24 Kasım 2020

selsoft
build better, deliver faster

Developing RESTful Web Services with Java

Chapter 1: Fundamentals of Web

Eğitmen:

Akın Kaldıroğlu

Çevik Yazılım Geliştirme ve Java Uzmanı

Topics

- **Some Terminology on Web**
- **Web Servers**
  - Tomcat
- **Application Servers**
  - Glassfish
- **HTTP**
  - URL & URI
  - Properties of HTTP
  - Operations of HTTP
  - Request & Response

Some Terminology on Web

selsoft
build better, deliver faster

Britannica Says

World Wide Web (**WWW**), byname the **Web**, the leading information retrieval service of the Internet (the worldwide computer network). The Web gives users access to a vast array of documents that are connected to each other by means of **hypertext** or **hypermedia** links—i.e., hyperlinks, electronic connections that link related pieces of information in order to allow a user easy access to them.

The development of the World Wide Web was begun in 1989 by Tim Berners-Lee and his colleagues at CERN, an international scientific organization based in Geneva, Switzerland. They created a protocol, **HyperText Transfer Protocol** (**HTTP**), which standardized communication between servers and clients. Their text-based Web **browser** was made available for general release in January 1992.

HTML - I

- **HTML** stands for **Hypertext Markup Language** and is a tagging langauge that presents text, graphics, video, etc. in a linked manner.
- **Hypertext** is text which contains links to other texts.
  - The term was coined by Ted Nelson around 1965.
- **HyperMedia** is a term used for hypertext which is not constrained to be text: it can include graphics, video and sound.

HTML - II

- HTML is a specialized **Standard Generalized Markup Language** (**SGML**) and is used to structure web pages.
- Markup is used labeling parts of a document in order to describe them.
  - LaTeX, HTML, XML all use markups.
  - In this sense HTML and XML are cousins.
- HTML resources are mainly HTML files that describes text with tags to represent both text and other resources referenced by links in the text.

Web Site vs. Web Application

- A **web site** is a collection of connected static pages structured using HTML and linked with other kinds of resources such as graphics, videos, etc.
- A **web application** (or **web app**) is a web site with a dynamic functionality.
  - From now on the term *web application* or *web app* will be used.
- In web apps, the content of most of HTML pages are produced dynamically by programs based on user's inputs & clicks from the browser.
- Web apps are mostly backed by databases and integrated with some other information systems.

Java Web Applications

- There are many technologies such as programming languages and frameworks to create web applications.
- In this course we will see Servlet, JSP, and JSF as standard web components of Java EE to create web applications.
- There are several other frameworks to develop web applications in Java world such as Struts, Spring MVC, Vaadin, GWT, etc.

Web Container/Server and Browser

- A web app runs in a **web container** that manages the lifecycle of resources in that web app and serves them when asked.
- That container is called **web server**.
- A web server is a piece of software that runs on servers, on-premise physical machines or more abstract environments such as cloud servers and serves the content of web apps to its clients.
- A **browser** is a client software that runs on desktops or mobile devices and renders what is served by web apps.

Web Servers

- Well-known web servers are:
  - Apache Apache
  - MS IIS
  - Apache Tomcat
  - Nginx
  - Lighttpd

HTTP

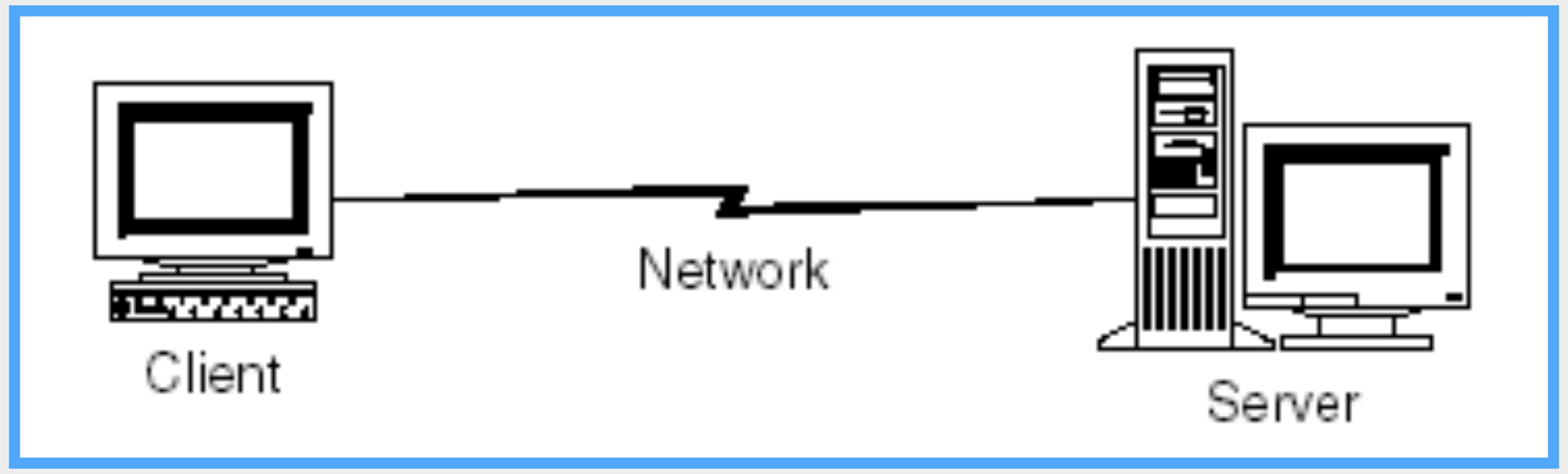
- **HyperText Transfer Protocol** (**HTTP**) is a special protocol for the communication between web apps and their clients.
- It specifies how a client should format and send a request to a web app residing in a web server and a web server should format and send a response produced by a web app back to its client.
  - Web servers are also called **HTTP server**.
- HTTP uses port 80 as its standard port.

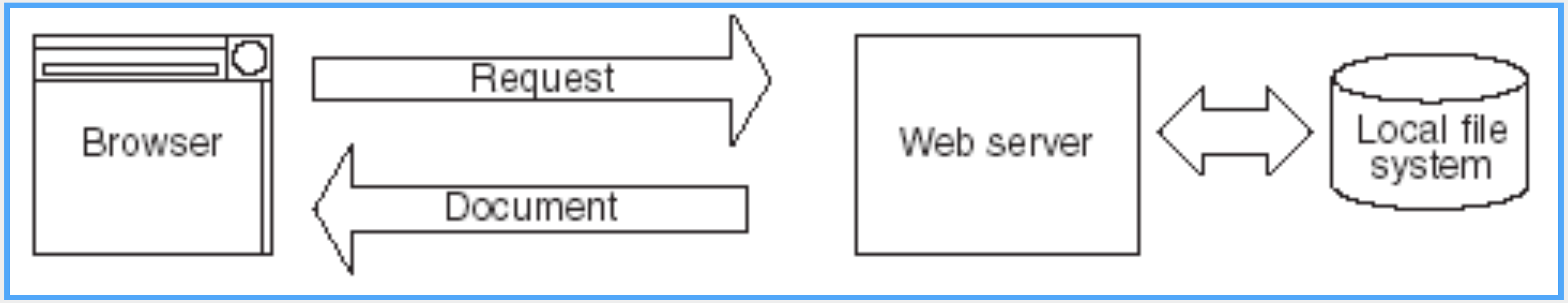
Web Clients

- Web apps are architecturally in client-server style.
- At the beginning of web phonemenon clients were all browsers that allowed the users interact with web apps.
  - So users were the main clients of web apps.
- But in time web apps also became the clients of each other.
  - This is totally web-based integration among applications.
- And of course web apps don’t use browsers to reach each other.

Client-Server Architecture of Web

- So **World Wide Web** or shortly Web is about communication between web clients and web servers through HTTP.

