Internet Essentials

Internet Applications, ID1354

Contents

Section

The IP Protocol

- All Internet communication is based on the Internet Protocol (IP).

The IP Protocol

- All Internet communication is based on the Internet Protocol (IP).
- IP provides basic functionality for sending and receiving data.

The IP Protocol

- All Internet communication is based on the Internet Protocol (IP).
- IP provides basic functionality for sending and receiving data.
- Data is sent in chunks, called packages. A package is like an envelope for a letter. It has sender and a receiver addresses and a content, which is the data being transmitted.

The IP Protocol

- All Internet communication is based on the Internet Protocol (IP).
- IP provides basic functionality for sending and receiving data.
- Data is sent in chunks, called packages. A package is like an envelope for a letter. It has sender and a receiver addresses and a content, which is the data being transmitted.
- A node (computer) receiving a packet can accept it, ignore it or retransmit it.

The IP Protocol

- All Internet communication is based on the Internet Protocol (IP).
- IP provides basic functionality for sending and receiving data.
- Data is sent in chunks, called packages. A package is like an envelope for a letter. It has sender and a receiver addresses and a content, which is the data being transmitted.
- A node (computer) receiving a packet can accept it, ignore it or retransmit it.
- A node dedicated to retransmitting packets across subnet borders is called a router.

IP Address

- An internet (version 4) address has 32 bits divided into four bytes, [0-255].[0-255].[0-255].[0-255]. Each node connected to the internet has one or more addresses.

IP Address

- An internet (version 4) address has 32 bits divided into four bytes, [0-255].[0-255].[0-255].[0-255]. Each node connected to the internet has one or more addresses.
- Normally, an IP address must be unique, assigned only to one node.

IP Address

- An internet (version 4) address has 32 bits divided into four bytes, [0-255].[0-255].[0-255].[0-255]. Each node connected to the internet has one or more addresses.
- Normally, an IP address must be unique, assigned only to one node.
- Some addresses, like 192.168.X.X are dedicated to private networks and can be used freely. Such an address is not transmitted on the public internet. Instead, it is translated to a public address by a router.

The TCP Protocol

- TCP, Transmission Control Protocol, is used on top of the IP protocol.

The TCP Protocol

- TCP, Transmission Control Protocol, is used on top of the IP protocol.
- TCP adds transport guarantees, for example the following.

The TCP Protocol

- TCP, Transmission Control Protocol, is used on top of the IP protocol.
- TCP adds transport guarantees, for example the following.
  - Packets are delivered to the receiver in the same order they are sent by the sender.

The TCP Protocol

- TCP, Transmission Control Protocol, is used on top of the IP protocol.
- TCP adds transport guarantees, for example the following.
  - Packets are delivered to the receiver in the same order they are sent by the sender.
  - Delivered packets have the same content as sent packets.

The TCP Protocol

- TCP, Transmission Control Protocol, is used on top of the IP protocol.
- TCP adds transport guarantees, for example the following.
  - Packets are delivered to the receiver in the same order they are sent by the sender.
  - Delivered packets have the same content as sent packets.
  - There are no lost packets.

The TCP Protocol (Cont'd)

- TCP is connection-oriented, think of a telephone line as opposed to sending a letter. To establish a TCP connection is a slow operation.

The TCP Protocol (Cont'd)

- TCP is connection-oriented, think of a telephone line as opposed to sending a letter. To establish a TCP connection is a slow operation.
- TCP handles ports, which makes it possible to have multiple connections with the same IP address open simultaneously. A port is identified by a number. An endpoint of a TCP connection has an IP address and a port number.

DNS

- IP addresses are normally translated to names (instead of numbers). Such a name is called domain name.

DNS

- IP addresses are normally translated to names (instead of numbers). Such a name is called domain name.
- Domain names are divided into subdomains, divided by dots (`.`)
  - The address **`www.ict.kth.se`** consists of the subdomain **`www`**, which is part of the subdomain **`ict`**, which is part of **`kth`**, which is part of **`se`**, which is part of the root, **`.`**

DNS

- IP addresses are normally translated to names (instead of numbers). Such a name is called domain name.
- Domain names are divided into subdomains, divided by dots (`.`)
    - The address **`www.ict.kth.se`** consists of the subdomain **`www`**, which is part of the subdomain **`ict`**, which is part of **`kth`**, which is part of **`se`**, which is part of the root, `.`
- The translation between numbers and names is managed by DNS, Domain Name System.

