VSR://EDU/SSE

Software Service Engineering

WS 2025/2026 – 3. Tutorial

Valentin Siegert M.Sc.

Maheshika Walpola M.Sc.

*VSR.Informatik.*TU-Chemnitz.*de*

1 Task 1

1. Answer the following questions:
   - What is the difference between URI and URL?

- URI is an abstract resource identifier (may be a unique name of the resource – URN or its location – URL)
- URL describes a location of the resource and the protocol used to access it

1. Answer the following questions:
   - What is the difference between URI and URL?
   - What is the meaning of the URL scheme?

- URL scheme describes a method to access a resource
- Often (but not always) corresponds to some specific protocol:
  - http
  - ftp
  - news
  - ssh
  - file
  - ldap
  - ...

1. Answer the following questions:
   - What is the difference between URI and URL?
   - What is the meaning of the URL scheme?
   - What, why and how should be encoded in URLS?

- What?
  - Segments of URLs
- Why?
  - Reserved characters : / ? # [ ] @ ! $ & % ' ( ) * + , ; =
  - Non-ASCII characters
  - Unsafe characters (whitespace, “,<,>…)
- How?
  - % + 2 Hexadecimal digits
  - Hexadecimal digits correspond to the ASCII-value of the character
  - Each byte of the UTF-8 encoding for non-ASCII symbols

2. Implement a class for parsing and generating URLs. Use the given template Task1-Template.zip as a start point:

- The constructor Url(string urlStr) should split the urlStr using a **regular expression** and fill in the instance variables Scheme, Host etc.
- The function string ToString() should concat the instance variables to the string representation of the URL.
- The static method string Encode(string s) should convert all characters from s, which are not in VALID_CHARACTERS into the %-form and give the resulting string back
- The static method string Decode(string s) should convert all %-escaped characters from sand give the resulting string back

Task 2

1. Explain the semantics of the following HTTP methods: GET, HEAD, PUT, DELETE, and POST. Which of them are **safe**, which are **idempotent** and which are **cacheable**?

Method Semantics

- GET – retrieve a resource
- HEAD – retrieve only resource metadata
- PUT – replace the resource with given representation
- DELETE – delete a resource
- POST – other actions

Method Characteristics

- A method is **safe** if it produces no side effects (no data is changed on the server-side)
- A method is **idempotent** if its multiple application yields the same side effects as if it was applied once (e.g. removal of a resource)
- A method is **cacheable** if the returned resources can be cached

	safe	idempotent	cacheable
GET	♥	♥	(♥)
HEAD	♥	♥	(♥)
PUT	⊘	♥	⊘
DELETE	⊘	♥	⊘
POST	⊘	⊘	⊘

1. Explain the semantics of the following HTTP methods: GET, HEAD, PUT, DELETE, and POST. Which of them are **safe**, which are **idempotent** and which are **cacheable**?
2. Explain the purpose of the following HTTP headers:
   a) Host
   b) Content-Type
   c) Content-Length
   d) Accept
   e) User-Agent
   f) Location

HTTP Headers

- **Host** - specifies virtual host and port number
- **Content-Type** – media-type of the resource representation
- **Content-Length** – length of the message body in bytes
- **Accept** – media-types supported by a client
- **User-Agent** – information about user's browser (agent in general)
- **Location** – URL of a redirect
- **Content-Location** – URL to use to access the resource

Task 3

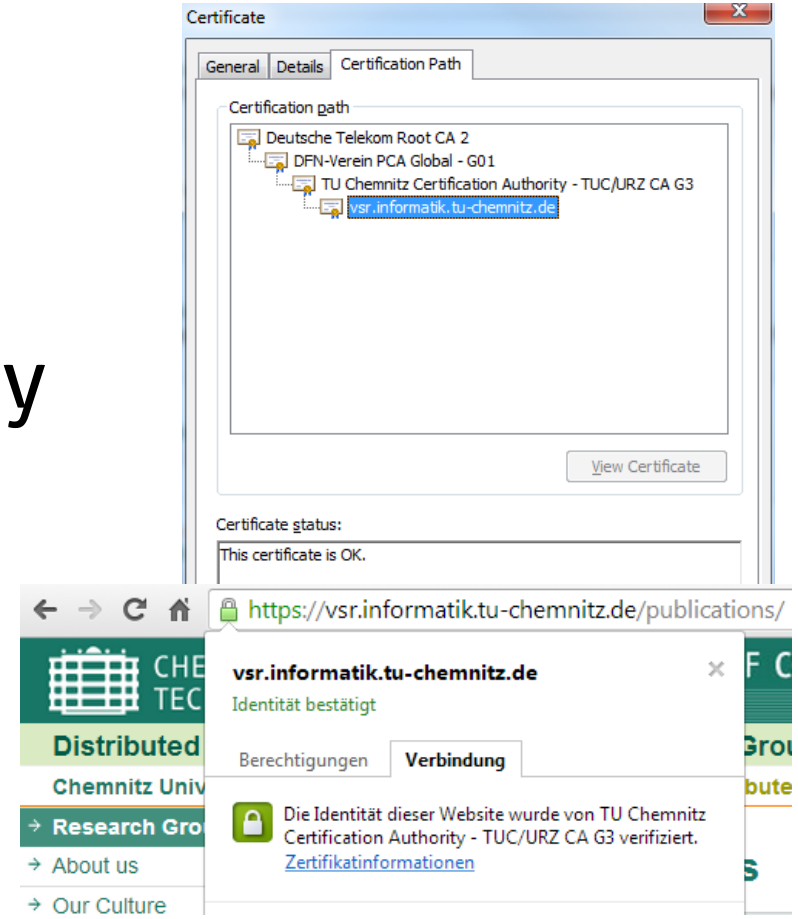
Implement an HTTP message parser and builder based on the template Task3-Template.zip (take care of differentiation between request and response messages).
Complete the methods HTTPMessage(string) and ToString.

Task 4

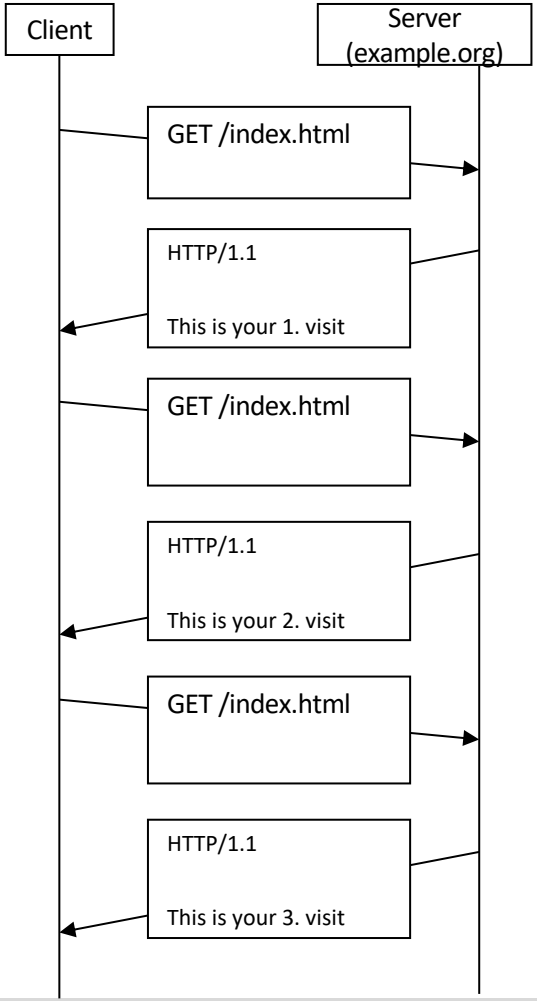
1. What are the goals of HTTPS and how they are achieved?
2. What is the difference between HTTP and HTTPS request/response messages?

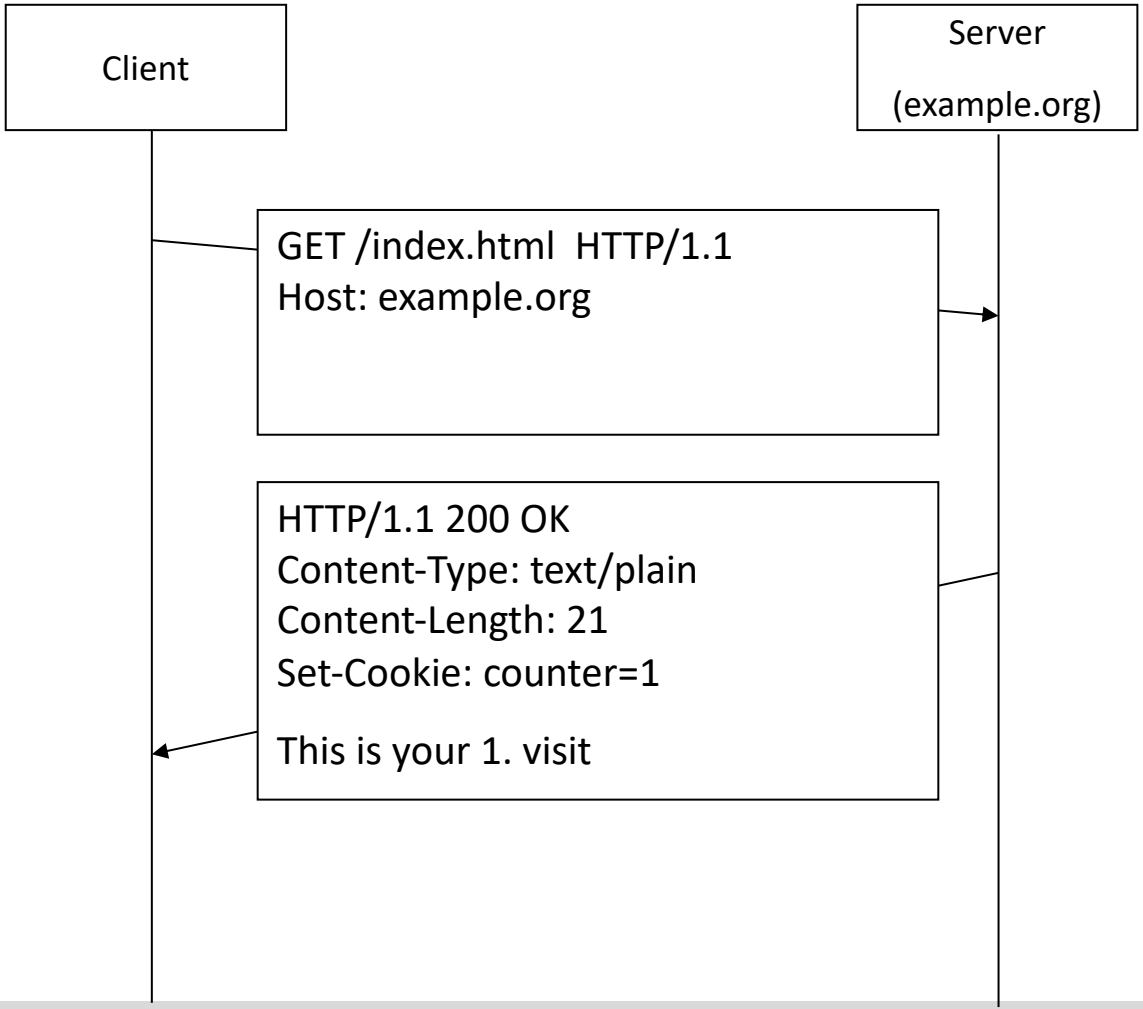
- In OSI-model in layer 6
- In TCP/IP-model
  - Above the Transport layer (i.e. TCP,...)
  - Below the Application layer (i.e. HTTP,...)
- Basic idea: generic security layer
- Protocol consists of 2 layers:

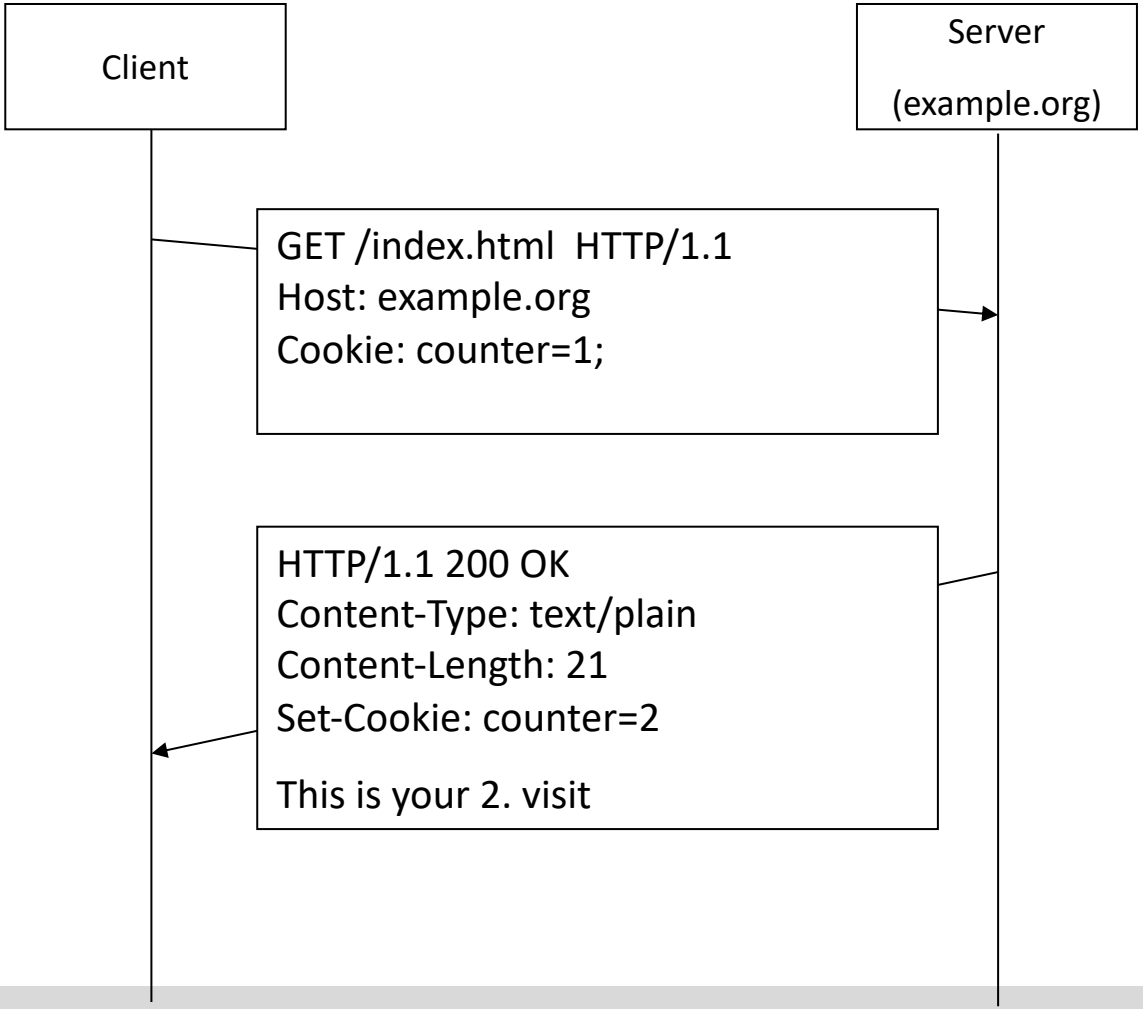
- Authentication using X509 certificates
- Authenticity of certificates is checked based on the Public Key Infrastructure (PKI)
- Encryption using asymmetric and symmetric algorithms
- Integrity using encrypted checksums