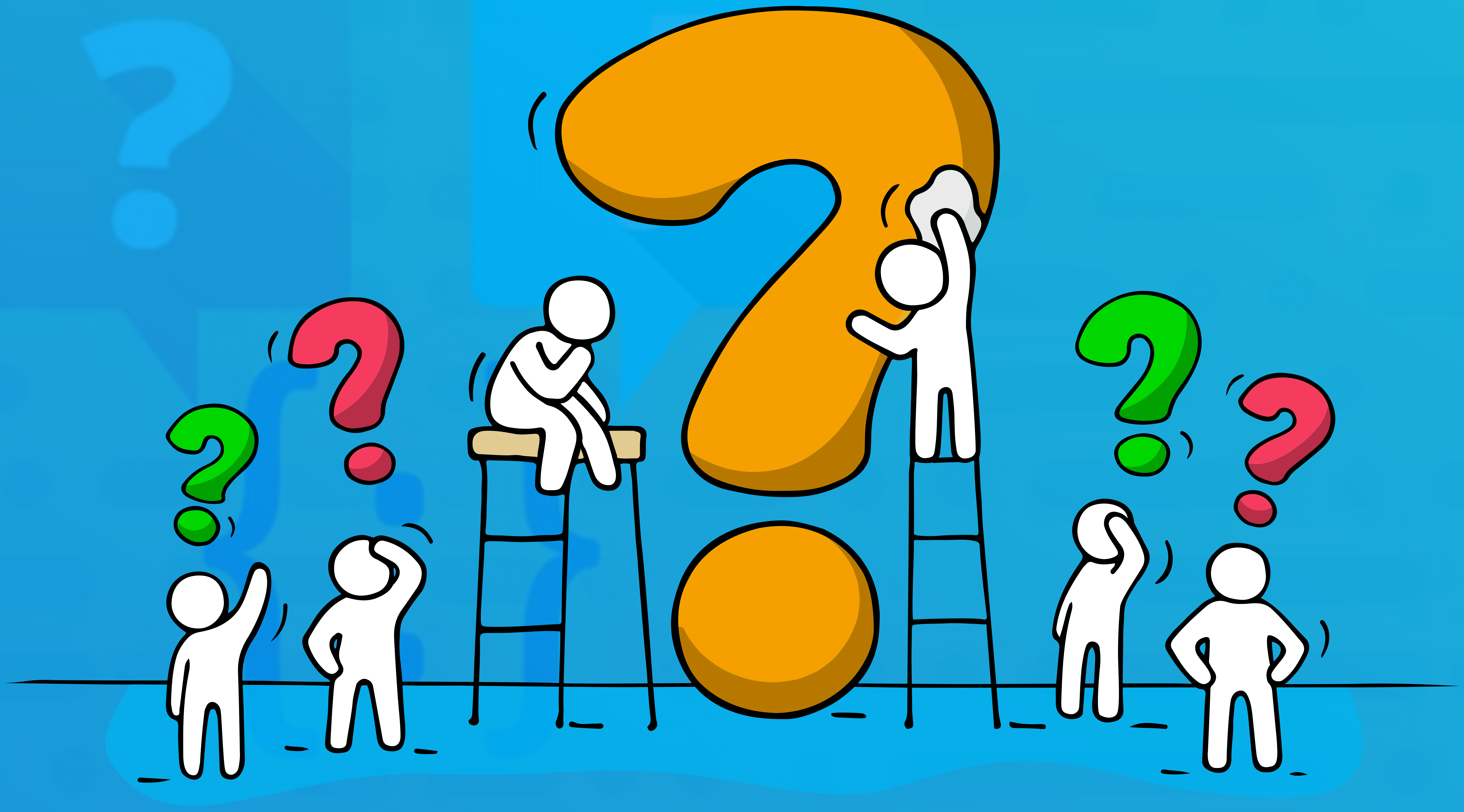




Web Service

- A web-based integration among systems is generally called **web service**.
- Web services provide system-to-system communication mechanism for the purpose of transfer of service.
- Web service is a solution to the problem of enterprise application integration (EAI).
- Why are they called web service?
  - Because web services use mainly web technologies such as HTTP, HTML along with others such as XML, JSON, SOAP, etc.

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

Web Servers

selsoft
build better, deliver faster

Web Servers

- As noted before web apps run on a piece of software called web server.
- Web servers are software solutions and products.
- There are many web servers with different technogical backgrounds and capabilities.
- In Java terminology web server is also called **servlet container**.
    - That's because servlet technology has always been the main vehicle to implement other web technologies in Java.

Tomcat and Jetty

- There are many open-source or commercial web servers for Java.
- The most known web servers are also open-source:
  - Apache Tomcat http://tomcat.apache.org/index.html
  - Eclipse Jetty https://www.eclipse.org/jetty/

Java Web Application Packaging

- Java web apps are packaged mostly in **WAR** files.
    - WAR stands for **Web ARchive**.
- It is also possible to package Java web apps in **EAR** files.
    - EAR stands for **Enterprise ARchive**.
    - EARs should be used if the web application includes some other enterprise Java components such as EJBs.
- We mostly package our web apps in war files in this course.

selsoft
build better, deliver faster

Tomcat

Tomcat

- **The Jakarta Tomcat w**eb server is an open source, Java-based servlet container.
    - http://tomcat.apache.org/index.html
- It exists under the Apache-Jakarta subproject, where it is supported and enhanced by a group of volunteers from the open source Java community.
- It is the most used web server in Java world.

Tomcat Versions

- Version 9 is the one that supports Java EE 8 which includes Servlet 4.0 specification.
  - It requires min Java 8.
- Version 10 is the new release that supports Java EE 9 which will include Servlet 5.0.
- Version 8 is the one that supports Java EE 7 which includes Servlet 3.1.
- For more info https://cwiki.apache.org/confluence/display/TOMCAT/Specifications and https://tomcat.apache.org/whichversion.html

selsoft
build better, deliver faster

Exercise

Exercise

- Download latest **Tomcat 9** installation from its site.
  - You can download it as a zip file and open anywhere you want.
- Go to its **`bin`** directory and start it using **`startup`** script.
  - Tomcat 9 needs min Java 8 and it first looks for an available JRE or JDK and if not found it looks for JAVA_HOME.
  - If Tomcat can't find any Java executable it can't start.
- After starting it go to http://localhost:8080 (or http://127.0.0.1:8080/) in browser to see its main page.

Exercise

- Then try to go to its *Server Status* and *Manager App* applications.
  - To be able to see related pages configure users in its **`conf/tomcat-users.xml`** file.
- Have a look at its examples.
- Have a look at its configuration in **`conf/server.xml`**.
  - Change the port for HTTP connector from 8080 to from example to 7070.

Exercise

- Configure Eclipse to use Tomcat.
- Create a simple **Dynamic Web Application** in Eclipse and deploy it to both Tomcat configured in Eclipse and standalone Tomcat using its war file.
- Go to its `bin` directory and shut it down using `shutdown` script.
- And notice that when you change settings or code other than HTML and JSP, Tomcat requires a republish or restart which becomes visible in Servers tab.

