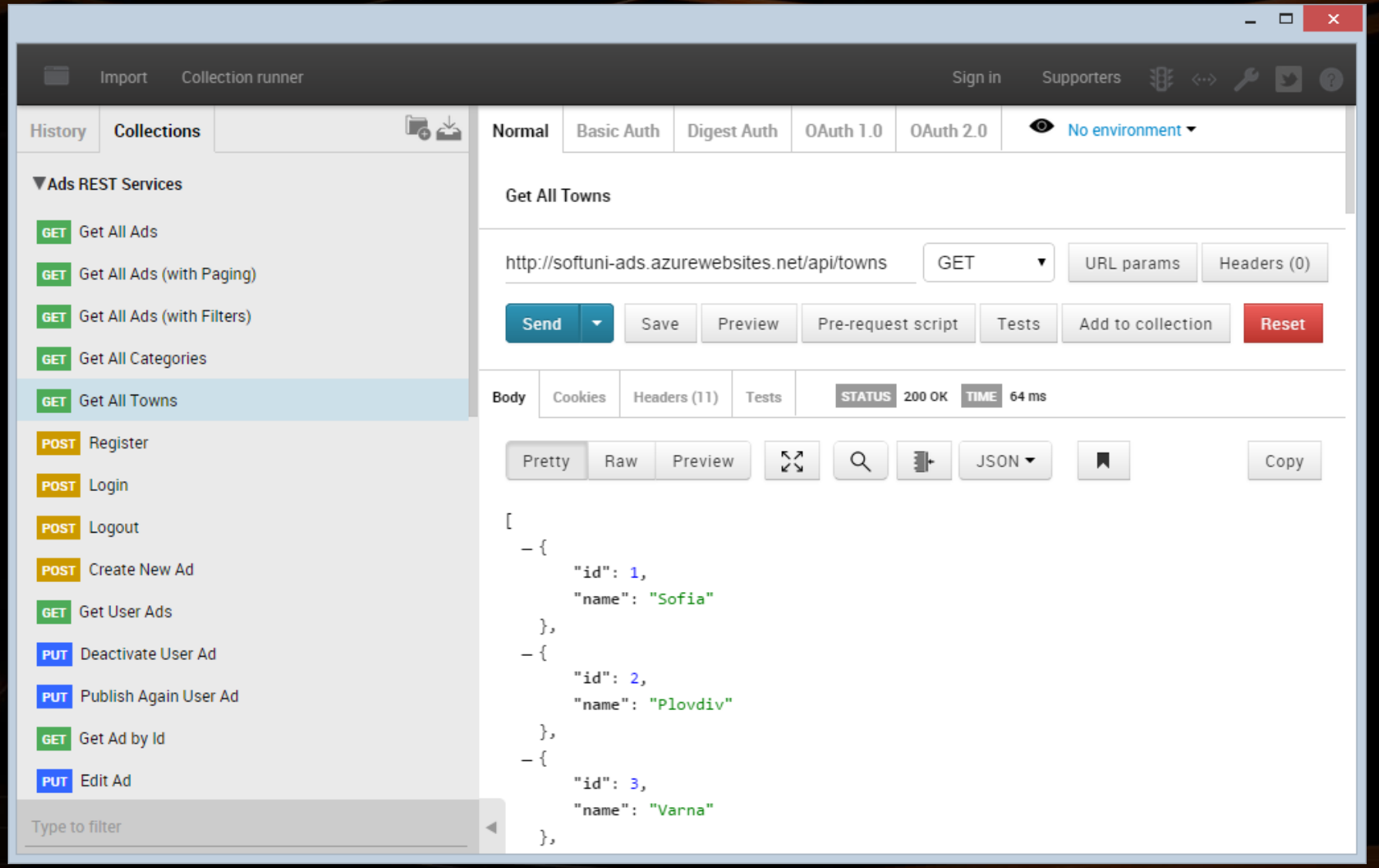
/mongohq

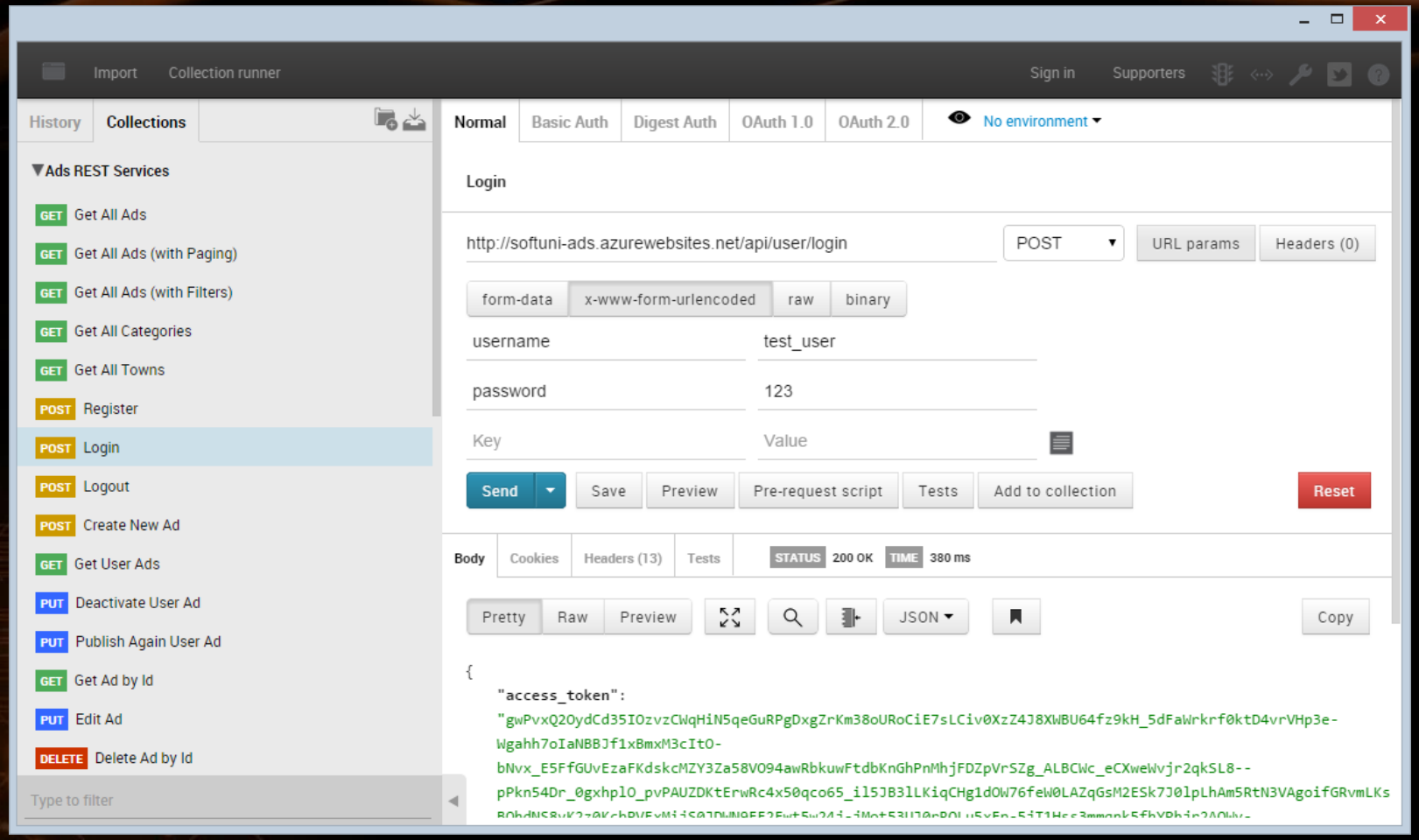
GET	/mongohq	List all tables.
POST	/mongohq	Create one or more tables.
PATCH	/mongohq	Update properties of one or more tables.
DELETE	/mongohq	Delete one or more tables.
GET	/mongohq/{table_name}	Retrieve multiple records.
POST	/mongohq/{table_name}	Create one or more records.
PUT	/mongohq/{table_name}	Update (replace) one or more records.
PATCH	/mongohq/{table_name}	Update (merge) one or more records.
DELETE	/mongohq/{table_name}	Delete one or more records.
GET	/mongohq/{table_name}/{id}	Retrieve one record by identifier.
POST	/mongohq/{table_name}/{id}	Create one record by identifier.
PUT	/mongohq/{table_name}/{id}	Update (replace) one record by identifier.
PATCH	/mongohq/{table_name}/{id}	Update (merge) one record by identifier.
DELETE	/mongohq/{table_name}/{id}	Delete one record by identifier.

