Introduction, Servlets

Petr Křemen

Knowledge-based and Software Systems, FEE CTU

Winter 2020

Contents

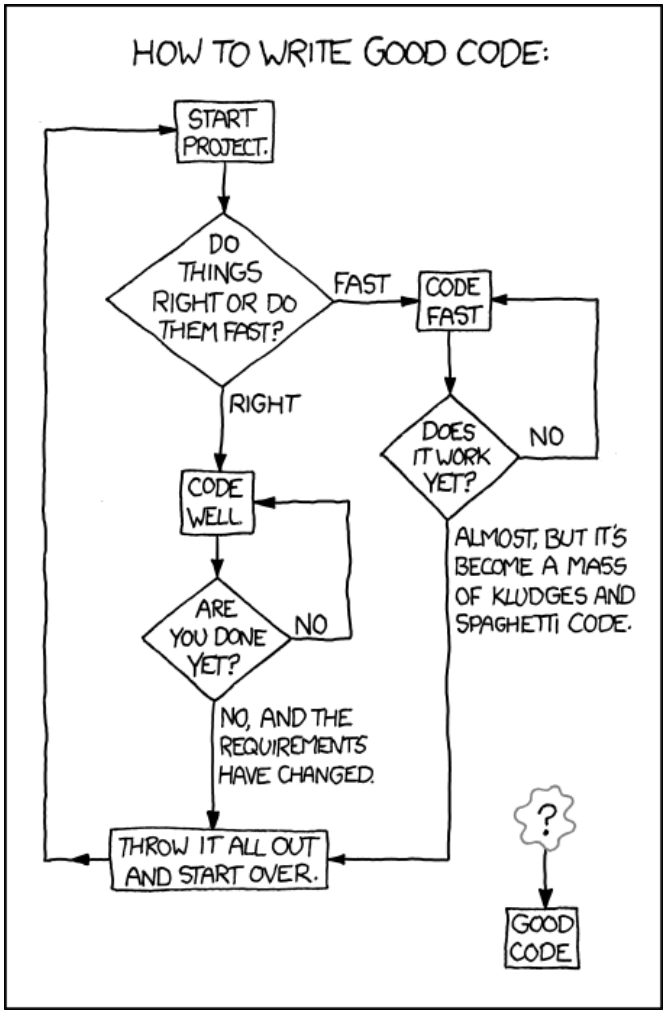
Information About the Course

You Will Learn How to

Design enterprise applications using Java web technologies, including pieces of the Java EE stack

Implement the applications in Java, Spring, EclipseLink

Think about high-availability, clustering, security, and other stuff...



Source: https://techcodegeek.wordpress.com

Teachers

Lecturers:

- Petr Křemen, petr.kremen@fel.cvut.cz
- Petr Aubrecht, aubrecht@asoftware.cz

Course Assistants:

- Martin Ledvinka, martin.ledvinka@fel.cvut.cz
- Petr Aubrecht, aubrecht@asoftware.cz
- Bogdan Kostov, kostobog@fel.cvut.cz
- Lama Saeeda, saeedla1@fel.cvut.cz

Course Organization

- **Go through `https://cw.fel.cvut.cz/wiki/courses/ear` carefully, including subsections:**
  - Lectures

    `https://cw.fel.cvut.cz/wiki/courses/ear/lectures/start`
  - Seminars

    `https://cw.fel.cvut.cz/wiki/courses/ear/tutorials`
  - Assessment

    `https://cw.fel.cvut.cz/wiki/courses/ear/hodnoceni`
  - Materials

    `https://cw.fel.cvut.cz/wiki/courses/ear/materials`
- The course will be split into two parts:

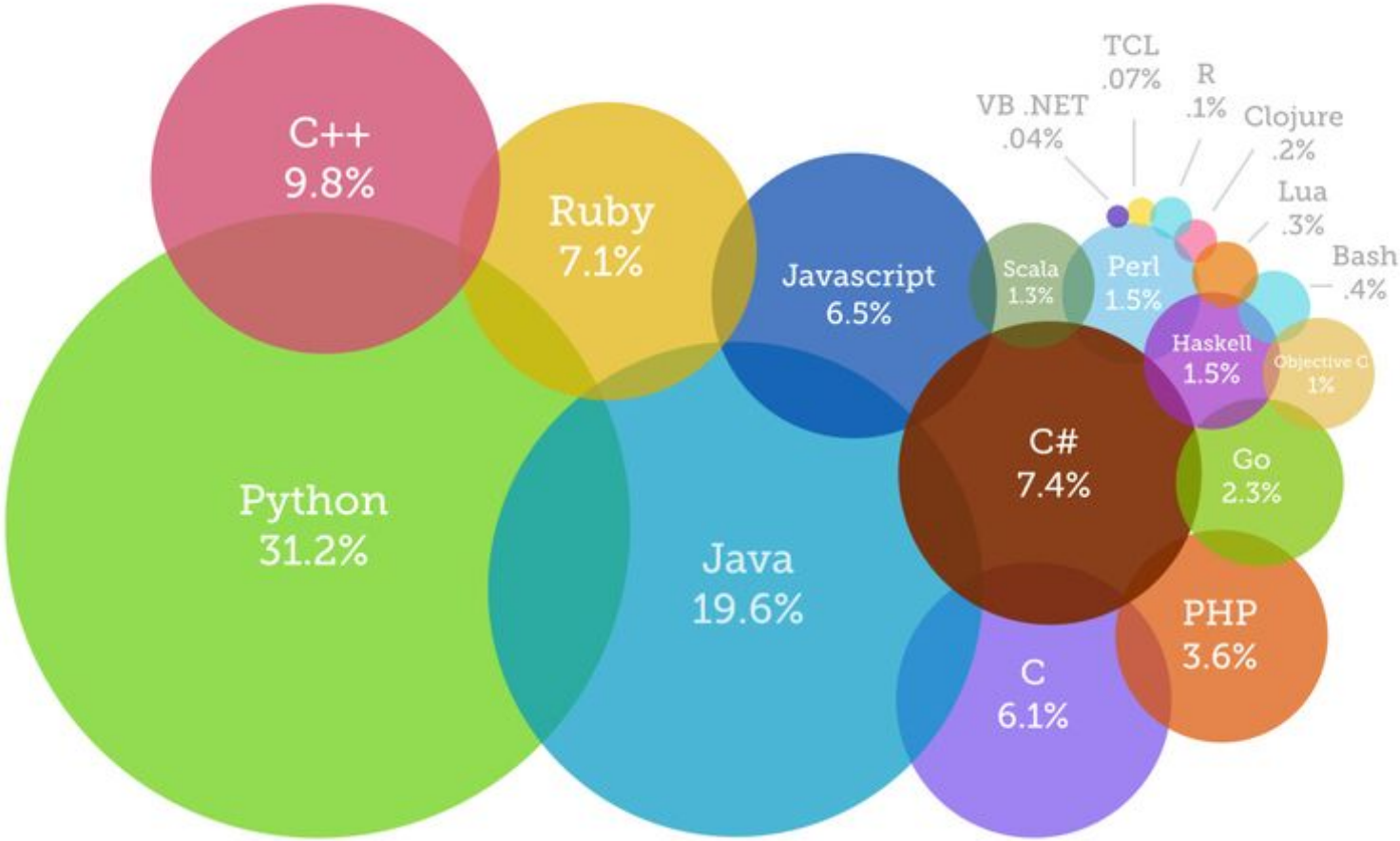
  Basic topics – lectures 1-7

  Advanced topics – lectures 8-13

Enterprise Applications

Why Java?

Answer #1: Usage of programming languages in 2020



Source:

https://www.devsaran.com/blog/10-best-programming-languages-2020-you-should-know/

Why Java?

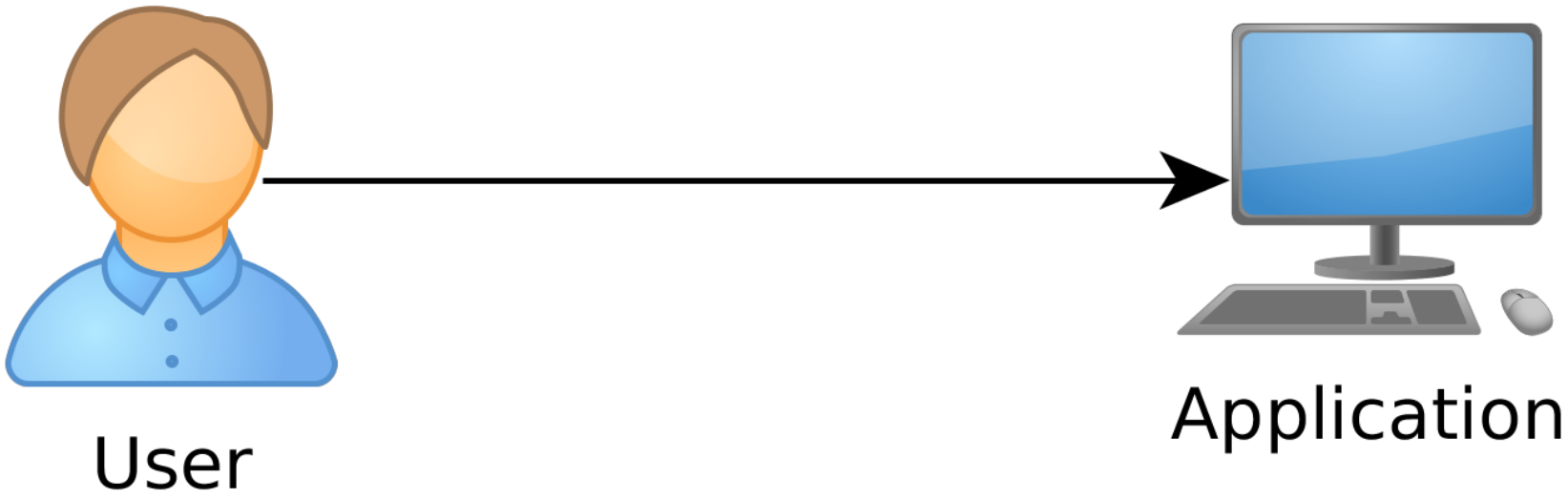
Answer #2:

- Well-established
- Portable (bytecode)
- Optimized in runtime
- Public specifications JSR based on community discussion
- Editions

Java Editions

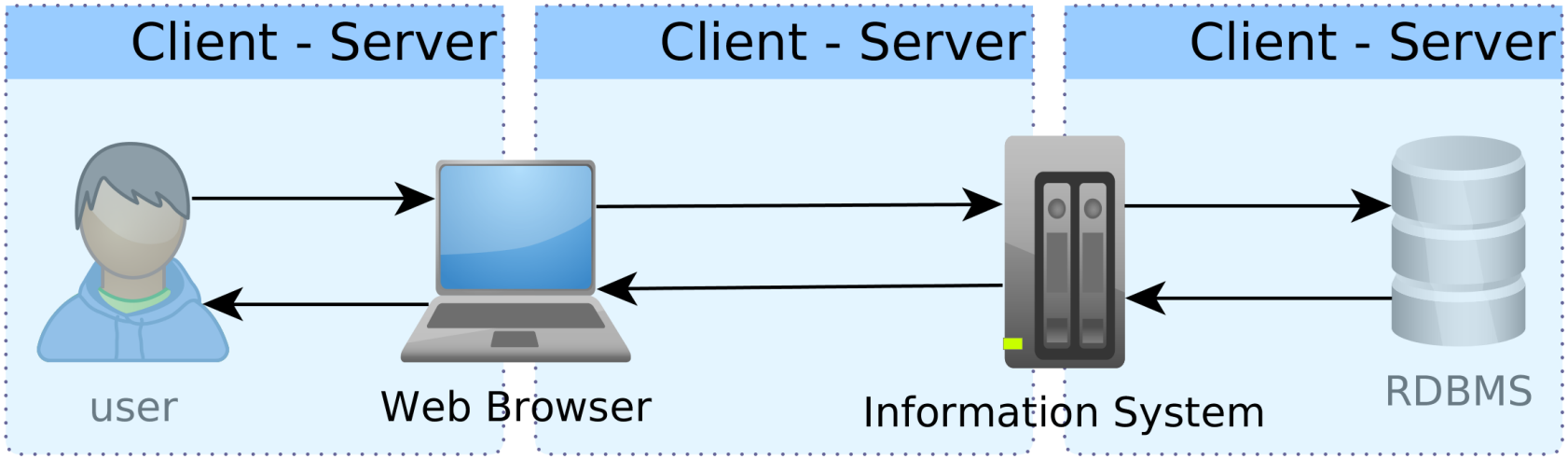
- Java ME – micro edition (Java ME 8.3)
- Java SE – standard edition (Java SE 15)
- Jakarta EE – enterprise edition (Jakarta EE 9)
  - Formerly Java EE, submitted to Eclipse Foundation by Oracle
- (Android), ...