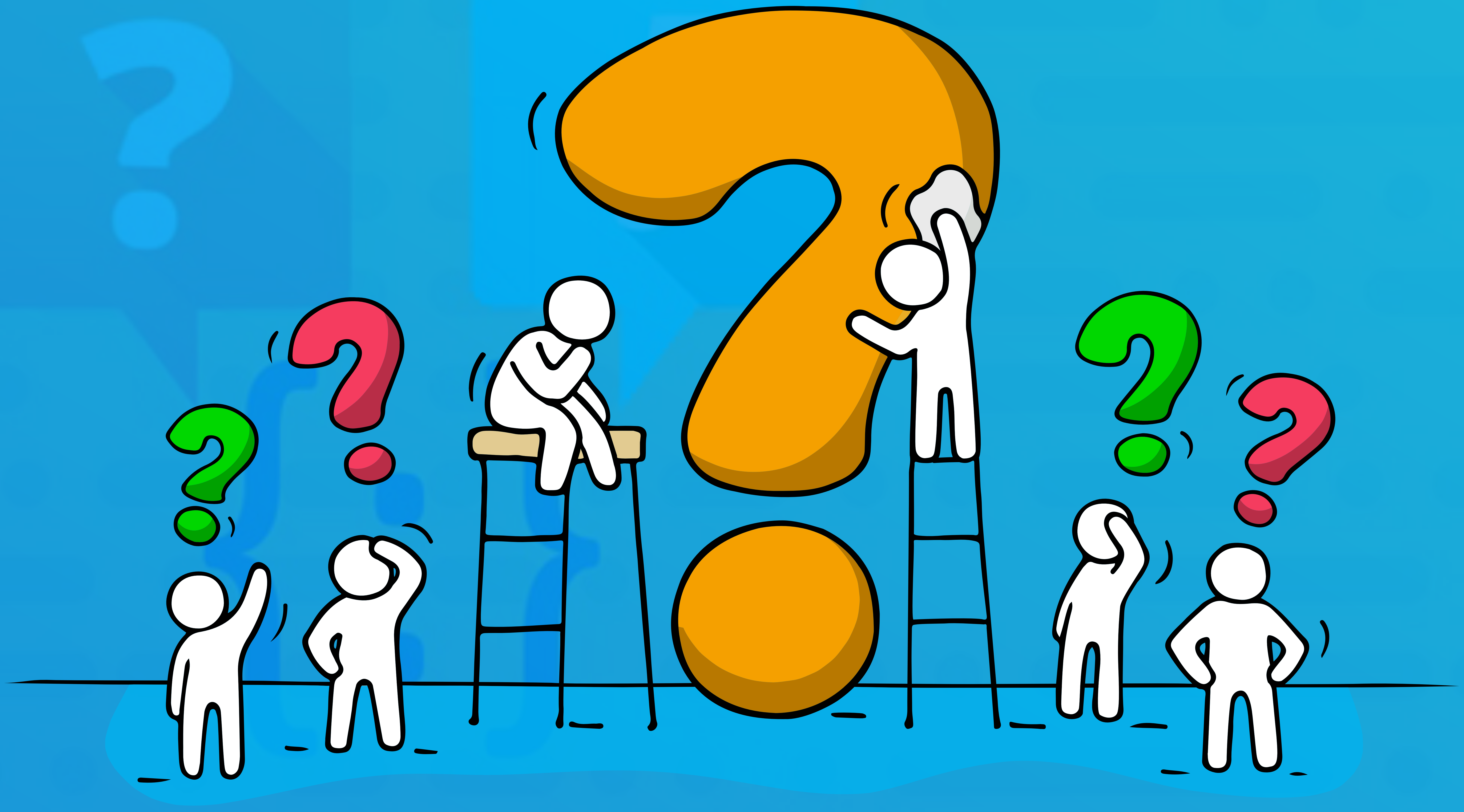
WAP4.0

- Import the project **WAP4.0** into Eclipse and run it on Tomcat.
- Configure its **`org.javaturk.wap.util.SourceCodeServlet`** to be able to serve source codes.
    - For this purpose read **`ReadMe.html`**.

CurrencyConverter

- Import the projects **CurrencyConverter** and **CurrencyConverterTest** into Eclipse and run it on Tomcat.
- Test it using SoapUI.

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

Application Servers

selsoft
build better, deliver faster

Application Servers - I

- There is another kind of container in Java: **EJB container**
- EJB container is a container that manages EJBs and other enterprise components such as JSF, JPA, JMS, etc.
- The server that both includes servlet and EJB containers is called **application server** or **app server**.
    - So app servers include web servers and more and they are are more complicated than web servers.

Application Servers - II

- There are many open-source app servers for Java.
  - Glassfish http://tomcat.apache.org/index.html
  - Apache TomEE https://tomee.apache.org/index.html
  - Apache Geronimo http://geronimo.apache.org/
  - RedHat WildFly (previously JBoss) https://www.wildfly.org/

Application Servers - III

- These are also very-well known commercial app servers:
  - Oracle WebLogic https://www.oracle.com/middleware/technologies/weblogic.html
  - IBM WebSphere https://www.ibm.com/cloud/websphere-application-server

Web Servers vs. App Servers - I

- Application servers include web servers and more.
- Web servers mostly come in a lighter packaging even in terms of web technologies too.
- For example Tomcat can serve servlets and JSPs but can’t serve JAX-RS web services in default, it needs an implementation of JAX-RS i.e. its jars added to be able to do that.
    - Same thing is true for most of Java EE technologies such as JPA, JAX-WS, etc.

Web Servers vs. App Servers - II

- But adding implementations of different Java EE components to a web server does not make it an app server.
  - For example EJB container is a totally different engine that is at the core of the app servers.
- App servers also provide different performance, scalability, security, managability, etc. capabilities.
- So even though a web server is technologically enough, an app server can be more suitable for more demanding applications for example in terms of scalability.
- That’s why web servers are much lighter than app servers in Java.

Web Servers vs. App Servers - III

- That’s why web servers are much lighter than app servers in Java.

selsoft

build better, deliver faster

Glassfish

Glassfish - I

- Glassfish is the reference implementation, RI, of Java EE.
- It has free community and commercial versions until 5.0.1 under the patronage of Oracle.
  - https://javaee.github.io/glassfish/
- It has been moved to Eclipse Foundation.
  - https://projects.eclipse.org/projects/ee4j.glassfish
  - So use this address to follow it.

Glassfish - II

- Glassfish 5.1 is for Java EE 8.
- Galssfish 6 will be for Java EE 9 whch is planned for its release in Dec. 2020.

selsoft

build better, deliver faster

Exercise

Exercise

- Download **Glassfish 5.1** installation from its site.
  - You can download it as a zip file and open anywhere you want.
- Go to its `glassfish/bin` directory and start it using `startserv` script.
- Then in browser go to http://localhost:4848 (http://127.0.0.1:4848) to see its admin page.
- Look around its menu and pages such as Nodes, Applications and Resources.

Exercise

- Install **Glassfish Tools** using https://download.eclipse.org/glassfish-tools/1.0.0/repository/ update site.
- Then configure Glassfish to use within Eclipse.

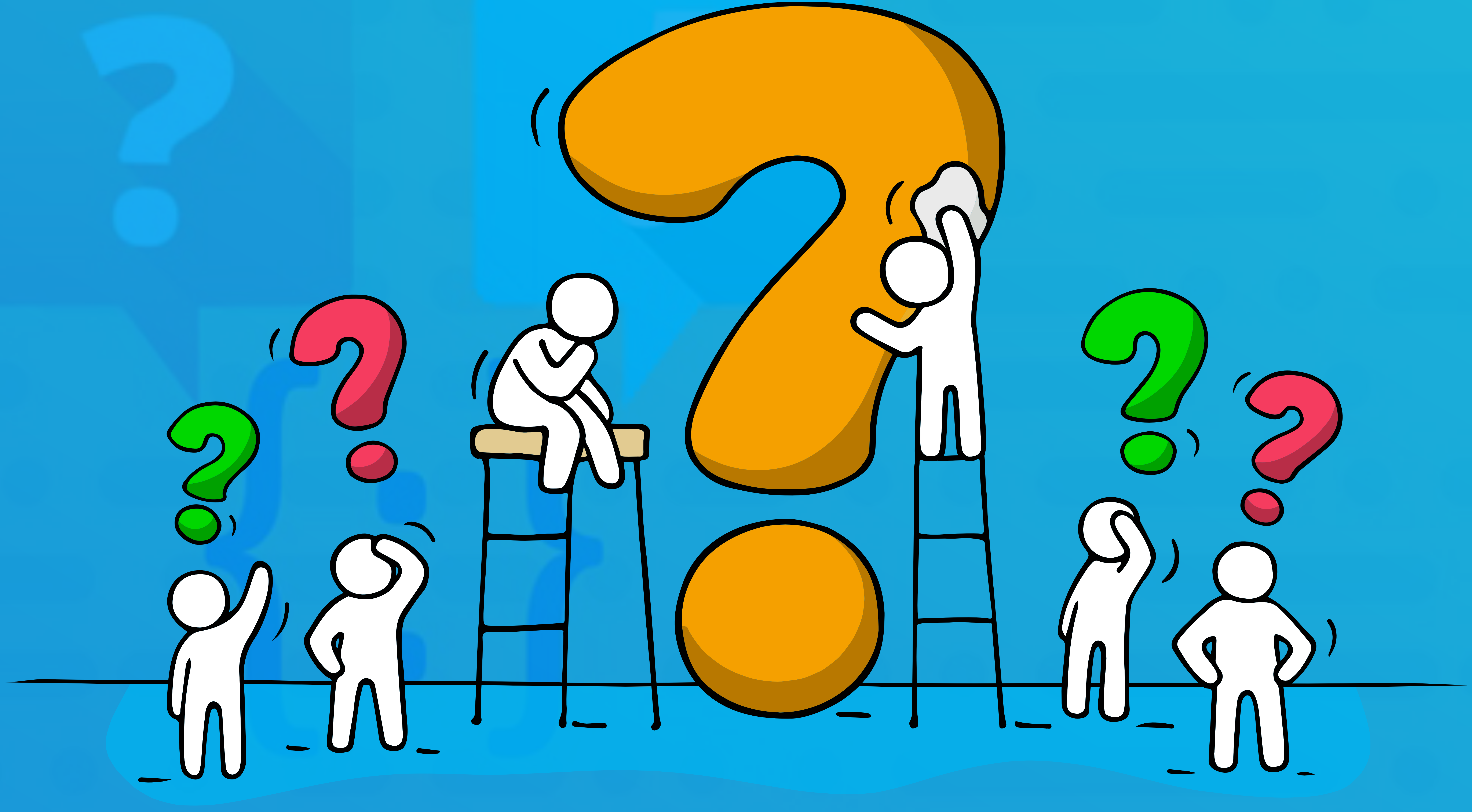
Exercise

- Create a simple Dynamic Web Application in Eclipse and deploy it to both Glassfish configured in Eclipse.

WAP4.0

- Run WAP4.0 on Glassfish.

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr

selsoft.com.tr

HTTP

selsoft
build better, deliver faster

Protocol

- A **protocol** is a set of rules or standards designed to enable computers to connect with one another and to exchange information with as little error as possible
- When two system communicate, their protocols interface directly with other protocols within each individual system.
- In fact protocols in each layer communicate with their peers in the other system.

HTTP - I

- **Hypertext Transfer Protocol**, **HTTP**, is a protocol to transfer hypertext documents.
- At the beginning, in HTTP/0.9 it was the main idea.
  - But HTTP can transfer other file types too including images, sound, etc.
- Development of HTTP was started by Tim Berners-Lee at CERN in 1989.
- HTTP/0.9 was documented in 1991 and HTTP/1.1 in 1997.
- HTTP/2.0 was published in 2015 and HTTP/3.0 has already started experimentally.

HTTP - II

- HTTP is an application level protocol in OSI layers.
- It has its own clients and servers:
  - Typical HTTP client is a browser
  - Typical HTTP server is a web server
- The most obvious difference between HTTP clients and servers is responsibility for initiating communication.

Initiation

- Only a client can initiate a communication in HTTP.
- A web server does something only when asked to do so by a client.
- So an HTTP client acts and an HTTP server reacts.
- In other words, a client sends a request to the Web server and the Web server produces a response for it.
- So **request** and **response** are two main objects of an HTTP communication.

Message Structure - I

- There are two kinds of messages in HTTP:
  - Request
  - Response
- Request and response messages have similar structures:

```
<method> <URL> <HTTP version>
<headers>

<entity-body>
```

```
<HTTP version> <status> <status-phrase>
<headers>
<entity-body>
```

Message Structure - II

- Status is a three-digit status code and status phrase is an explanatory phrase.
- A header is a name-value(s) pair that gives more information regarding the message to the server or client.

```
<method> <URL> <HTTP version>
<headers>

<entity-body>
```

```
<HTTP version> <status> <status-phrase>
<headers>
<entity-body>
```

Headers - I

- HTTP requests and responses may include one or more message headers
- HTTP headers provides extra information about their associated objects, requests or responses
- You can consult **RFC 2616** and **7540** about comprehensive HTTP header information
- Header names are not case sensitive

Headers - II

- There are three types of headers:
  - **General headers** apply to HTTP communication in general such as `Date`.
  - **Request/Response headers** apply to specific request or response.
    - All `Accept` headers except `Accept-Ranges` are request headers.
    - `Accept-Ranges` is a response header

Headers - III

- **Entity headers** apply to the message body included in the request or the response such as `Last-Modified` or `Content-Type`.

Headers - IV

- Message headers begin with a field name and a colon, : and then field value follows
- Sometimes field name itself is sufficient and there would be no colon and information
- If there is more than one piece of information such as a list, a comma is used

Header Examples

```
GET /WAP4.0/SelamServlet HTTP/1.1
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: JSESSIONID=3a8dfc71ce09c25e6be6abfac03f; JSESSIONID=394967ca90899fe7d12078489d4b;
treeForm_tree-hi=treeForm:tree:applications
Upgrade-Insecure-Requests: 1
```

Header Examples

```
HTTP/1.1 301 Moved Permanently
Server: GlassFish Server Open Source Edition  5.1.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition  5.1.0  Java/Oracle Corporation/1.8)
Location: http://localhost:4040/WAP4.0/
Content-Language: en-TR
Content-Type: text/html;charset=ISO-8859-1
Connection: close
Content-Length: 179
```

```
HTTP/1.1 200 OK
Server: GlassFish Server Open Source Edition  5.1.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition  5.1.0  Java/Oracle Corporation/1.8)
Accept-Ranges: bytes
ETag: W/"17208-1605990230000"
Last-Modified: Sat, 21 Nov 2020 20:23:50 GMT
Content-Type: text/html
Connection: close
Content-Length: 17208
```

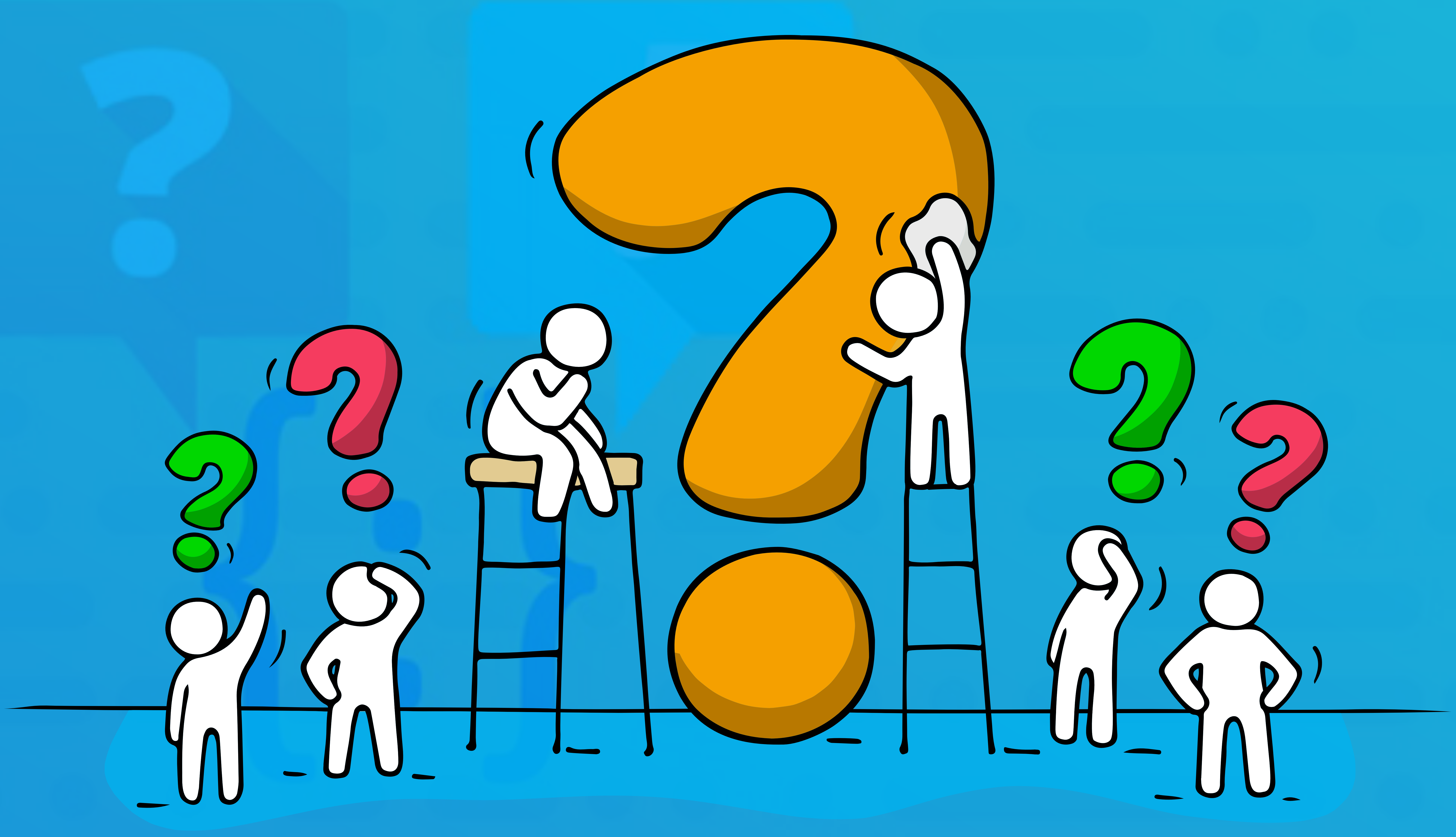
WebClient

- Run `org.javaturk.wap.util.webclient.WebClient` to send HTTP requests.

EchoServer

- Run `org.javaturk.wap.util.echoserver.EchoServer` which listens to port 8888.
- It just echos back whatever submitted as an HTTP request.

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

selsoft

build better, deliver faster

Request

Parts of Request - I

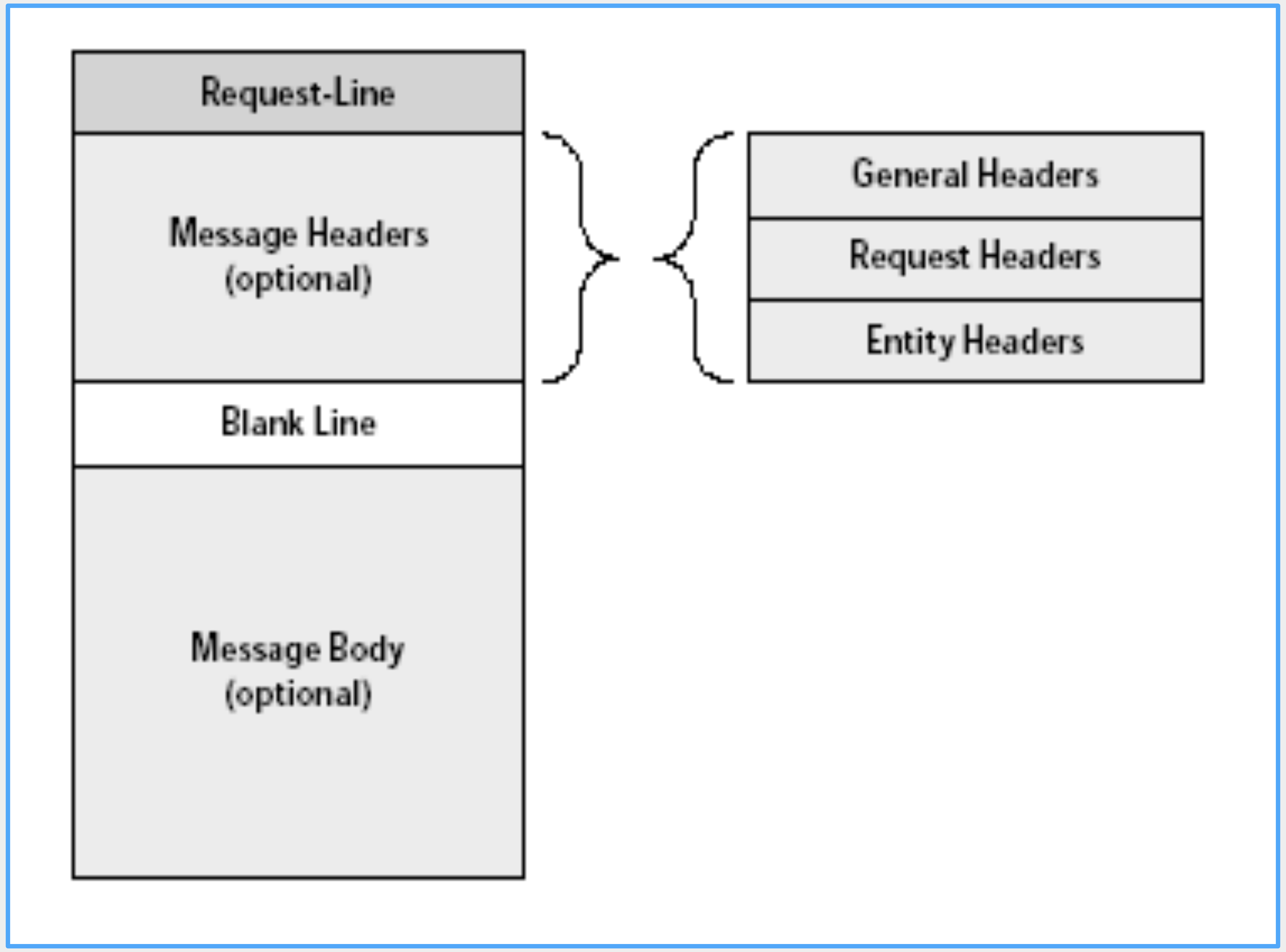
- A request has three parts:
  - A request line that consist of following three items:
    - HTTP method
    - URL
    - HTTP version
  - Headers
  - Entity body as arbitrary data

```
<method> <URL> <HTTP version>
<headers>

<entity-body>
```

Parts of Request - II

- An HTTP request always includes a blank line after the request line and any included headers.
  - It is used by the server to indicate the end of the request headers.



An Example Request

```
GET /WAP4.0/SelamServlet HTTP/1.1
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: JSESSIONID=3a8dfc71ce09c25e6be6abfac03f; JSESSIONID=394967ca90899fe7d12078489d4b;
treeForm_tree-hi=treeForm:tree:applications
Upgrade-Insecure-Requests: 1
```

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.ft.com
Connection: Keep-Alive
        Blank Line
```

selsoft

build better, deliver faster

Request

Method

HTTP Methods

- HTTP defines several methods for different kinds of request:
  - GET
  - POST
  - HEAD
  - PUT
  - DELETE
  - CONNECT
  - TRACE
  - OPTIONS
  - PATCH

Safety & Idempotency - I

- There are two important points for HTTP methods:
  - **Safety**: Safe methods don’t cause any change on the statet of the server.
    - GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval.
  - **Idempotency**: Calling the same method at different times should result in the same response.
    - The methods GET, HEAD, PUT and DELETE share this property. Also, the methods OPTIONS and TRACE SHOULD NOT have side effects, and so are inherently idempotent.

Safety & Idempotency - II

- Safe methods are supoosed to be read-only methods.
- Idempotent methods whether called once or many times, should always produce the same effect on the server side.
- Safe methods are also idempotent but not vice versa.
- DELETE is idempotent meaning that the resource no longer exists on the server side.
- PUT is idempotent meaning that the resource exists on the server side.

Safety & Idempotency - III

Method	Safe	Idempotent
GET	yes	yes
HEAD	yes	yes
PUT	no	yes
POST	no	no
DELETE	no	yes
TRACE	yes	yes
PATCH	no	no
OPTIONS	yes	yes

GET & HEAD

- GET is used to retrieve the resource expressed in URL and the server sends back the resource if exist in response.
- HEAD is the same as GET except that the server only returns headers, it does not send the resource itself.
  - It is used to check if a resource exists or if a resource has been changed or not for caching purpose, etc.
- Both are safe and idempotent methods