URL

- A Uniform Resource Locator, URL defines a resource's location on the internet.

URL

- A Uniform Resource Locator, URL defines a resource's location on the internet.
- A URL consists of four parts.
  1. A protocol, e.g., **`http`**

URL

- A Uniform Resource Locator, URL defines a resource's location on the internet.
- A URL consists of four parts.
  1. A protocol, e.g., **`http`**
  2. A host (IP address or name), **`http://www.kth.se`**

URL

- A Uniform Resource Locator, URL defines a resource's location on the internet.
- A URL consists of four parts.
  1. A protocol, e.g., **`http`**
  2. A host (IP address or name),
     `http://www.kth.se`
  3. A port number (optional). The default HTTP port number is 80.
     `http://www.kth.se:8080`

URL

- A Uniform Resource Locator, URL defines a resource's location on the internet.
- A URL consists of four parts.
  1. A protocol, e.g., `http`
  2. A host (IP address or name), `http://www.kth.se`
  3. A port number (optional). The default HTTP port number is 80. `http://www.kth.se:8080`
  4. A path, which identifies the resource's location on the server. `http://www.kth.se:8080/abc/index.html`

URN and URI

- A Uniform Resource Name, URN is a resource identifier without host name and port number. A typical example is a isbn identifying a book.

URN and URI

- A Uniform Resource Name, URN is a resource identifier without host name and port number. A typical example is a isbn identifying a book.
- A Uniform Resource Identifier, URI is either a URL or URN.

Section

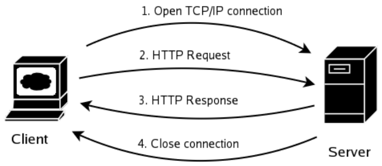
HTTP

- HyperText Transfer Protocol, HTTP is used for communication between web browsers and web servers.

HTTP

- HyperText Transfer Protocol, HTTP is used for communication between web browsers and web servers.
- HTTP is based on TCP, which means a TCP connection is established for each browser-server communication.

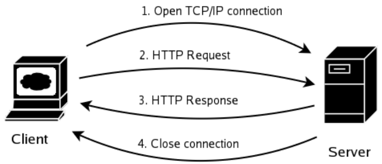
The Request-Response Cycle



A HTTP communication typically proceeds as follows.

1. The client opens a TCP connection to the server.

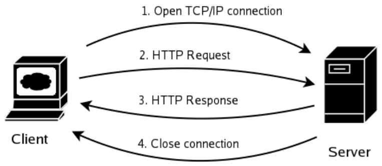
The Request-Response Cycle



A HTTP communication typically proceeds as follows.

1. The client opens a TCP connection to the server.
2. The client sends a request for a resource on the server. The request consists of a HTTP header, and data if the user submitted data to the server.

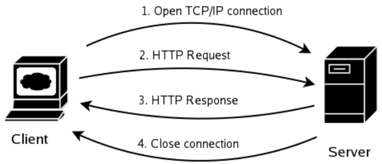
The Request-Response Cycle



A HTTP communication typically proceeds as follows.

1. The client opens a TCP connection to the server.
2. The client sends a request for a resource on the server. The request consists of a HTTP header, and data if the user submitted data to the server.
3. The server sends a response to the client. Also the response consists of HTTP headers, and data if the response required data.

The Request-Response Cycle



A HTTP communication typically proceeds as follows.

1. The client opens a TCP connection to the server.
2. The client sends a request for a resource on the server. The request consists of a HTTP header, and data if the user submitted data to the server.
3. The server sends a response to the client. Also the response consists of HTTP headers, and data if the response required data.
4. The server closes the TCP connection.

The Request-Response Cycle (Cont'd)

- HTTP is stateless. Neither server nor browser remembers anything about previous request-response cycles. Session handling must be added in server-side code.

The Request-Response Cycle (Cont'd)

- HTTP is stateless. Neither server nor browser remembers anything about previous request-response cycles. Session handling must be added in server-side code.
- To establish a TCP connection is expensive. Therefore, TCP connections might be kept alive and reused for multiple request-response cycles. This is specified with the **`keep-alive`** HTTP header, se below.

Cookies

- A cookie is a piece of data that is stored on the client.

Cookies

- A cookie is a piece of data that is stored on the client.
- The cookie is tagged with the server's domain name and included in every request to that server.

Cookies

- A cookie is a piece of data that is stored on the client.
- The cookie is tagged with the server's domain name and included in every request to that server.
- This enables the server to associate data with a specific client.

Cookies

- A cookie is a piece of data that is stored on the client.
- The cookie is tagged with the server’s domain name and included in every request to that server.
- This enables the server to associate data with a specific client.
- Cookies can be used to store the user’s settings, for example display language.

HTTP Sessions

- As mentioned above, HTTP is stateless.

HTTP Sessions

- As mentioned above, HTTP is stateless.
- Still, the server must be able to recognize which calls originate from the same client. Otherwise for example log in is impossible.

HTTP Sessions

- As mentioned above, HTTP is stateless.
- Still, the server must be able to recognize which calls originate from the same client. Otherwise for example log in is impossible.
- One commonly used method to solve this problem is to use cookies.

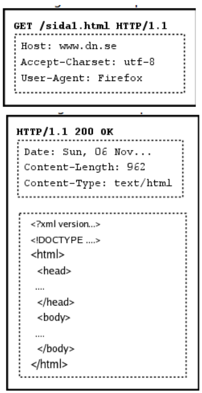
HTTP Sessions

- As mentioned above, HTTP is stateless.
- Still, the server must be able to recognize which calls originate from the same client. Otherwise for example log in is impossible.
- One commonly used method to solve this problem is to use cookies.
- If a request has a cookie with a session identifier, it identifies the user. If there is no such cookie, the user does not have a running session.

HTTP Sessions

- As mentioned above, HTTP is stateless.
- Still, the server must be able to recognize which calls originate from the same client. Otherwise for example log in is impossible.
- One commonly used method to solve this problem is to use cookies.
- If a request has a cookie with a session identifier, it identifies the user. If there is no such cookie, the user does not have a running session.
- On the server, the session id can be associated with any amount of data related to the user with that session. This is called conversational state.

HTTP Message Format



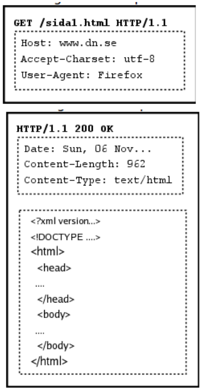
```
GET /sida1.html HTTP/1.1
Host: www.dn.se
Accept-Charset: utf-8
User-Agent: Firefox
```

```
HTTP/1.1 200 OK
Date: Sun, 06 Nov...
Content-Length: 962
Content-Type: text/html

<?xml version...>
<!DOCTYPE ....>
<html>
  <head>
 ....
  </head>
  <body>
 ....
  </body>
</html>
```

- A HTTP message has start-line, headers and body.

HTTP Message Format



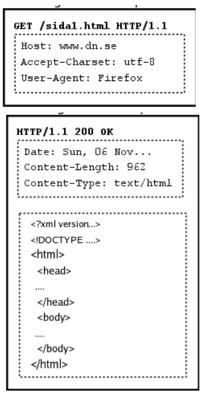
```
GET /sida1.html HTTP/1.1
Host: www.dn.se
Accept-Charset: utf-8
User-Agent: Firefox
```

```
HTTP/1.1 200 OK
Date: Sun, 06 Nov...
Content-Length: 962
Content-Type: text/html

<?xml version...>
<!DOCTYPE ....>
<html>
 <head>
 ....
 </head>
 <body>
 ....
 </body>
</html>
```

- A HTTP message has start-line, headers and body.
- The request start-line consists of HTTP method (se left), URL path and HTTP version, e.g., **`GET /page1.html HTTP/1.1`**

HTTP Message Format



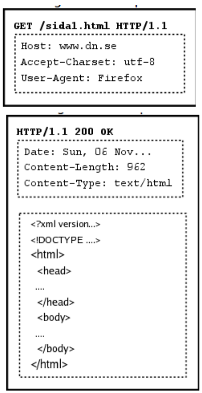
```
GET /sida1.html HTTP/1.1
Host: www.dn.se
Accept-Charset: utf-8
User-Agent: Firefox
```

```
HTTP/1.1 200 OK
Date: Sun, 06 Nov...
Content-Length: 962
Content-Type: text/html

<?xml version...>
<!DOCTYPE ....>
<html>
  <head>
  ....
  </head>
  <body>
  ....
  </body>
</html>
```

- A HTTP message has start-line, headers and body.
- The request start-line consists of HTTP method (se left), URL path and HTTP version, e.g., **`GET /page1.html HTTP/1.1`**
- The response start-line consists of HTTP version, status code and reason, e.g., **`HTTP/1.1 200 OK`**

HTTP Message Format



```
GET /sida1.html HTTP/1.1
Host: www.dn.se
Accept-Charset: utf-8
User-Agent: Firefox
```

```
HTTP/1.1 200 OK
Date: Sun, 06 Nov...
Content-Length: 962
Content-Type: text/html

<?xml version...>
<!DOCTYPE ....>
<html>
  <head>
  ....
  </head>
  <body>
  ....
  </body>
</html>
```

- A HTTP message has start-line, headers and body.
- The request start-line consists of HTTP method (se left), URL path and HTTP version, e.g., **`GET /page1.html HTTP/1.1`**
- The response start-line consists of HTTP version, status code and reason, e.g., **`HTTP/1.1 200 OK`**
- Sample request (top) and response (bottom) messages are depicted to the left.

Status Codes

- A HTTP response contains a status code to indicate the outcome of the request. There are five different categories of status codes.

1xx	Reply contains information, for example **`101`**, Switch Protocol.

Status Codes

- A HTTP response contains a status code to indicate the outcome of the request. There are five different categories of status codes.

1xx	Reply contains information, for example **`101`**, Switch Protocol.
2xx	Success, for example **`200`**, OK.

Status Codes

- A HTTP response contains a status code to indicate the outcome of the request. There are five different categories of status codes.

1xx	Reply contains information, for example **`101`**, Switch Protocol.
2xx	Success, for example **`200`**, OK.
3xx	Redirection, for example **`301`**, Moved Permanently.

Status Codes

- A HTTP response contains a status code to indicate the outcome of the request. There are five different categories of status codes.

1xx	Reply contains information, for example **`101`**, Switch Protocol.
2xx	Success, for example **`200`**, OK.
3xx	Redirection, for example **`301`**, Moved Permanently.
4xx	Client error, for example **`404`**, Not Found.

Status Codes

- A HTTP response contains a status code to indicate the outcome of the request. There are five different categories of status codes.

1xx	Reply contains information, for example **`101`**, Switch Protocol.
2xx	Success, for example **`200`**, OK.
3xx	Redirection, for example **`301`**, Moved Permanently.
4xx	Client error, for example **`404`**, Not Found.
5xx	Server error, for example **`500`**, Internal Server Error

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.
DELETE	Delete the resource at the given URL.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.
DELETE	Delete the resource at the given URL.
HEAD	Like GET, but only deliver headers.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.
DELETE	Delete the resource at the given URL.
HEAD	Like GET, but only deliver headers.
TRACE	Return the request message.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.
DELETE	Delete the resource at the given URL.
HEAD	Like GET, but only deliver headers.
TRACE	Return the request message.
OPTIONS	Tell which HTTP methods can be used with the specified URL.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.
DELETE	Delete the resource at the given URL.
HEAD	Like GET, but only deliver headers.
TRACE	Return the request message.
OPTIONS	Tell which HTTP methods can be used with the specified URL.
CONNECT	Connect to another host.

HTTP Methods

- HTTP 1.1 has eight different methods that requires the following server actions.

GET	Deliver resource identified by the specified URL.
POST	Accept message body and deliver it to the resource at the specified URL.
PUT	Accept message body and store it as a resource with the specified URL.
DELETE	Delete the resource at the given URL.
HEAD	Like GET, but only deliver headers.
TRACE	Return the request message.
OPTIONS	Tell which HTTP methods can be used with the specified URL.
CONNECT	Connect to another host.

- **`GET`** and **`POST`** are the most common methods and the only ones we will use in this course.

Safe and Idempotent Methods

- **GET** and **HEAD** are safe methods, which means they should not take any action other than to retrieve the specified resource.

Safe and Idempotent Methods

- **`GET`** and **`HEAD`** are safe methods, which means they should not take any action other than to retrieve the specified resource.
- **`GET`**, **`HEAD`**, **`PUT`**, **`DELETE`**, **`OPTIONS`** and **`TRACE`** are idempotent methods, which means the same request can be sent multiple times without any side-effects on the server.

Safe and Idempotent Methods

- **`GET`** and **`HEAD`** are safe methods, which means they should not take any action other than to retrieve the specified resource.
- **`GET`**, **`HEAD`**, **`PUT`**, **`DELETE`**, **`OPTIONS`** and **`TRACE`** are idempotent methods, which means the same request can be sent multiple times without any side-effects on the server.
- **`POST`** is not idempotent. If you submit the same purchase order multiple times in a web shop you will probably by multiple items. The purchase is typically a **`POST`** request.

When to Use GET

- Use **`GET`** when
  - The only desired action is to retrieve the specified resource.

When to Use GET

- Use **`GET`** when
  - The only desired action is to retrieve the specified resource.
  - If it shall be possible to bookmark the link.

When to Use GET

- Use **GET** when
  - The only desired action is to retrieve the specified resource.
  - If it shall be possible to bookmark the link.
  - The URL is shorter than 255 bytes. Note that a **GET** URL is longer than a **POST** URL since data is included in the URL which **GET**, but is in the message body with **POST** (see below).

When to Use GET

- Use **GET** when
  - The only desired action is to retrieve the specified resource.
  - If it shall be possible to bookmark the link.
  - The URL is shorter than 255 bytes. Note that a **GET** URL is longer than a **POST** URL since data is included in the URL which **GET**, but is in the message body with **POST** (see below).
  - You want to be able to write the entire request, including data, in the browser. This is useful when debugging.

When to Use POST

- Use **`POST`** when
  - The required action updates server state, for example saves something in a database.

When to Use POST

- Use **`POST`** when
  - The required action updates server state, for example saves something in a database.
  - The data does not fit within the 255 byte limit for URLs.

When to Use POST

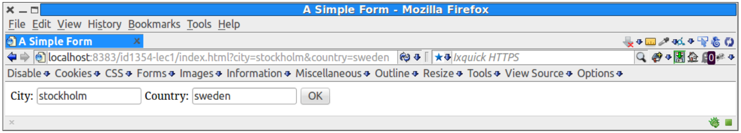
- Use **`POST`** when
  - The required action updates server state, for example saves something in a database.
  - The data does not fit within the 255 byte limit for URLs.
  - The data shall not appear in the URL. Note that this is not a matter of security, data is sent in clear text also when using **`POST`**.

HTTP Parameters

- HTTP parameters are data included in a request to a web server.

HTTP Parameters

- HTTP parameters are data included in a request to a web server.
- A typical example is when the user has entered data in a HTML form.


A Simple Form - Mozilla Firefox

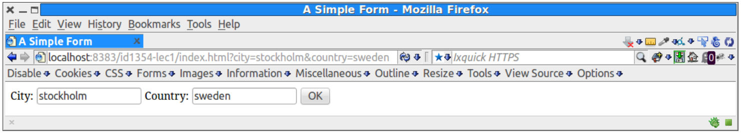
localhost:8383/id1354-lec1/index.html?city=stockholm&country=sweden

City: stockholm Country: sweden OK

File Edit View History Bookmarks Tools Help
A Simple Form
Ixquick HTTPS
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

HTTP Parameters

- HTTP parameters are data included in a request to a web server.
- A typical example is when the user has entered data in a HTML form.


A Simple Form - Mozilla Firefox

localhost:8383/id1354-lec1/index.html?city=stockholm&country=sweden

City: stockholm Country: sweden OK
File Edit View History Bookmarks Tools Help
A Simple Form
Ixquick HTTPS
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

- When using the **GET** method, parameters are appended to the URL as a query string, **http://some.domain/some/path?city=stockholm&country=sweden**

HTTP Parameters

- HTTP parameters are data included in a request to a web server.
- A typical example is when the user has entered data in a HTML form.

- When using the **`GET`** method, parameters are appended to the URL as a query string, **`http://some.domain/some/path?city=stockholm&country=sweden`**
- When using the **`POST`** method, parameters are included in the message body.

HTTP Headers

- HTTP headers have the syntax
  `name: value`

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.
User-Agent	Identifies the sender browser and operating system.

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.
User-Agent	Identifies the sender browser and operating system.
Content-Length	Message body length in bytes.

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.
User-Agent	Identifies the sender browser and operating system.
Content-Length	Message body length in bytes.
Connection	Keep connection open future requests.

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.
User-Agent	Identifies the sender browser and operating system.
Content-Length	Message body length in bytes.
Connection	Keep connection open future requests.

- Sample response headers are:

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.
User-Agent	Identifies the sender browser and operating system.
Content-Length	Message body length in bytes.
Connection	Keep connection open future requests.

- Sample response headers are:

Content-Length	Message body length in bytes.

HTTP Headers

- HTTP headers have the syntax
  `name: value`
- There are several predefined headers, and it is also allowed to add new headers.
- Sample request headers are:

Host	The receiver address or domain name.
User-Agent	Identifies the sender browser and operating system.
Content-Length	Message body length in bytes.
Connection	Keep connection open future requests.

- Sample response headers are:

Content-Length	Message body length in bytes.
Content-Type	Media Type (see below) of response.

Media Type

- Media Type (or MIME Type) defines message content. This tells the receiver how to interpret the data.

Media Type

- Media Type (or MIME Type) defines message content. This tells the receiver how to interpret the data.
- Some media types are:

text/html	HTML markup
text/plain	Plain text
image/png	A png image
video/ogg	A ogg video.

Section

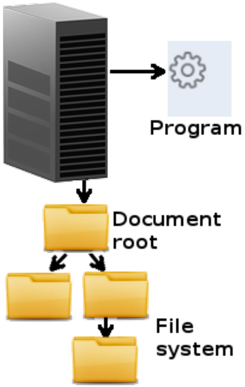
Web Browsers

- It is important to test the web application with all different browsers that shall be able to display it.

Web Browsers

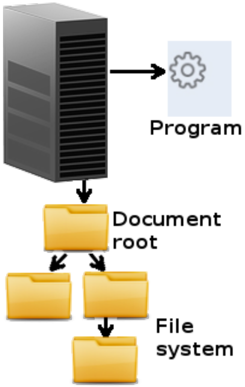
- It is important to test the web application with all different browsers that shall be able to display it.
- Browsers behave differently, and you should expect that some break specifications.

Web Servers



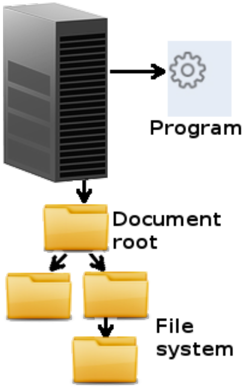
- The web server can deliver static content and also call server-side programs, like PHP, Java or .NET programs.

Web Servers



- The web server can deliver static content and also call server-side programs, like PHP, Java or .NET programs.
- The most commonly used web server is apache, ***https://httpd.apache.org/***

Web Servers



- The web server can deliver static content and also call server-side programs, like PHP, Java or .NET programs.
- The most commonly used web server is apache, *https://httpd.apache.org/*
- Other common web servers are nginx, ***http://wiki.nginx.org/Main*** and Microsoft IIS.

Web Servers (Cont'd)

- You need to install a web server on your laptop. All labs will be reported on your own laptop, there is no web server in ICT school where you can run all the labs.

Web Servers (Cont'd)

- You need to install a web server on your laptop. All labs will be reported on your own laptop, there is no web server in ICT school where you can run all the labs.
- It might take time to get the web server running. You are advised to start installing the web server now.