Desktop Application

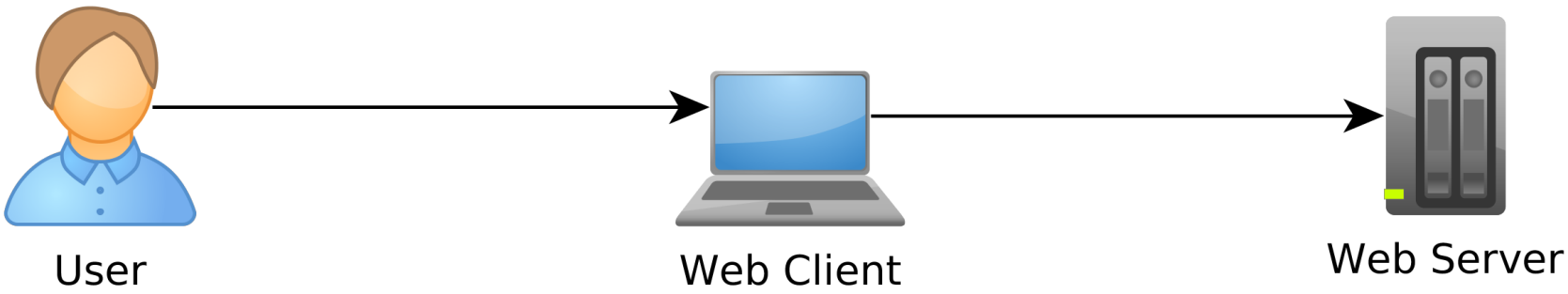


Desktop Application. Single-user access.

Client – Server Pattern

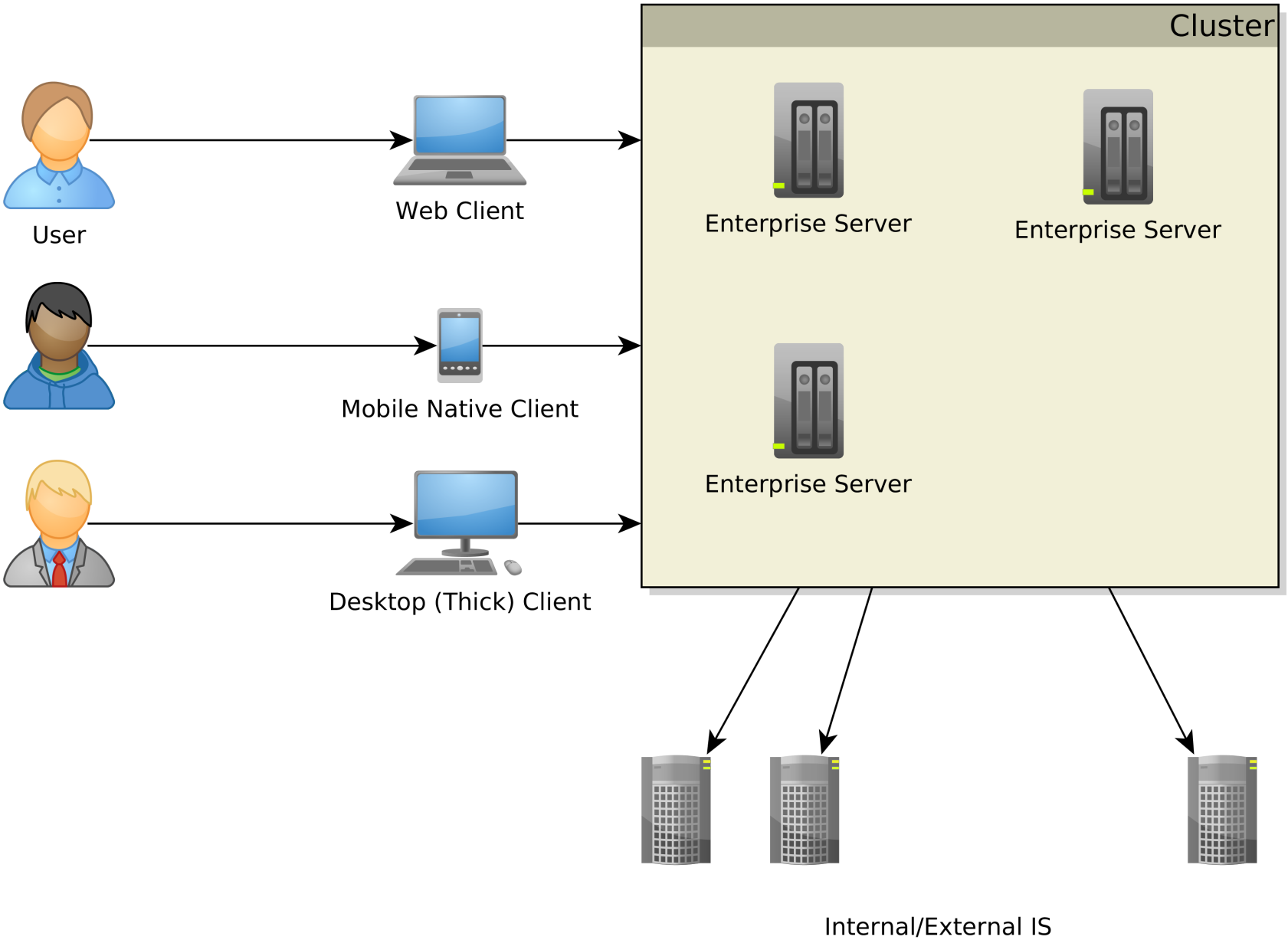


Web Application



Web Application. Multi-user access, single client (web), no integration with other systems.

Enterprise Application (EA)



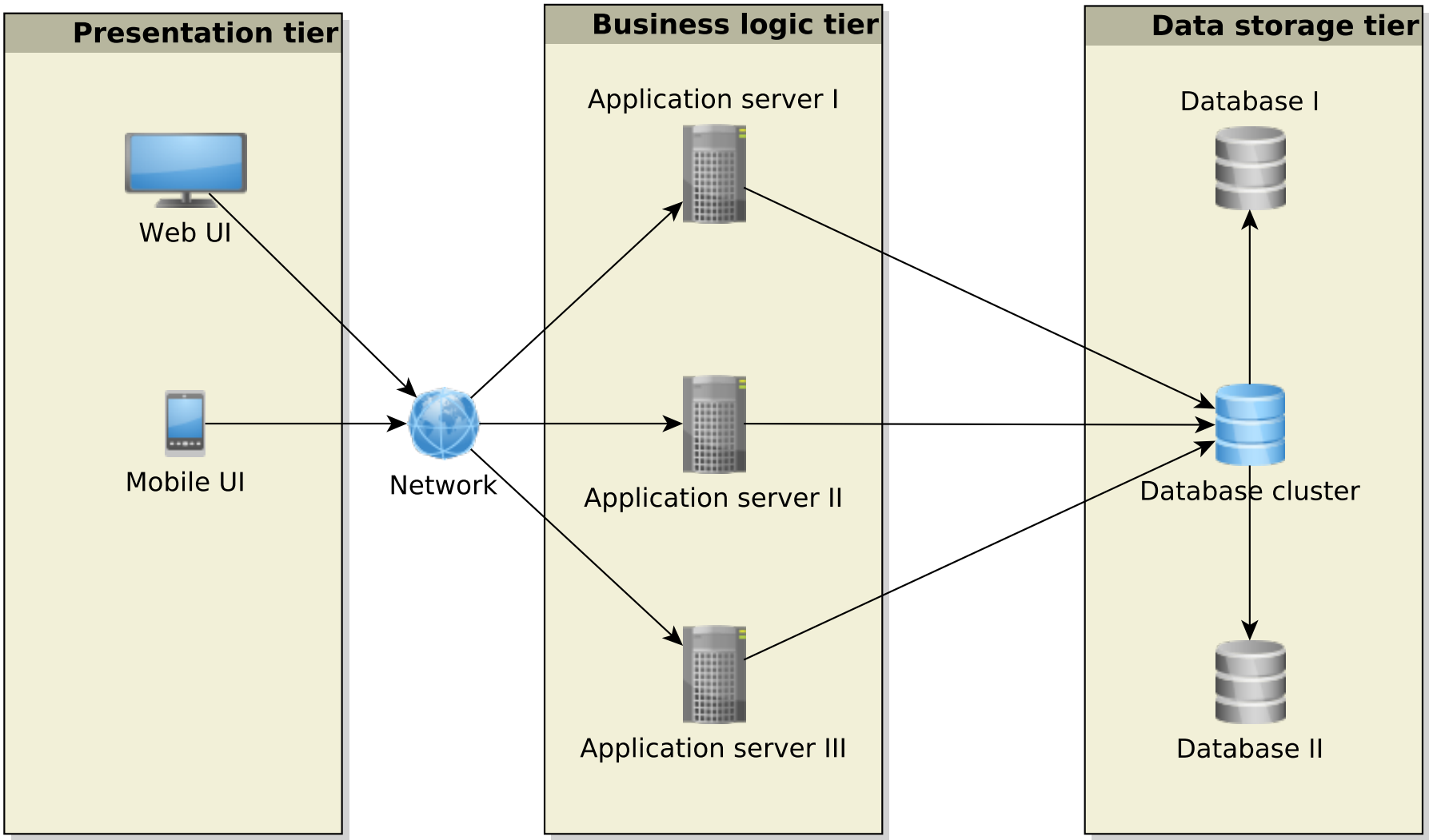
Multi-tier Architecture

Application split into tiers which can be run in separate processes or even on separate machines.
Typically **three-tier**

1. Presentation
2. Business logic
3. Data storage

Unidirectional control flow – top-down.

Multi-tier Architecture



Enterprise Application Architecture

Martin Fowler: Patterns of Enterprise Application Architecture

"... display, manipulation and storage of large amounts of complex data and the support or automation of business processes with that data."

Enterprise Applications – Requirements

Persistent Data using relational databases, graph databases, NoSQL databases, RDF triple stores,

Complex Data Integration of different volume, accuracy, update frequency, quality and meaning $\rightarrow$ data integration,

Concurrent Data Access by many users at once with different scenarios (writing, reading different parts of data),

Multiple Input Interfaces involving complex user interfaces (many forms, web pages), (sensoric) data sources, operational data,

Process Automation involving integration with other enterprise applications, batch processing, etc.

Performance, Robustness involving (horizontal/vertical) scalability, load balancing, high-availability

Data Integration

Enterprise Conceptual Models – produce among others **shared vocabularies (ontologies)** to avoid data ambiguity

Master Data[1] – data spanning the whole enterprise, like *customers*, *products*, *accounts*, *contracts* and *locations*

[1] source: `http://www.ibm.com/developerworks/data/library/techarticle/dm-0804oberhofer`

Why Enterprise Conceptual Models?

9/11 – One or Two Events ?

... matter of billions of USD

Source:https://www.metabunk.org/larry-silversteins-9-11-insurance.t2375

Integration with Other Enterprise Applications

Messaging systems for asynchronous messaging

- Java Message Service (JSR 343)

Remote Procedure Calls for synchronous calls

- RPC
- RMI
- CORBA
- Web Services

Performance Testing[2]

Metrics

Response time – server-side request processing time,

Latency – request processing time perceived by client (response time + network delay),

Throughput – transactions per seconds,

Scalability – sensitivity to resource (hardware) addition/removal,

- **Scaling up (vertical)** – add resource (RAM) to one server
- **Scaling out (horizontal)** – add more servers

Contextual Information

Load – number of requests/transactions

Load sensitivity – sensitivity of a metric w.r.t load

[2] https://nirajrules.wordpress.com/2009/09/17/measuring-performance-response-vs-latency-vs-throughput-vs-load-vs-scalability-vs-stress-vs-robustness

Use Case – External B2C System

Like e-shops, social networks

Characteristics

- Many concurrent users
- Web client
- Relational database with a simple model
- Enterprise data store integration

Use Case – Internal Enterprise System

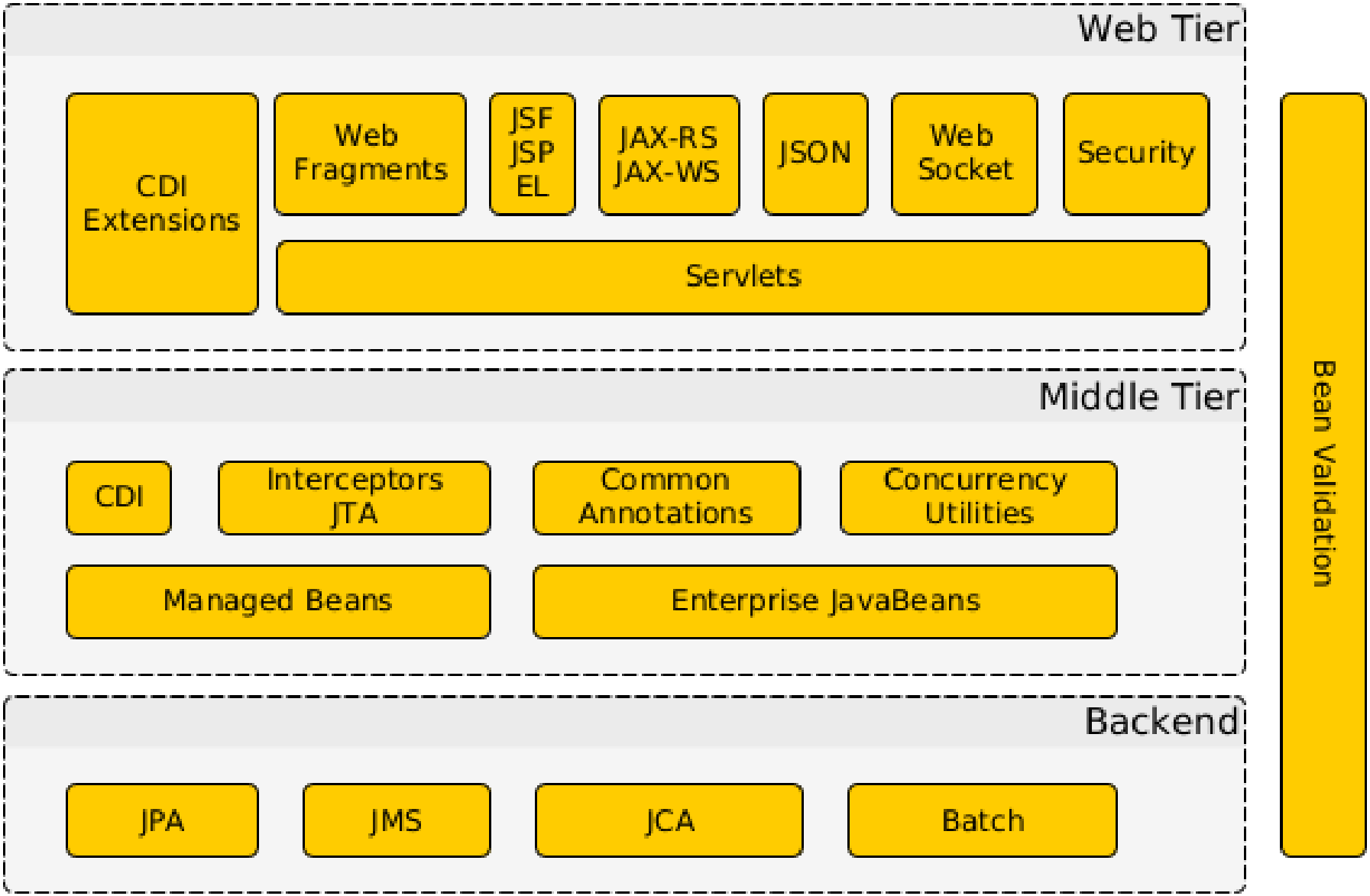
Like Car Insurance System

Characteristics

- (Not so many) Concurrent users – mainly company employees
- Thick client for company employees
- Relational database, complex domain model capturing enterprise know-how
  - E.g., conditions for obtaining an insurance contract
- ERP, CRM integration

Jakarta EE

Jakarta EE = Jakarta Enterprise Edition



Jakarta EE Principles

- Single specification, more implementations
- Bunch of technologies integrated in a single platform

  Application server – full Jakarta EE stack (e.g. Payara, Glassfish, WildFly (RedHat),...)

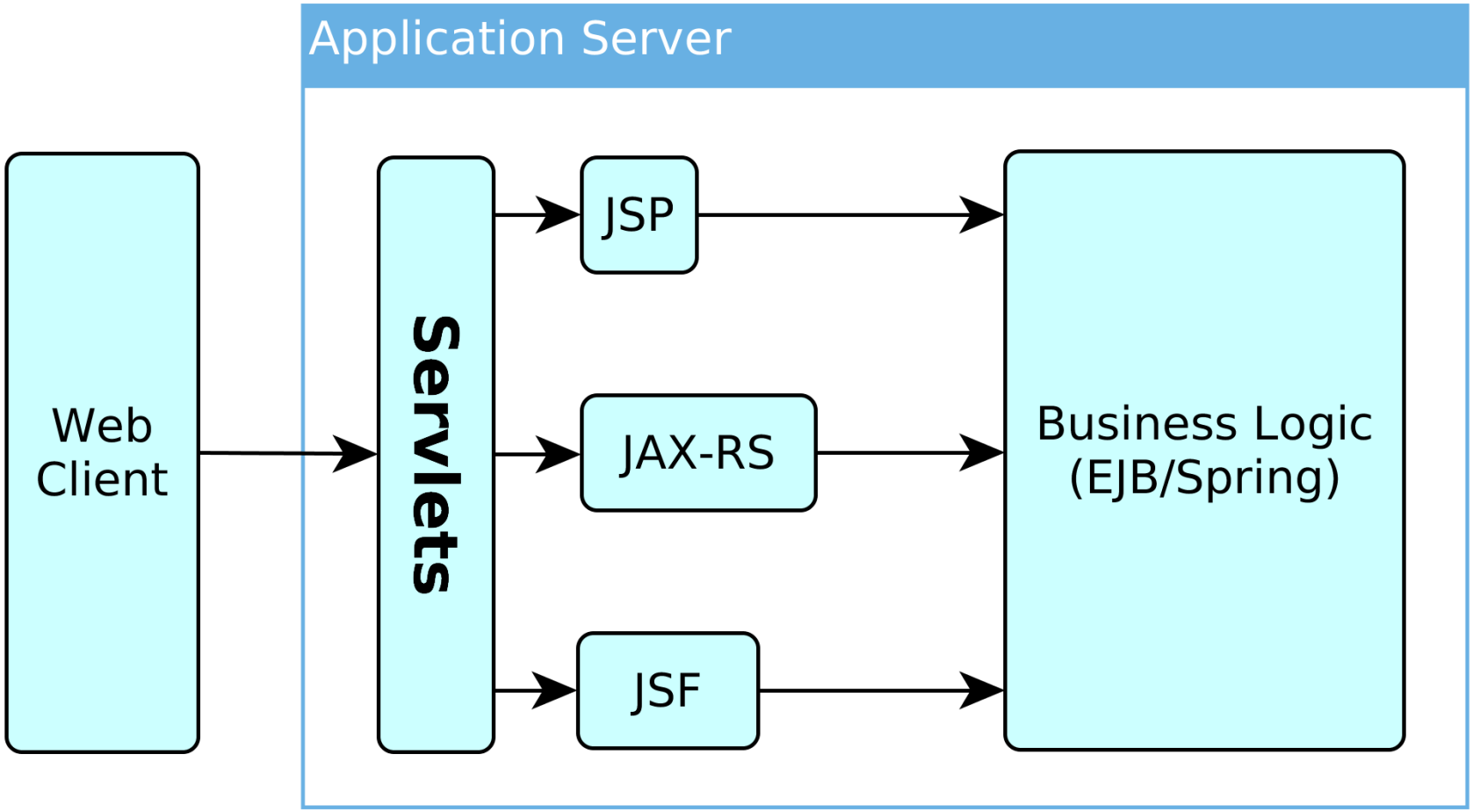
  Web Container – only Jakarta EE web profile (all the above + partially e.g. Apache Tomcat, ...)

Technologies Used in This Course

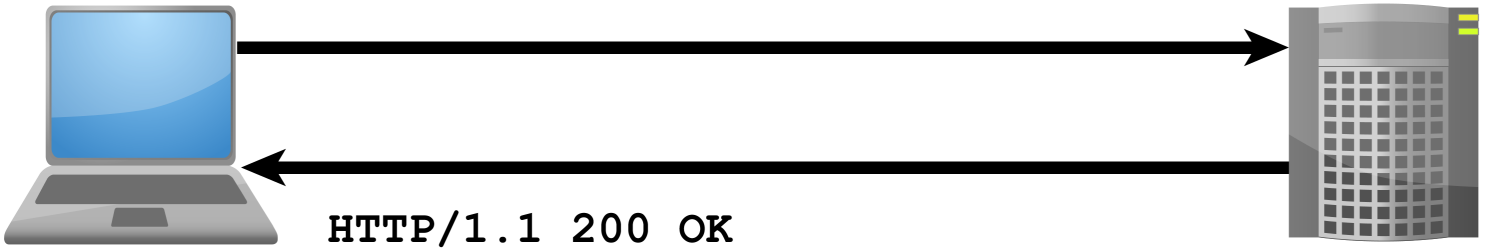
Technology	Java EE	Description
JPA (EclipseLink)	✓	object persistence layer, alternative to Hibernate, OpenJPA, etc.
Spring	×	alternative to Java EE Session Beans, CDI
Spring Web Services	×	web service layer, alternative to JAX-RS
Websockets	✓	client-server bidirectional communication
Servlets	✓	basic HTTP request processing

Servlets

Context


Application Server
Web Client
Servlets
JSP
JAX-RS
JSF
Business Logic (EJB/Spring)

HTTP Protocol




```
GET /redmine HTTP/1.1
Host: kbss.felk.cvut.cz
```

```
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2018 11:53:16 GMT
Server: Apache/2.4.25 (Debian)
Cache-Control: max-age=0, private, must-revalidate
Set-Cookie: _redmine_session=eHdS...
ETag: W/"c21df2f4bbdcdbcb0b68fc26e971e5d6-gzip"
Status: 200 OK
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4489
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

HTTP Response

```
HTTP/1.1 200 OK
Date: Fri, 14 Sep 2018 12:07:38 GMT
Server: Apache
X-Content-Type-Options: nosniff
X-Frame-Options: sameorigin
X-XSS-Protection: 1; mode=block
Referrer-Policy: same-origin
Allow: GET, POST
Access-Control-Allow-Origin: https://www.fel.cvut.cz
Set-Cookie: PHPSESSID=5ccksgfok3f75o08tq9jdt8405; path=/, ;HttpOnly;
    Secure;samesite=strict
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
    check=0
Pragma: no-cache
Set-Cookie: lang=cz; expires=Sun, 14-Oct-2018 12:07:38 GMT; path=/
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

HTTP methods

HTTP 1.0

GET – requests a **representation** of a resource

POST – requests the server to accept the entity enclosed in the request as a **new subordinate** of the web resource identified by the URI

HEAD – same as GET, but **no response body** is expected

HTTP methods II

HTTP 1.1 (rfc2616, rfc5789)

OPTIONS – returns the HTTP methods supported for URL

PUT – requests that the enclosed entity is **stored** under the supplied URI

DELETE – requests deletion of the specified resource

TRACE – echoes the received request (to see the changes made by intermediate servers)

CONNECT – converts the connection to a transparent TCP/IP tunnel (for HTTPs)

PATCH – applies partial modifications to a resource

First Servlet

```
package cz.cvut.kbss.ear.servlet;
import java.io.IOException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet(urlPatterns = {"/hello/*"})
public class HelloWorldServlet extends HttpServlet {

    protected void doGet(HttpServletRequest req,
      HttpServletResponse resp) throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().write("HELLO");
    }
}
```

https://gitlab.fel.cvut.cz/ear/servlet-demo

Servlet

- Java runtime is running (no need to run it)
- Thread pool for request processing
- Memory sharing
- JSESSIONID in cookies
- Local/remote debugging
- Might be a singleton or not

Servlet Container Ensures

- TCP/IP connection
- HTTP protocol processing
- Parameter processing
- Resource management (thread pools)

General servlets are in `javax.servlet.*` package, but we will deal with HTTP servlets (`javax.servlet.http.*` package)

GET vs. POST

Often processed the same way ...

```
public class AServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request,
   HttpServletResponse response)
 { processRequest(request, response); }

 public void doPost(HttpServletRequest request,
   HttpServletResponse response)
 { processRequest(request, response); }

 public void processRequest(HttpServletRequest
   request, HttpServletResponse response)
 { _processRequest(request, response); }
}
```

web.xml

```
@WebServlet(urlPatterns = {"/hello/*"})
public class HelloWorldServlet extends HttpServlet {
...
```

Can be alternatively expressed in `web.xml` as

```
<servlet>
   <servlet-name>HelloWorldServlet</servlet-name>
   <servletclass>cz.cvut.kbss.ear.
   servlet.HelloWorldServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>HelloWorldServlet</servlet-name>
   <url-pattern>/hello/*</url-pattern>
</servlet-mapping>
```

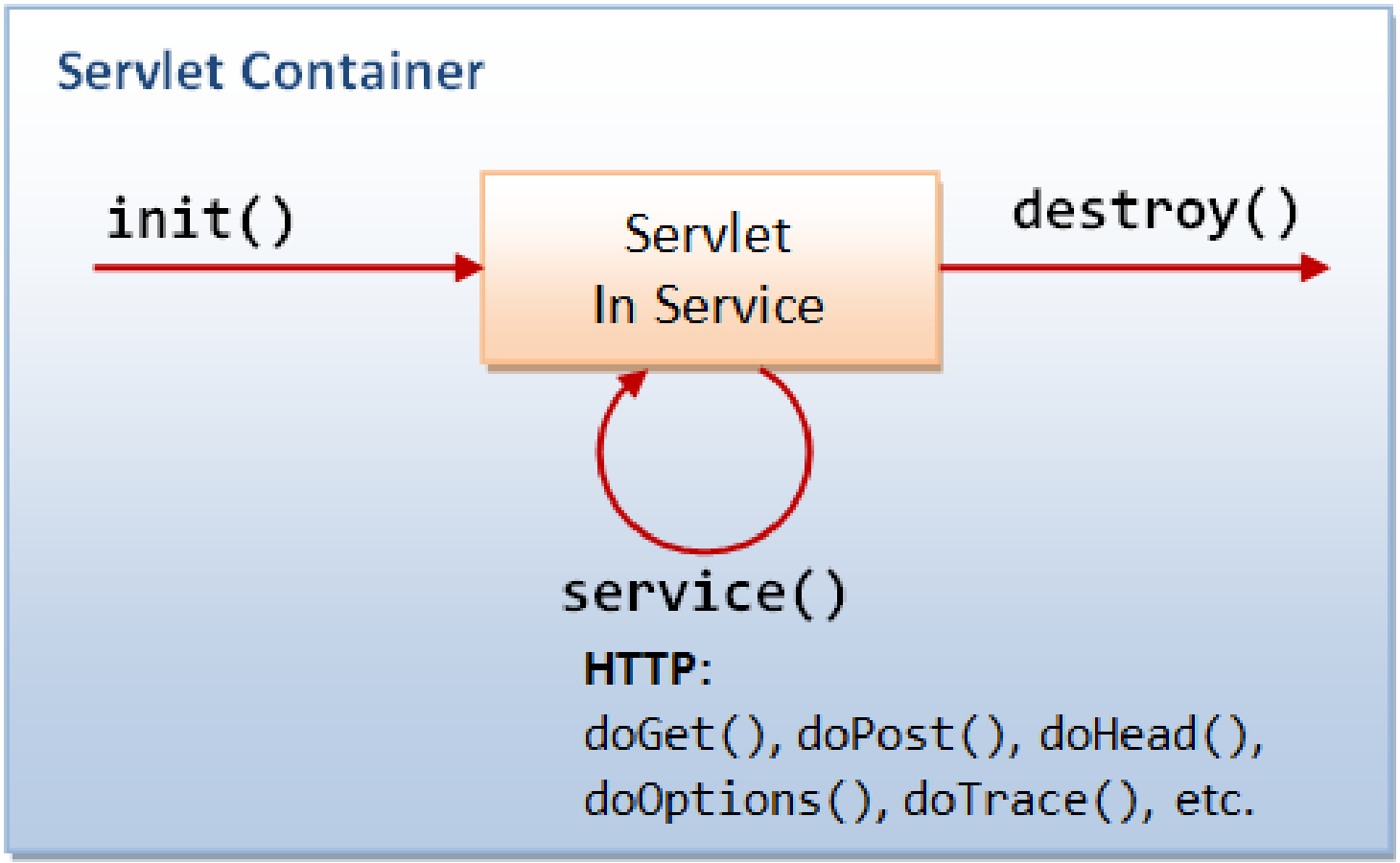
XMLs are an "old-style" solution, but they can (sometimes) do more than annotations (e.g. error-page configuration). They *override annotations*.

Init parameters

```
java.lang.Object
    javax.servlet.GenericServlet
        javax.servlet.http.HttpServlet
```

```
public class HelloWorldServlet extends HttpServlet {
   public void init(ServletConfig config) throws
     ServletException {
       super.init(config);
       System.out.println("Created by " +
         getInitParameter("brand"));
   }
   public void destroy(){
       super.destroy();
       System.out.println("Closing down.");
   }
   ...
}
```

Servlet Lifecycle



Source: `http://idlebrains.org/tutorials/java-tutorials/servlets-init-service-destroy/`

How to share data between requests ?

- Application-wide – `request.getServletContext()`
- Session-wide – `request.getSession()`
- Request-wide – `request`

example

```
String product_id =
  request.getParameter("product_id");
User login = (User)
  request.getSession().getAttribute("currentuser");
```

Client Session State

HTTP is stateless and the session state might be large... Web client can store the session using

URL parameters – but the URL length is limited, problems with bookmarking, parameters shown to the user

Hidden input fields – not shown to the user

Cookies – might be banned by the client; cookies might become mixed up when two apps from the same domain use the same cookie ...

HTTP Cookies

```
GET/index.html HTTP/1.0
Host: www.example.org
```

request

response

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT
...
```

request

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

Client Session State

- State management on the client helps in clustering (stateless failover)
- Should be encrypted for sensitive data $\rightarrow$ extra time
- Server should check the incoming data to ensure their consistency

Server Session State

- The client state contains only server session identifier – `JSESSIONID`. **Beware of Session stealing – the user modifies session ID in order to get someone else's session.**
- The server session is represented either as a BLOB (binary object) or as a text (e.g. XML)

  Locally – in the application server (AS) memory, in AS filesystem, in AS local DB. *Failover problematic*.

  Unstructured shared DB – in a database as BLOBs, session expiration needs to be handled

  Structured shared DB – in a database tables (e.g. session ID column)

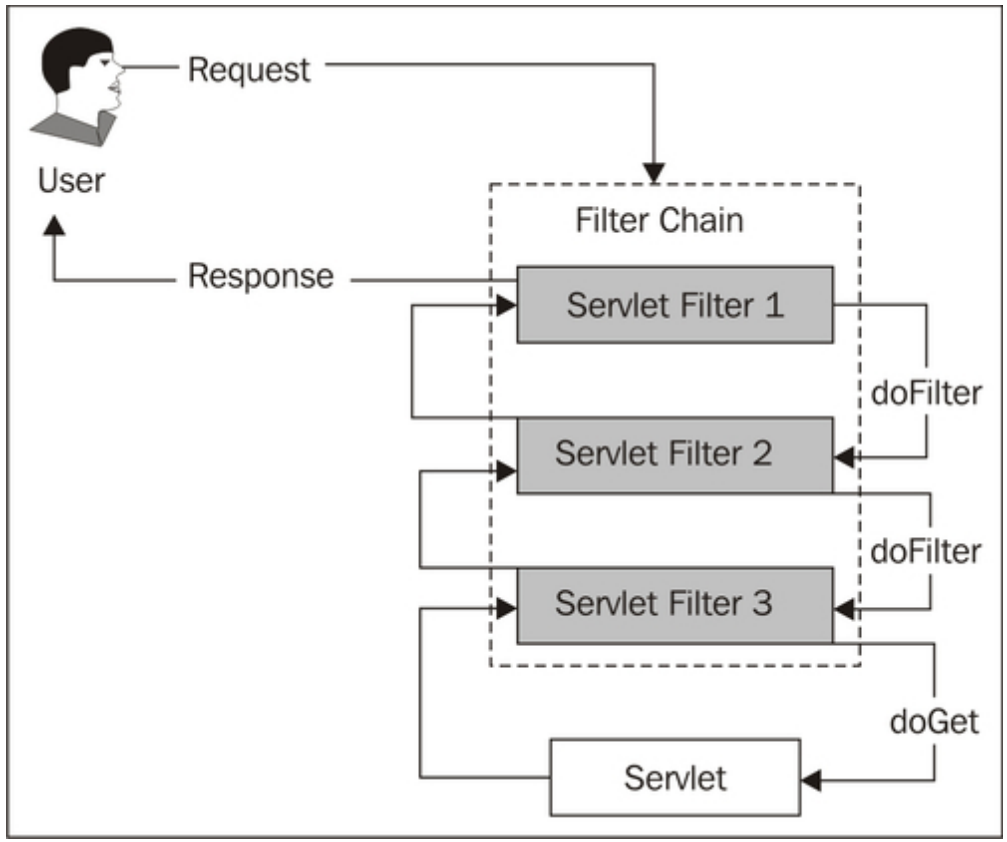
Connection Info

The `HttpServletRequest` offers a lot of information about the HTTP connection

- Client
  - `request.getRemoteAddr()`
  - `request.getRemoteHost()`
- Server
  - `request.getServerName()`
  - `request.getServerPort()`
  - `request.getContextPath()`

E.g., for authorization (`isSecure`, `isUserInRole`, `getAuthType`, `getCookies`, `getHeaderNames`).

Filter Chains



Source:
https://www.packtpub.com/mapt/book/application_development/9781847199744/2/ch02lvl1sec15/security-is-complicated--the-architecture-of-secured-

First Filter

```
package cz.cvut.kbss.ear.servlet;
import java.io.IOException;
import javax.servlet.*;

@WebFilter(filterName = "Only localhost requests")
public class LocalhostFilter implements Filter {
    public void doFilter(ServletRequest req,
      ServletResponse res, FilterChain ch) throws
      IOException, ServletException {
       final String addr = req.getLocalAddr();
       if (addr.matches("127.0.0.1")) {
          ch.doFilter(req, res);
       }
    }
}
```

Filter Logic

```
public class FilterTemplate implements Filter {

    public void init(FilterConfig cfg) { ... }
    public void doFilter(ServletRequest req,
      ServletResponse res, FilterChain ch) throws
      IOException, ServletException {

        // actions before servlet processing

        ch.doFilter(req, res);

        // actions after servlet processing
    }
    public void destroy { ... }
}
```

What can be filters good for?

- Authentication
- Logging and auditing
- Image conversion
- Data compression
- Encryption
- Tokenizing
- Resource access events
- XSL/T
- Mime-type chain

Servlets 4.0

HTTP/2 Support

- Client requests an HTML file `page.html`
- Server finds out that `page.html` links other resources, say `page.css` and `page.js`
- Server pushes `page.css` and `page.js` to the client
- Server responds with `page.html` and closes the request

```
PushBuilder pb = req.newPushBuilder();
pb.path("/page.css");
pb.path("/page.js");
pb.push();
```

HttpServletMapping

- Checking the pattern matched upon request

Summary

Summary

Don't forget!

- Servlets provide an API for HTTP processing
- Many other Java EE technologies are based on servlets

And the next week?

- Enterprise application architectures
- Design patterns

THANK YOU