```
GET /index.html HTTP/1.1
HEAD /index.html HTTP/1.1
```

GET & HEAD

- Using **WebClient** send a GET and HEAD requests to www.google.com.tr.
- Use **Firefox Developer** or **Google Chrome** to inspect request and response.

POST

- POST is used to send data to the server.
- So it is neither safe nor idempotent.
- It is always used in forms to submit data.
- Submitted data is called **request parameters**.

POST

- Use **WebClient** send a POST requests to **WAP4.0** deployed on a server.
  - Use **`postFormServlet.html`** in ch07 (http://localhost:4040/WAP4.0/html/ch07/postFormServlet.html)
- Use **Firefox Developer** or **Google Chrome** to inspect request and response when posting using **`postFormServlet.html`**.

PUT

- PUT is inverse of GET and is used to put a new document or change an existing one on the server.
- It is not safe but idempotent.
- It is mostly used to upload a new HTML page or change the existing one for example in sites that alow to manage pages remotely.

DELETE

- DELETE asks the server delete the resource.
- Servers are free to decide whether it should be deleted or not.
- It is not safe but idempotent.
- It is mostly disabled or not allowed.

DELETE

- Use **WebClient** send a DELETE requests to different servers.

TRACE

- TRACE allows tracing the request on the server side.
- There might lots of intermediaries such as firewall, gateway, proxy, etc. the request goes through and those intermediaries may change the request.
- SO TRACE method is used to invoke a remote, application-layer loop-back of the request message.
- It is both safe and idempotent.
- It is mostly disabled.

TRACE

- Use **WebClient** send a TRACE requests to different servers.

OPTIONS

- OPTIONS is used to list the capabilities of the server.
- Generally servers send back the methods they support.
- It is both safe and idempotent.
- It is sometimes disabled.

OPTIONS

- Use **WebClient** send a OPTIONS requests to different servers.

Request

URI & URL

URI vs. URL

- **Uniform Resource Identifier** or **URI** is the textual description of an object on the Internet.
    - It is kind of a name for a resource on the Internet.
- **Uniform Resource Locator** or **URL** describes an object by giving its location on the Internet, including the server storing the object, the application protocol needed to retrieve it, and the name of the object.
    - Its format is as follows

```
http: // hostname [:port] / path [;parameters] [?query]
https: // hostname [:port] / path [;parameters] [?query]
```

Uniform Resource Locator - URL - I

- **URL** is what Tim Berners-Lee invented along with HTTP.
- It has mainly three parts:
  - **URL Scheme is** what is before `://` such as **`http`** or **`https`** for HTTP
  - **Host**, what is after `://` such as www.google.com or www.selsoft.com.tr
    - Host's IP address is looked up at an DNS server so its IP can also be used instead of host.

Uniform Resource Locator - URL - II

- **URL path**, what is after the host such as `/egitimler` or `/search?q=java&oq=java+&aqs=chrome`...
- URL path is case sensitive.
  - But most web servers behave case-insensitive just to avoid broken links due to mistakes in paths.

Separators

- / separates directories and files and // separates the protocol from the site and documents

```
https://www.selsoft.com.tr/egitimler
https://google.com/search?q=java&oq=java+&aqs=chrome…
http://localhost/WAP4.0
```

Uniform Resource Locator - URL - III

- Rarely a user and password info may become part of the URL.

```
<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>
```

selsoft

build better, deliver faster

Request

URI & URL

Ports, Queries and Fragments

Port

- Standard port for HTTP is 80.
- HTTP servers typically listens to the port 80 and browsers always send HTTP requests to that port on the host.
- If the standard port is used it is not required in URL.
- But if a non-standard port is used then it needs to be specified in URL.

```
http://localhost:8080/WAP4.0
```

Query and Request Parameters

- In URL path everything after the `?` is called **query**.
- A query mosty includes information that is submitted to the server of the host so that they are processed.
- They are mostly key value pairs in the form of **`name1=value1&name2=value2`**…
- They are called **request parameters**.

```
https://google.com/search?q=java&oq=java+&aqs=chrome…
http://localhost:8080/WAP4.0/GetFormServlet?firstName=Ali&lastName=Ozen
```

Fragment

- Fragment points to a portion of the resource for example a web page
- It is after the number sign `#`.
- It is used on the client side i.e. by a browser when rendering the resource.
- Browsers tries to show that portion of the page.

```
http://selsoft.com.tr/egitimler/#iletisim
```

selsoft

build better, deliver faster

Request

URL Encoding

URL Encoding - I

- HTTP is designed to be simple and portable so it uses the most basic and safe character set US-ASCII.
- US-ASCII is a 7-bit set and its alphanumeric chacracters are called safe.
  - Safe characters include some other s such as : and /
- For unsafe characters **URL encoding** is used.
- URL encoding represents the unsafe character by escaping, i.e. a percent sign `%` followed by two hexadecimal digits that represent the ASCII code of the character.

URL Encoding - II

```
http://www.öğrenci.com/ara?isim=ayşe
http://www.%C3%B6%C4%9Frenci.com/ara%3Fisim%3Day%C5%9Fe
```

selsoft

build better, deliver faster

Request

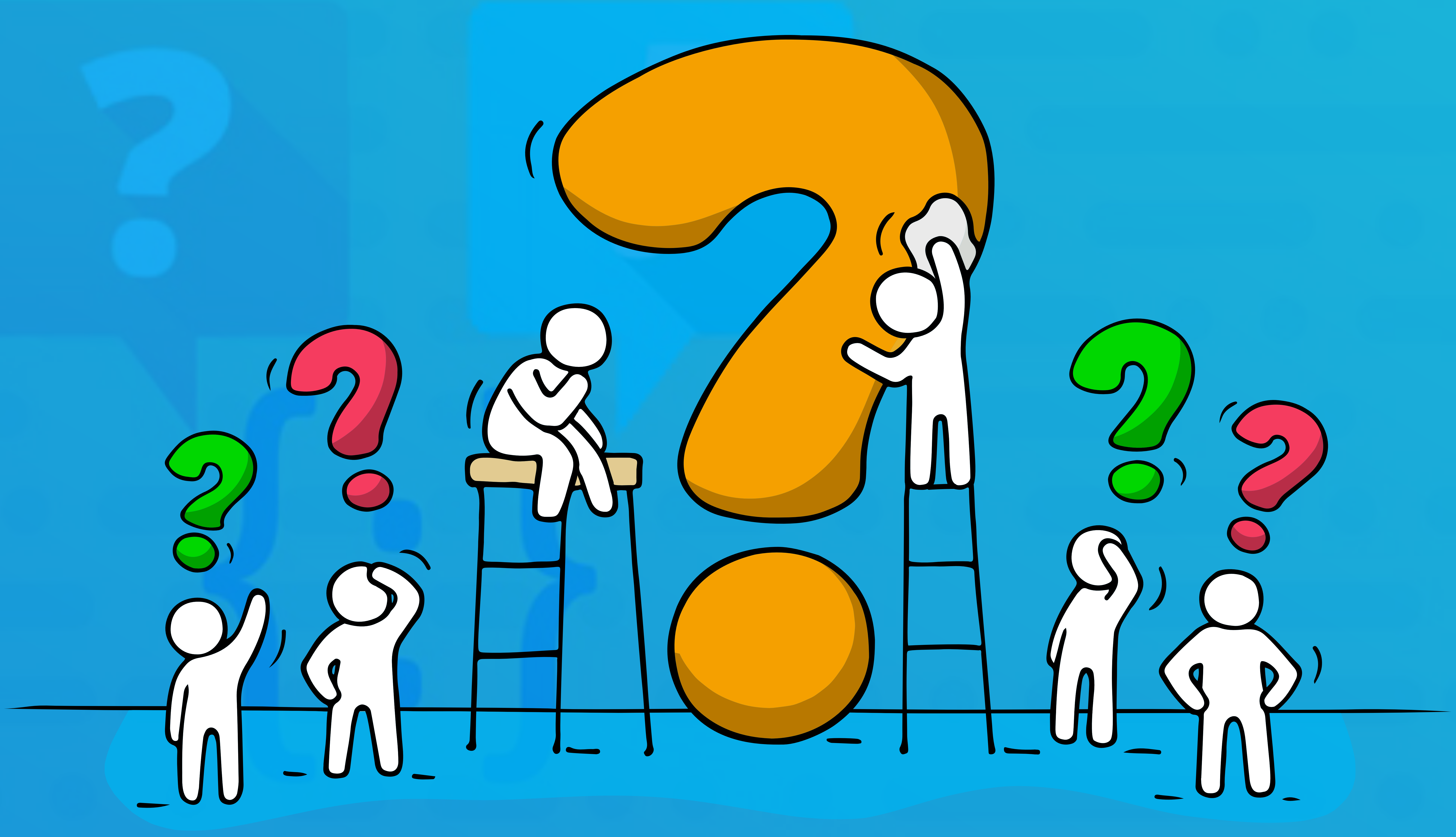
HTTP Version

HTTP Version

- HTTP version part tells the server what dialect of HTTP the client is speaking.
- There are four version of HTTP: 0.9, 1.0, 1.1 and 2.0.
- 3.0 is in the phase of experimentation.
- The most used version is 1.1.

```
GET / HTTP/1.1
POST /shoppingCard.html HTTP/2.0
```

Time for questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

selsoft
build better, deliver faster

Response

Response - I

- HTTP response is very much like request.
- It starts with a status line that consists of HTTP version of the response and a status code, which indicates how successfully the request was serviced
- HTTP version is the version the server is capable of supporting.
- One or more optional headers may follow status line.
- Then comes a blank line.

Response - I

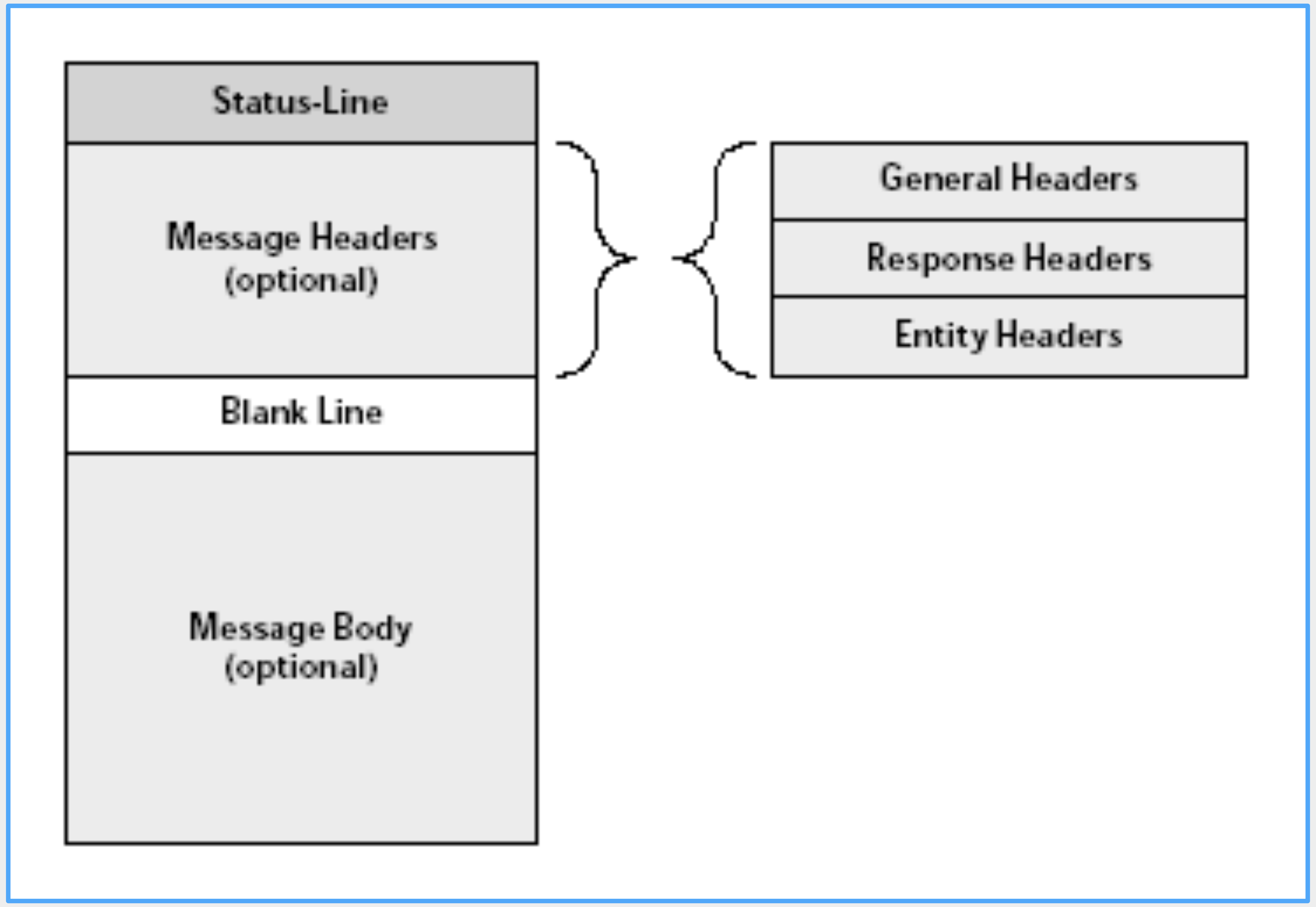
- After the blank line an optional comes mesage body.
- If the request is successful, message body would include the requested resource.
- In case of fail, there would be no body.

Parts of Response - I

- Response has three parts:
  - A status line that consist of following three items:
    - HTTP version
    - Status code
    - Status phrase
  - Headers
  - Entity body as arbitrary data

```
<HTTPversion> <status> <status-phrase>
<headers>
<entity-body>
```

Parts of Response - II



An Example Response

```
GET /WAP4.0/SelamServlet HTTP/1.1
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: JSESSIONID=3a8dfc71ce09c25e6be6abfac03f; JSESSIONID=394967ca90899fe7d12078489d4b; treeForm_tree-
hi=treeForm:tree:applications
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 400 Bad Request
Server: GlassFish Server Open Source Edition  5.1.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition  5.1.0  Java/Oracle
Corporation/1.8)
Date: Wed, 02 Dec 2020 14:57:20 GMT
Connection: close
Content-Length: 0
```

An Example Response

```
GET /WAP4.0/SelamServlet HTTP/2.0
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: JSESSIONID=3a8dfc71ce09c25e6be6abfac03f; JSESSIONID=394967ca90899fe7d12078489d4b; treeForm_tree-
hi=treeForm:tree:applications
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: GlassFish Server Open Source Edition  5.1.0
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition  5.1.0  Java/Oracle Corporation/1.8)
Content-Type: text/html;charset=ISO-8859-1
Connection: close
Content-Length: 295

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML>
<HEAD><TITLE>SelamServlet</TITLE></HEAD>
<BODY>
<h1 align="center">SelamServlet</h1>
<H1>Selam via GET!</H1>
<p><h4><a href="SourceCodeServlet?name=org.javaturk.wap.ch04.SelamServlet">For Source Code</h4></a>
</BODY></HTML>
```

```
HTTP/1.1 200 OK
Server: Sun-ONE-Application-Server/7.0
Date: Fri, 27 Dec 2002 21:16:11 GMT
Content-length: 31259
Content-type: text/html
Last-modified: Wed, 25 Dec 2002 22:55:12 GMT
Accept-ranges: bytes
Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head><title>Sun ONE Application Server 7</title>

<meta http-equiv="content-type" content="text/html; charset=iso-
8859-1">
<meta name="keywords" value="">
<meta name="description" value="">
. . . . .
```

```
HTTP/1.1 400 Bad request
Server: Sun-ONE-Application-Server/7.0
Date: Fri, 27 Dec 2002 21:11:15 GMT
Content-length: 147
Content-type: text/html
Connection: close

