5. Tutorial

Task 1

1. Explain the semantics of the following HTTP methods: HEAD, GET, 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
3. Implement a class for sending HTTP/1.1 requests to a given URL. Use the template below. Create a HTTP request message using **string concatenation** only.

Tutorial5-Task1-Template.zip
Shared on Dropbox

Tutorial5-Task1-Solution.zip
Shared on Dropbox

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	cachable

HEAD	yes	yes	(yes)
GET	yes	yes	(yes)
PUT	no	yes	no
DELETE	no	yes	no
POST	no	no	no

HTTP Header

- **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 –** information about new location of a resource

Task 2

Implement an HTTP message *parser* and *builder* based on the template below (take care of differentiation between request and response messages). Complete the methods `Parse` and `ToString`.

Tutorial5-Task2-Template.zip
Shared on Dropbox

Tutorial5-Task2-Solution.zip
Shared on Dropbox

Task 3

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

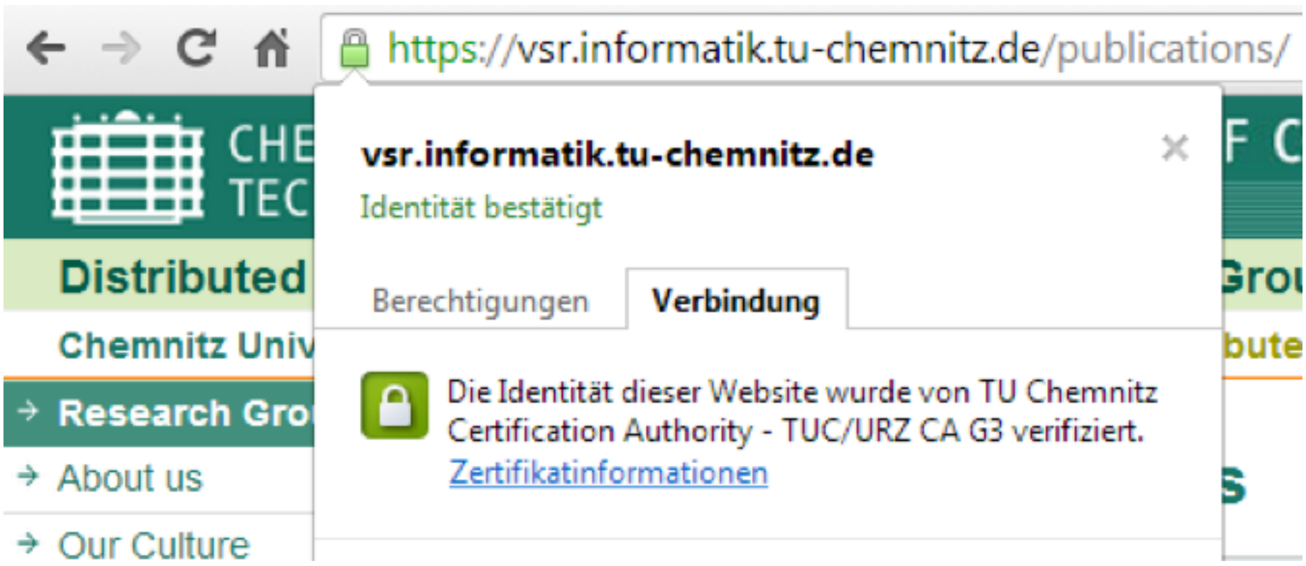
SSL/TLS – Architecture

- In OSI-model in layer 6 (presentation)
- In TCP/IP-model
    - Above the Transport layer (i.e. TCP,...)
    - Below the Application layer (i.e. HTTP,...)

- Basic idea: generic security layer

SSL/TLS

- 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

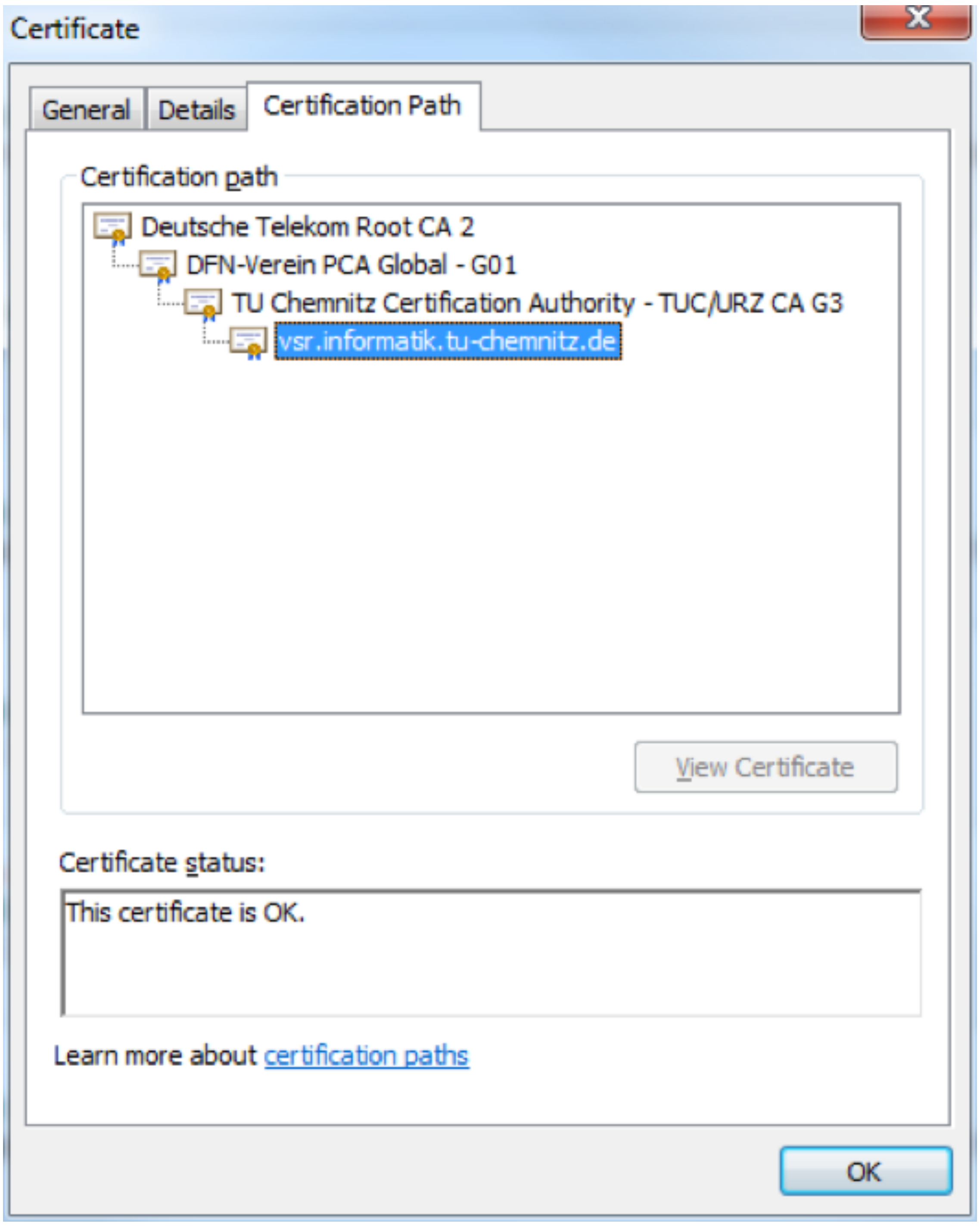

https://vsr.informatik.tu-chemnitz.de/publications/

vsr.informatik.tu-chemnitz.de

Identität bestätigt

Die Identität dieser Website wurde von TU Chemnitz Certification Authority - TUC/URZ CA G3 verifiziert.

Zertifikatinformationen



Certification path

Deutsche Telekom Root CA 2
DFN-Verein PCA Global - G01
TU Chemnitz Certification Authority - TUC/URZ CA G3
vsr.informatik.tu-chemnitz.de

Certificate status:

This certificate is OK.

Learn more about certification paths

Assignment 1

Inform yourself about the "chunked" transfer encoding and its purpose. Extend the HTTP message parser and builder from Task 2 with the support for “chunked” transfer encoding.

- Message body is transferred as a series of chunked, each with its own size indicator
- Goals:
  - Delivery of dynamically produced content
  - Keep connection open

```
HTTP/1.1 200 OK
Content-Type: text/html
Transfer-Encoding: chunked
5
Hello
6
_world
0
```

Tutorial5-Assignment1-Solution.zip
Shared on Dropbox

Assignment 2

Based on the template below implement a server, which is able to deliver requested resources from the `HttpServer.DOCUMENT_ROOT` folder (for POST requests return only `201 Created`). Test your implementation in the browser.

Tutorial5-Assignment2-Template.zip
Shared on Dropbox

Tutorial5-Assignment2-Solution.zip
Shared on Dropbox

Assignment 3

Modify[1] the HTTP request implementation of Task 1 to request the following resource: https://www.tu-chemnitz.de (HTTPS)

Tutorial5-Assignment3-Solution.zip
Shared on Dropbox

---

[1] Use http://msdn.microsoft.com/en-us/library/system.net.security.sslstream.aspx as a reference