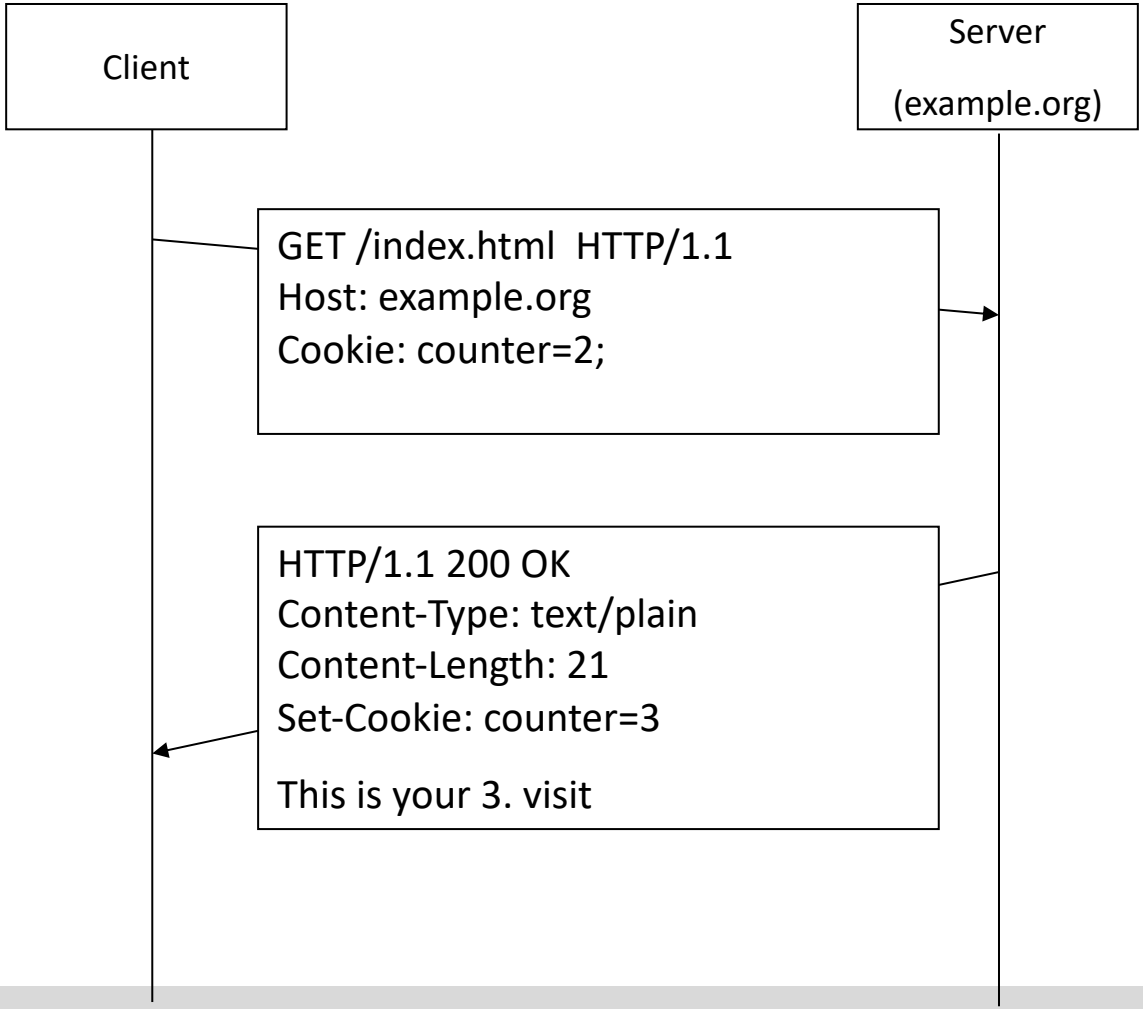


3. What is the purpose of HTTP cookies?
4. Consider the following scenario. A client requests the same resource on the server for several times. Each time he requests a resource, the number of prior requests is returned in the resource representation. Complete the following diagram with missing HTTP headers and status codes required for implementation of the above scenario


Client
Server (example.org)
GET /index.html
HTTP/1.1
This is your 1. visit
GET /index.html
HTTP/1.1
This is your 2. visit
GET /index.html
HTTP/1.1
This is your 3. visit


Client
Server
(example.org)
GET /index.html HTTP/1.1
Host: example.org
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 21
Set-Cookie: counter=1
This is your 1. visit


Client
Server
(example.org)
GET /index.html HTTP/1.1
Host: example.org
Cookie: counter=1;
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 21
Set-Cookie: counter=2
This is your 2. visit


Client
Server
(example.org)
GET /index.html HTTP/1.1
Host: example.org
Cookie: counter=2;
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 21
Set-Cookie: counter=3
This is your 3. visit

3. What is the purpose of HTTP cookies?
4. Consider the following scenario. A client requests the same resource on the server for several times. Each time he requests a resource, the number of prior requests is returned in the resource representation. Complete the following diagram with missing HTTP headers and status codes required for implementation of the above scenario
5. Extend the template *Task1-Template.zip* towards the above scenario.

Your feedback on today's session:

mytuc.org/ttbw

Questions?

vsr-sse@informatik.tu-chemnitz.de

VSR.*Informatik*.TU-Chemnitz.*de*