RESTful Web Services and HTTP Methods (2)

- Delete (remove) a resource
  - DELETE http://myservice.com/api/Books/3
- Update resource fields (partial update)
  - PATCH http://myservice.com/api/Books/3
- Retrieve resource meta-data
  - HEAD http://myservice.com/api/Books/3
- Inspect resource (typically used in AJAX to request permissions)
  - OPTIONS http://myservice.com/api/Books/3

Postman – REST Client

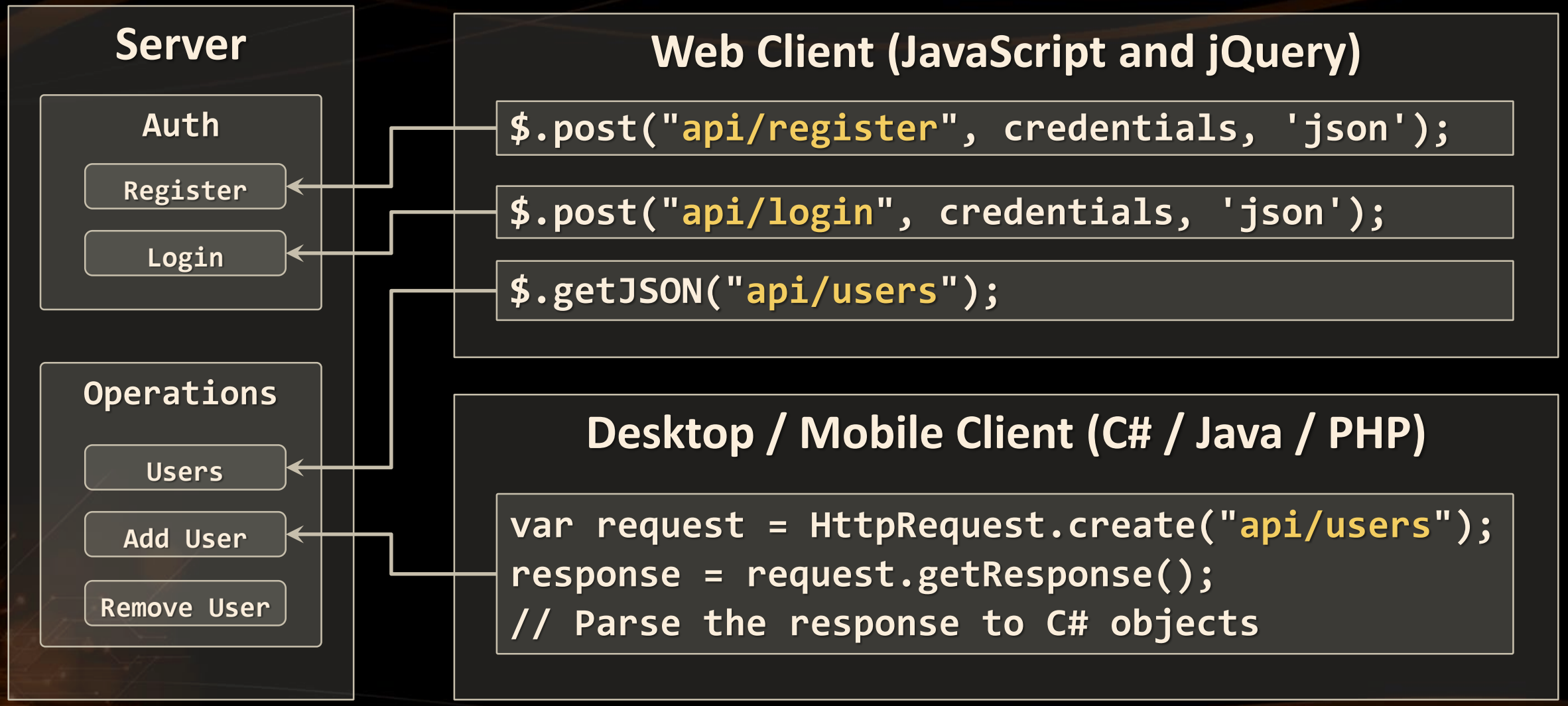


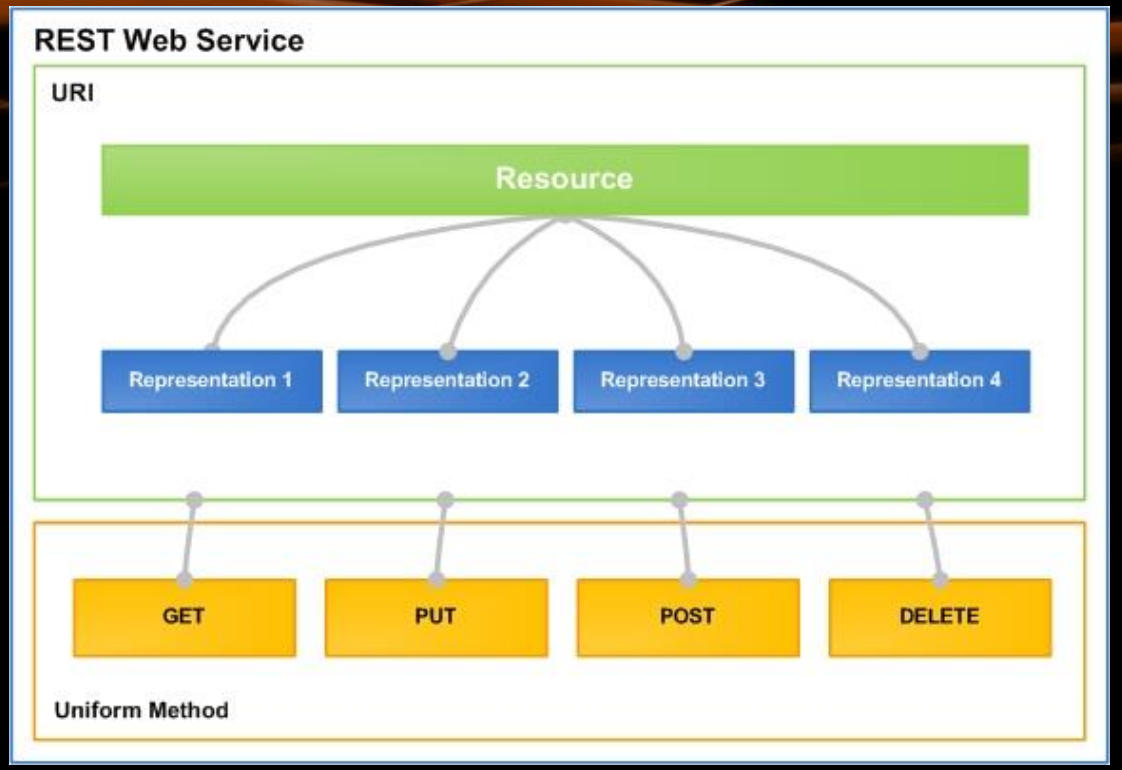


Postman

Live Demo

RESTful API – Example





RESTful Web Services

Live Demo

SOFTWARE UNIVERSITY FOUNDATION

XML, JSON, RSS, Atom

Comparing the Common Service Data Formats

XML

- XML is markup-language for data representation
  - Used for encoding documents in machine-readable form
  - Text-based format, consists of tags, attributes and content
  - Provide data and meta-data in the same time

```
<?xml version="1.0"?>
<library>
  <book><title>HTML 5</title><author>Bay Ivan</author></book>
  <book><title>WPF 4</title><author>Microsoft</author></book>
  <book><title>WCF 4</title><author>Kaka Mara</author></book>
  <book><title>UML 2.0</title><author>Bay Ali</author></book>
</library>
```

JSON

- JSON (JavaScript Object Notation)
  - Standard for representing data structures and associative arrays
  - Lightweight text-based open standard
  - Derived from the JavaScript language

```
{
  "firstName": "John", "lastName": "Smith", "age": 25,
  "address": { "streetAddress": "17 Tintyava Str.",
     "city": "Sofia", "postalCode": "1113" },
  "phoneNumber": [{ "type": "home", "number": "212 555-1234"},
    { "type": "fax", "number": "646 555-4567" }]
},
{ "firstName": "Bay", "lastName": "Ivan", "age": 79 }
```

RSS / Atom

- RSS (**R**eally **S**imple **S**yndication)
  - Family of Web feed formats for accessing site publications
    - E.g. blog entries, news headlines, videos, etc.
  - Based on XML, with standardized XSD schema
- RSS documents (feeds) are list of items
  - Each containing title, author, publish date, summarized text, and metadata
- Atom protocol aimed to enhance RSS and allows publishing

RSS – Example

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
<channel>
  <title>W3Schools Home Page</title>
  <link>http://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>http://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>
</rss>
```

Web Services, SOA and REST

SOFTWARE UNIVERSITY FOUNDATION

Questions?

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Web Services and Cloud" course by Telerik Academy under CC-BY-NC-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg