

Performance testing

Required technical knowledge

User workflow vs Application workflow

● User

● Clicks

● Selects

● Checks

● Types

● Uploads

● Downloads

● Navigates

Click here

A screenshot of a web form. At the top, there is a red button with the text "Click here". Below it is a dropdown menu titled "Screen saver:" with a mouse cursor pointing to the "Aurora" option. The dropdown menu lists several options: 3D Text, Aurora, Blank, Bubbles, Mystify, Photos, Ribbons, Windows Energy, and Windows Logo. Below the dropdown are input fields for "Name", "Email", and "Website", and a "Comment" text area. To the right of the form are three checkboxes, each with a checked mark and a range: 0000 - 0100, 0100 - 0200, and 0200 - 0300.

A screenshot of a mobile-style option menu. The menu is titled "Select an option" and has a gear icon in the top right corner. The menu is open, showing four options: "Liquid", "Fixed", "Responsive", and "Masonry". The "Liquid" option is currently selected.

NOT

● Application

● Sends request

● Receives response

● Sends request

● Receives response

● Sends request

● Receives sub-response

● Sends sub-request

● Receives sub-response

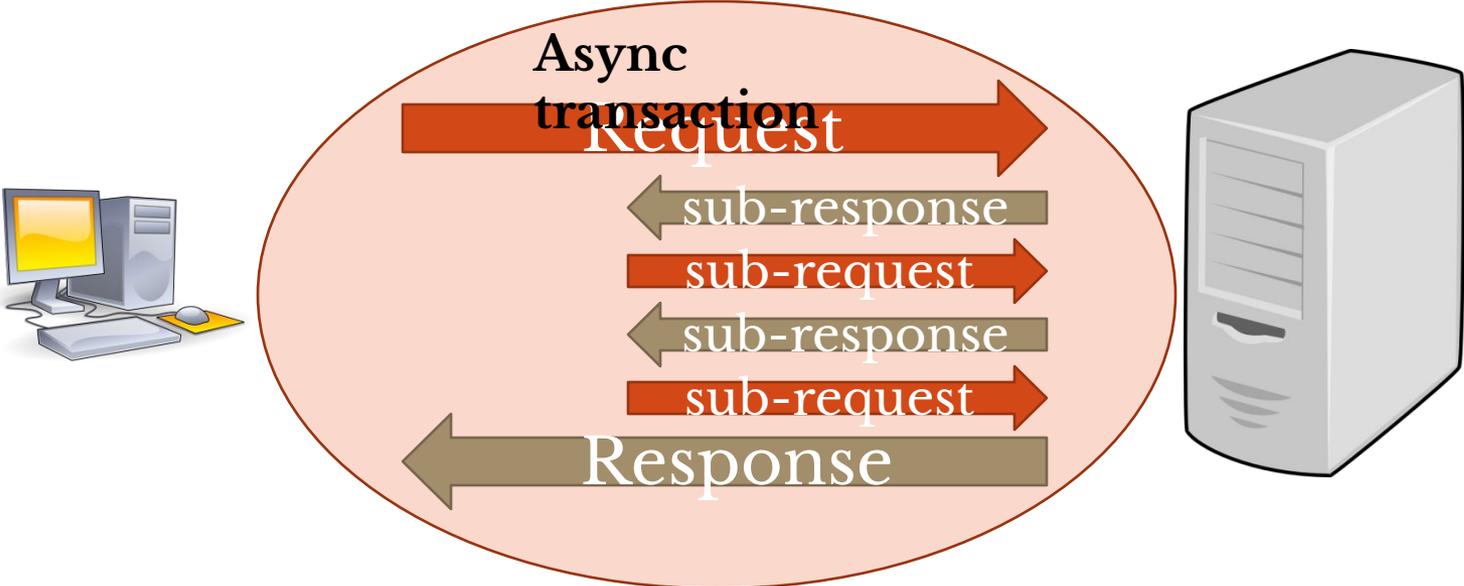
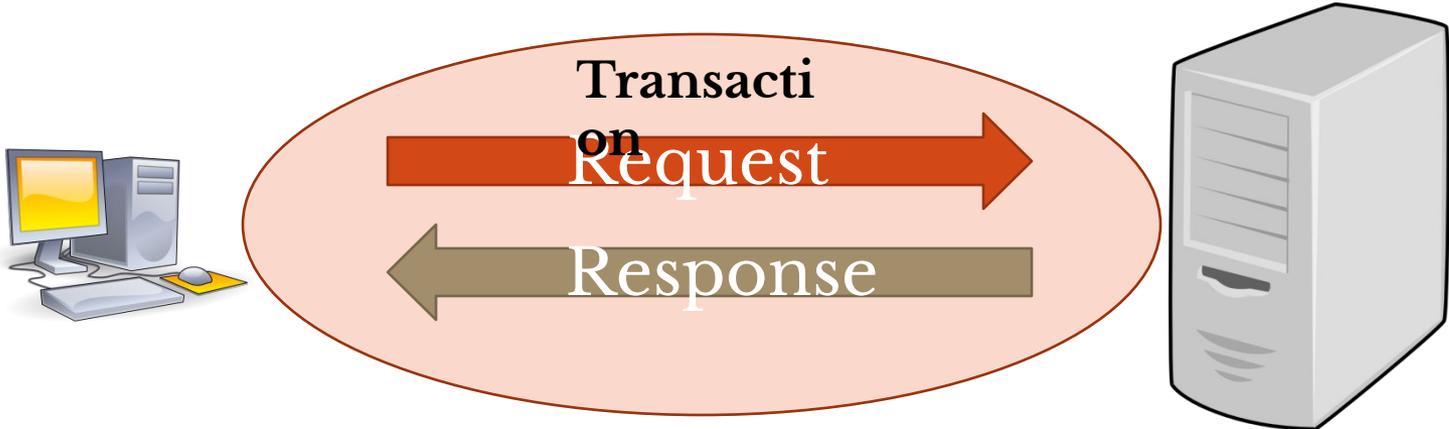
● Sends sub-request

● Receives response

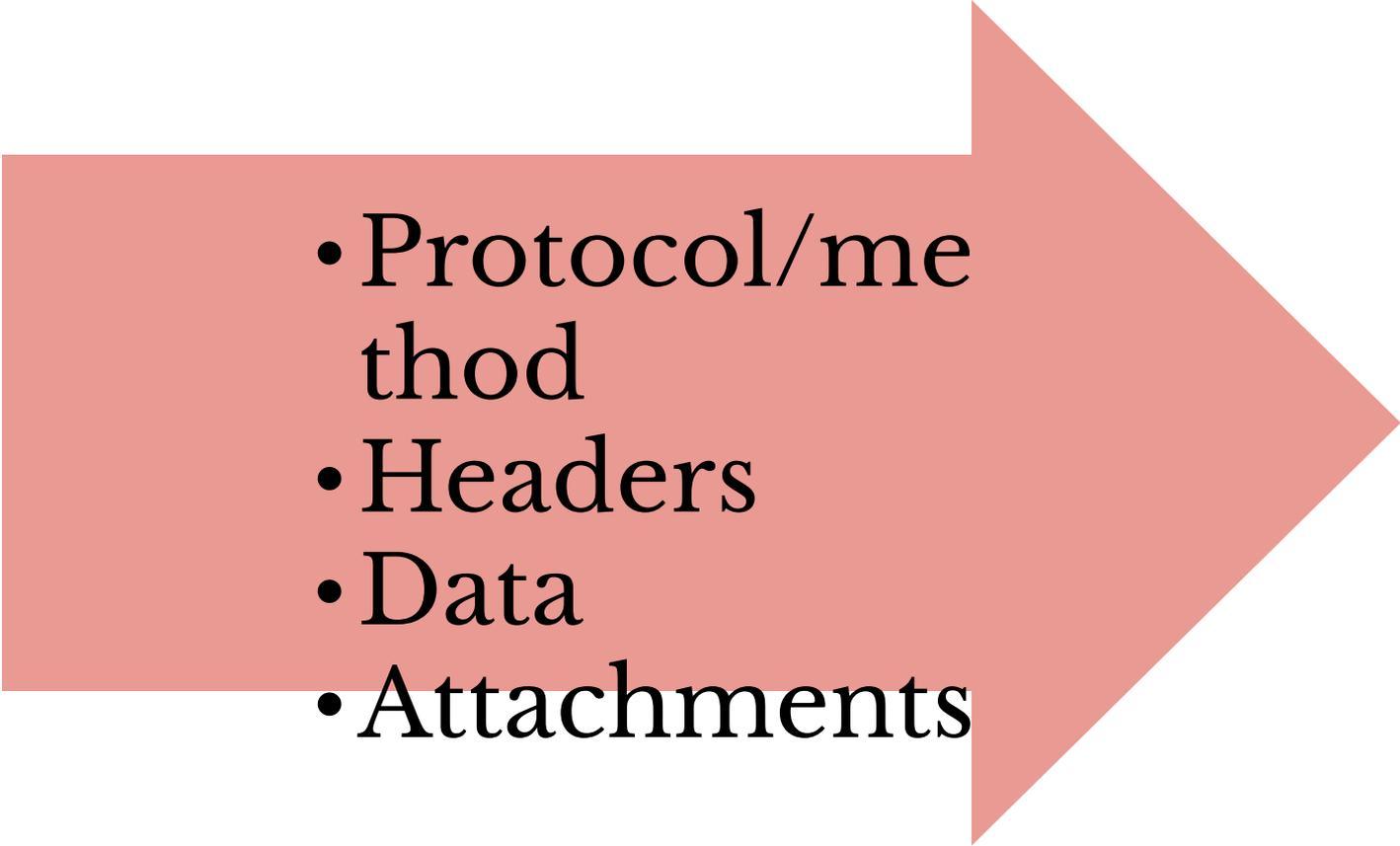
THI

Performance tests

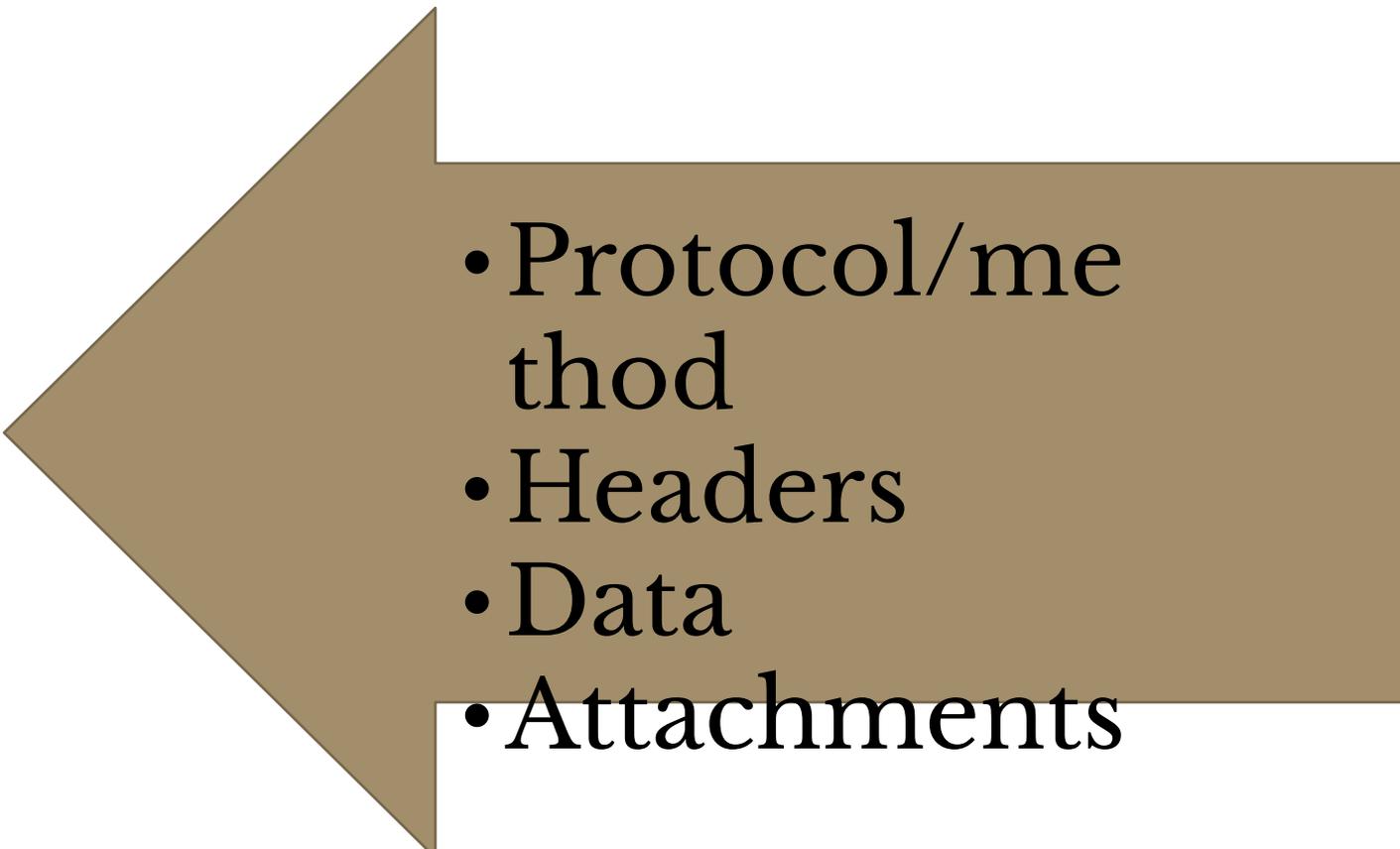
Client-server application architecture



Request structure

- 
- Protocol/method
 - Headers
 - Data
 - Attachments

Response structure

- 
- Protocol/method
 - Headers
 - Data
 - Attachments

HTTP Protocol

- Methods
 - Get — “show me the data that I want”
 - Post — “take the data and process it”
 - Put — “keep the data please”
 - Delete — “delete the data please”
 - +5 more

Each method has its
own role

**Theoretically | Best Practices | Classic
approach**

HTTP Protocol

- Methods

- Get – “show me the data that I want”
 - Post – “take the data and process it”
 - Put – “keep the data please”
 - Delete – “delete the data please”
 - +5 more
-

Technically – “Nothing is impossible”

“Always be prepared...”

HTTP Protocol

- Examples where http methods are used properly:
 - in public web services
 - in projects where coding best practices are strictly followed
- Examples where http methods can be messed up:
 - in http server based web applications
 - in projects where best practices aren't strictly enforced
- “Bad” practices that you can face
 - use POST for searches
 - use POST to delete something
 - never use DELETE, PUT
 - etc.

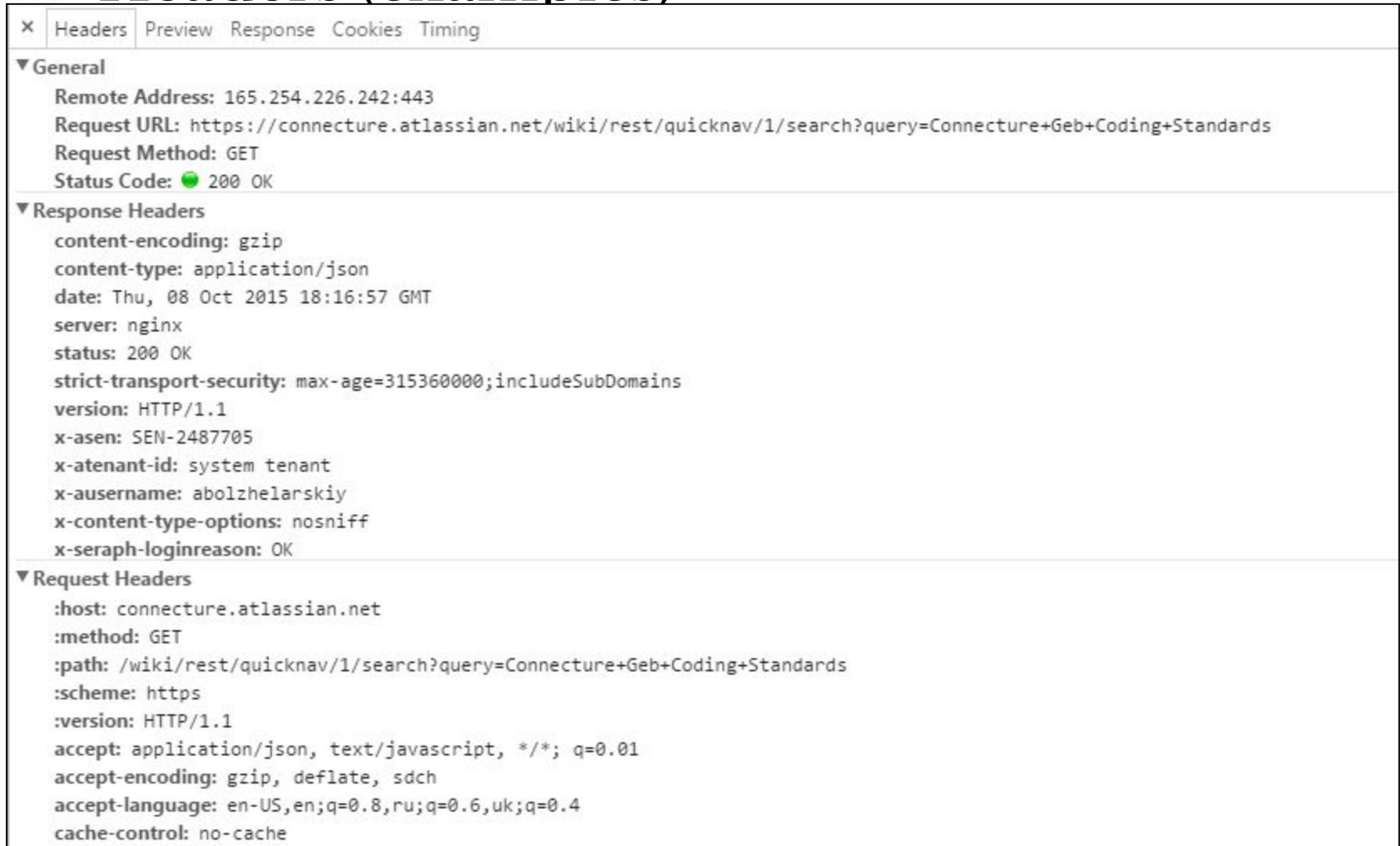
“Always be prepared...”

HTTP Protocol

- Headers
 - format – [header name]:[header value]
 - groups
 - General Headers – must be present in ALL requests
 - Request Headers – are present in CLIENT requests only
 - Response Headers – are present in SERVER responses only
 - Entity Headers – details related to content of a request/response
 - some headers belong to few groups

HTTP Protocol

● Headers (examples)



The image shows a screenshot of a web browser's developer tools, specifically the 'Headers' tab. The interface includes a tab bar with 'Headers', 'Preview', 'Response', 'Cookies', and 'Timing'. The 'General' section shows the remote address as 165.254.226.242:443, the request URL as https://connecture.atlassian.net/wiki/rest/quicknav/1/search?query=Connecture+Geb+Coding+Standards, the request method as GET, and the status code as 200 OK. The 'Response Headers' section lists various headers including content-encoding: gzip, content-type: application/json, date: Thu, 08 Oct 2015 18:16:57 GMT, server: nginx, status: 200 OK, strict-transport-security: max-age=315360000;includeSubDomains, version: HTTP/1.1, x-asen: SEN-2487705, x-tenant-id: system tenant, x-username: abolzhelarskiy, x-content-type-options: nosniff, and x-seraph-loginreason: OK. The 'Request Headers' section lists headers such as :host: connecture.atlassian.net, :method: GET, :path: /wiki/rest/quicknav/1/search?query=Connecture+Geb+Coding+Standards, :scheme: https, :version: HTTP/1.1, accept: application/json, text/javascript, */*; q=0.01, accept-encoding: gzip, deflate, sdch, accept-language: en-US,en;q=0.8,ru;q=0.6,uk;q=0.4, and cache-control: no-cache.

```
X Headers Preview Response Cookies Timing
▼ General
Remote Address: 165.254.226.242:443
Request URL: https://connecture.atlassian.net/wiki/rest/quicknav/1/search?query=Connecture+Geb+Coding+Standards
Request Method: GET
Status Code: 200 OK

▼ Response Headers
content-encoding: gzip
content-type: application/json
date: Thu, 08 Oct 2015 18:16:57 GMT
server: nginx
status: 200 OK
strict-transport-security: max-age=315360000;includeSubDomains
version: HTTP/1.1
x-asen: SEN-2487705
x-tenant-id: system tenant
x-username: abolzhelarskiy
x-content-type-options: nosniff
x-seraph-loginreason: OK

▼ Request Headers
:host: connecture.atlassian.net
:method: GET
:path: /wiki/rest/quicknav/1/search?query=Connecture+Geb+Coding+Standards
:scheme: https
:version: HTTP/1.1
accept: application/json, text/javascript, */*; q=0.01
accept-encoding: gzip, deflate, sdch
accept-language: en-US,en;q=0.8,ru;q=0.6,uk;q=0.4
cache-control: no-cache
```

HTTP Protocol

- Message body
 - Optional
 - Content
 - Overall: any text can be sent
 - In particular: text that target server understands
 - + Entity Headers if/where needed

HTTP Protocol

- ~~Attachments~~ Message body
 - Sent in forms
 - Key http headers:
 - Content-Type: multipart/form-data
 - Content-Disposition: form-data
 - Content-Type:[text/plain | application/x-object | etc.]

HTTP Protocol

- Message body (example)

```
POST /send-message.html HTTP/1.1
Host: webmail.example.com
Referer: http://webmail.example.com/send-message.html
User-Agent: BrowserForDummies/4.67b
Content-Type: multipart/form-data; boundary=Asrf456BGe4h
Content-Length: 1831983
Connection: keep-alive
Keep-Alive: 300
```

```
--Asrf456BGe4h
Content-Disposition: form-data; name="DestAddress"
```

```
brutal-vasya@example.com
--Asrf456BGe4h
Content-Disposition: form-data; name="MessageTitle"
```

```
--Asrf456BGe4h
Content-Disposition: form-data; name="MessageText"
```

```
Hellow! Checkout some pictures from yesterdays party.
It was the best party ever. You guys are awesome!
Here are the photos
```

```
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile1"; filename="horror-photo-1.jpg"
Content-Type: image/jpeg
```

```
(... contents of file goes here ...)
```

```
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile2"; filename="horror-photo-2.jpg"
Content-Type: image/jpeg
```

```
(... contents of file goes here ...)
```

```
--Asrf456BGe4h--
```

HTTP Protocol

● Message body (example)

Request header

1

2 empty lines

Message body

```
POST /send-message.html HTTP/1.1
Host: webmail.example.com
Referer: http://webmail.example.com/send-message.html
User-Agent: BrowserForDummies/4.67b
Content-Type: multipart/form-data; boundary=Asrf456BGe4h
Content-Length: 1831983
Connection: keep-alive
Keep-Alive: 300
```

Delimiter of form fields

```
--Asrf456BGe4h
Content-Disposition: form-data; name="DestAddress"

brutal-vasya@example.com
--Asrf456BGe4h
Content-Disposition: form-data; name="MessageTitle"

--Asrf456BGe4h
Content-Disposition: form-data; name="MessageText"
```

Content of a particular field

```
Hello! Checkout some pictures from yesterdays party.
It was the best party ever. You guys are awesome!
Here are the photos
```

Empty line between headers and content

```
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile1"; filename="horror-photo-1.jpg"
Content-Type: image/jpeg
```

```
(... contents of file goes here ...)
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile2"; filename="horror-photo-2.jpg"
Content-Type: image/jpeg
```

Content of a file is sent as binary code

```
(... contents of file goes here ...)
--Asrf456BGe4h--
```

HTTP Protocol

- Status codes
 - 1xx Informational
 - Request received, continuing process
 - 2xx Success
 - Request was received, understood, accepted and processed successfully
 - 3xx Redirection
 - Client must take additional action to complete the request
 - 4xx Client Error
 - Client seems to have erred
 - 5xx Server Error
 - The server failed to fulfil an apparently valid request

HTTP Protocol

- Status codes:
 - We knew it!:
 - 200 – everything is OK
 - 401 – something is wrong with sent credentials
 - 404 – requested page is absent
 - 500 – server is down
 - Do you know more codes?
 - 1xx – 3 codes
 - 2xx – 10 codes
 - 3xx – 10 codes
 - 4xx – 43 codes
 - 5xx – 16 codes

Regular expressions

- Regular expressions are quite easy to learn

Regular expressions

- But doing this by slides is as easy...

Regular expressions

1. 本字典供查阅简化字与繁体字对照关系之用,只识简化字的读者可以从中查出相应的繁体字,只识繁体字的读者也可以从中查出相应的简化字。

2. 本字典收入《简化字总表》(1986年版)中的全部简化字,共2235个。《第一批异体字整理表》中的39个选用字习惯上被看作简化字,本字典也全部收入。

3. 本字典以简化字为字头,横线后列出相应的繁体字。字典解说用简化字。

4. 字典正文按汉语拼音音序排列。正文前面有汉语拼音音节索引。同音字按笔画数多少排列,少的在前,多的在后。笔画数相同的,按起笔笔形横、竖、撇、点、折的顺序排列。正文前面还有笔画检字索引,包括两部分:从简体查繁体,从繁体查简体。

● ...as learning Chinese

第六节 民族自治地方的自治机关

第一百一十二条 民族自治地方的自治机关是自治区、自治州、自治县的人民代表大会和人民政府。

第一百一十三条 自治区、自治州、自治县的人民代表大会中,除实行区域自治的民族的代表外,其他居住在本行政区域内的民族也应当有适当名额的代表。

自治区、自治州、自治县的人民代表大会常务委员会中应当有实行区域自治的民族的公民担任主任或者副主任。

第一百一十四条 自治区主席、自治州州长、自治县县长由实行区域自治的民族的公民担任。

第一百一十五条 自治区、自治州、自治县的自治机关行使宪法第三章第五节规定的地方国家机关的职权,同时依照宪法、民族区域自治法和其他法律规定的权限行使自治权,根据本地方实际情况贯彻执行国家的法律、政策。|

Regular expressions

- You will have to learn it by yourself

Regular expressions

- Theory
 - https://en.wikipedia.org/wiki/Regular_expression
 - <http://www.rexegg.com/>
- Interactive online tutorial
 - <http://regexone.com/>
- Online regular expression `...`
 - <http://rubular.com/>
 - <http://www.regexr.com/>
 - <https://regex101.com/#javascript>
- Desktop regular expression `...`
 - <http://www.weitz.de/regex-coach/>



**it's easy
if you try.**

Regular expressions

- But here is some basic understanding

Regular expressions

- What to search
- Where to search
- How many times it should appear
- Searching algorithm

Regular expressions

- What to search
- Characters `abc123`
 - *use backslash for meta characters!* `^$.?*`
 - Character classes `[]`
 - Groups `()`
 - Alternatives `|`

Regular expressions

- Where to search
 - Start/End of text `^` and `$`
 - Start/End of a word `\b`
 - After text `?=<`
 - Before text `?=`

Regular expressions

- How many times it should appear (quantification)
- Particular number of times $\{n\}$
- Range $\{m, n\}$
- Not less than $\{m, \}$
- Not more than $\{,n\}$
- Zero or one time $\{?\}$
- Zero or any number of times $\{*\}$
- One or more times $\{+\}$

Regular expressions

- Searching algorithm
 - **Greedy**
 - repeat a quantifier as many times as possible
 - **Lazy**
 - Repeat a quantifier as little as possible

Regular expressions

Example: *“stress” testing and “capacity” testing are not the same*

Take anything that starts from a quote and ends with a quote, and it doesn't matter what is between the quotes



`/".*/"`

“stress” testing and “capacity”



● Greedy

He said “anything” – I’ll take “everything”

`/".*?/"`

“stress”, “capacity”



● Lazy

The shorter the better

Tools that can help

- SoapUI
 - Check available functions on an endpoint (WADL/WSDL)
 - List of all functions
 - Structure of request/response
 - Try your requests before using them in JMeter
- Developer Tools (Chrome browser)
 - Compare requests in your tests with requests sent by application
 - Headers
 - Content
 - Cookies

Tools that can help

- SoapUI <http://www.soapui.org>
- Chapters recommended for reading/watching
 - About SoapUI
 - Videos
 - Functional Testing
 - REST Testing
 - Getting started
 - Installing SoapUI
 - Your First SoapUI Project
 - REST Testing
 - SOAP and WSDL
 - Operations and Requests
 - Headers and Attachments
 - REST
 - Understanding REST Parameters

Tools that can help

- Chrome Developer Tools

<https://developer.chrome.com/devtools/docs/network>

- Network tab is the most important here
- **Note!** Even if you have been working with Chrome Developer Tools for ages we strongly recommend you to read the tool documentation anyway.

Questions?