<HTML><HEAD><TITLE>Bad request</TITLE></HEAD>
<BODY><H1>Bad request</H1>
Your browser sent a query this server could
not understand.
</BODY></HTML>
```

selsoft

build better, deliver faster

Response

Status Codes

Status Codes - I

- Status codes are important part of the HTTP responses.
- They give short and quick information regarding the response.
- Every status code is a three-digit number.
- There are mainly five categories for the codes.

Status Codes - II

- They are classified on their first digit
  - **1xx** codes are for information, in fact there is only one code, 100
  - **2xx** codes indicate success
  - **3xx** codes redirect client
  - **4xx** codes indicate a client error
  - **5xx** codes are for server problems

Status Codes - III

- All 1xx, 204 (No content) and 304 (Not modified) responses can not include any message body.
- For all 3xx responses, if there is an alternative URI, it is specified in `Location` header of the response.

Example Status Codes

- 100 Continue
- 200 OK
- 201 Created
- 301 Moved Permanently
- 303 Not Modified
- 307 Temporary Redirect
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 500 Internal Server Error
- 501 Non Implemented
- 503 Server Unavailable

selsoft
build better, deliver faster

Characteristics of HTTP

Characteristics of HTTP

- HTTP has the following set of characteristics:
  - Application level
  - Request-response
  - Stateless
  - Bi-directional transfer
  - Capability negotiation
  - Support for caching
  - Support for intermediaries

HTTP is Stateless

- Each HTTP request is self-contained, i.e. each request is independent of all others.
- The server does not keep a history of previous requests and treats each request alone.
- Nor does HTTP have a specific mechanism to relate a request to another.
- In applications web server needs to keep some state information about each of its clients.

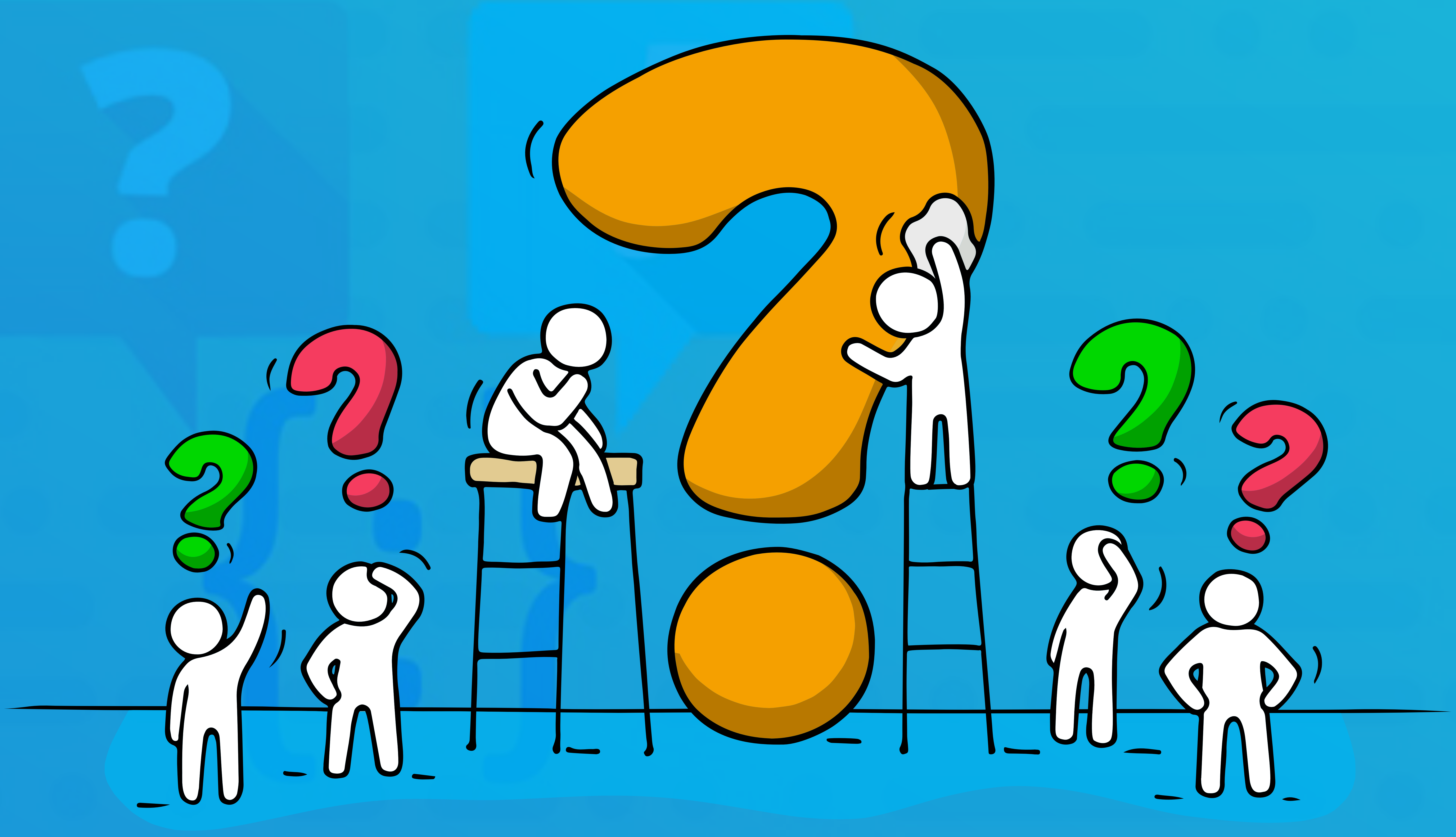
Cookie

- This is called state tracking and ables the server to associate one HTTP request with another.
- The mechanism that HTTP defines for this purpose is known as cookie.

WAP4.0 Ch08

- Chapter 08 of WAP is on session management.

Time for questions!

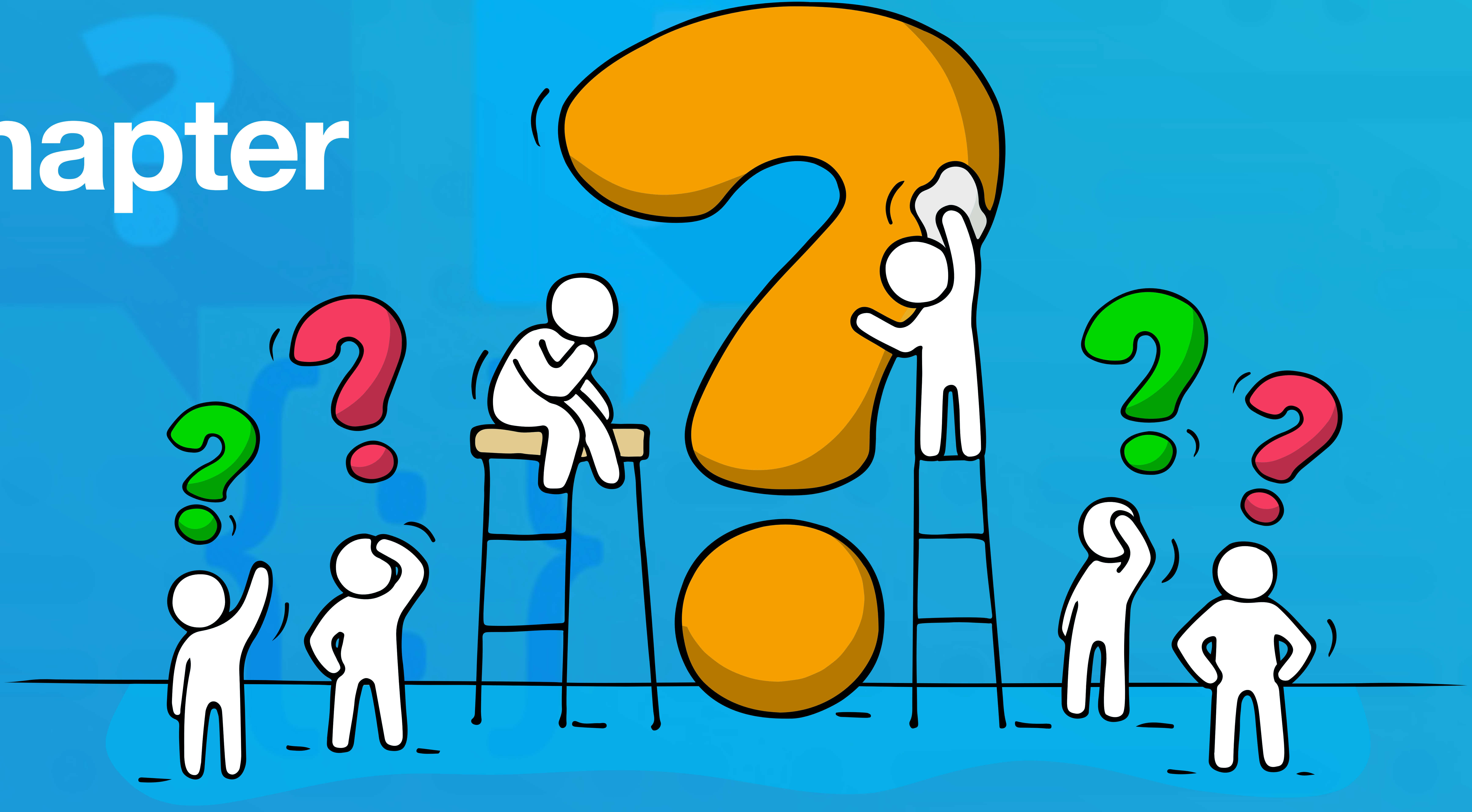


selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

End of Chapter

Time for Questions!



selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr