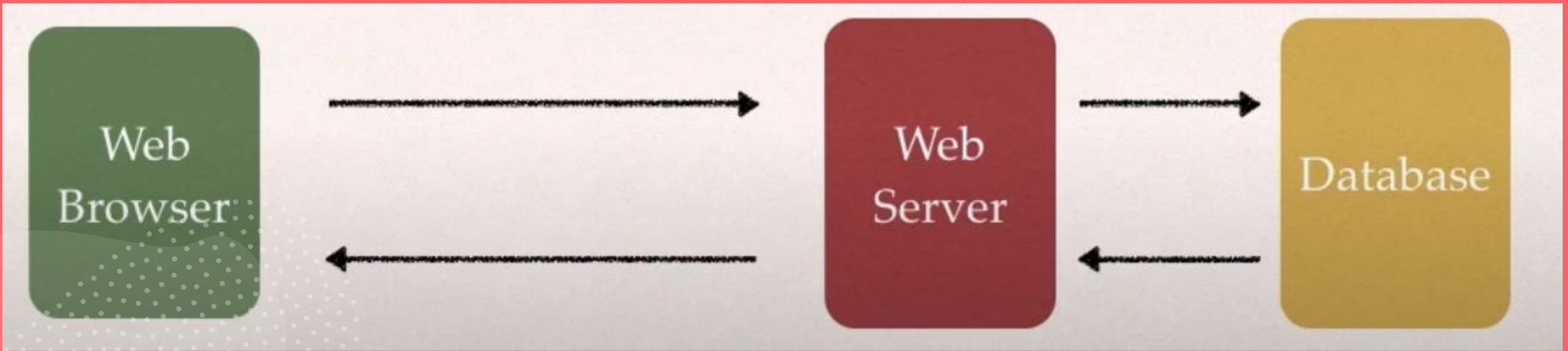
Webbutveckling i Java

JSP

What are Web Applications?

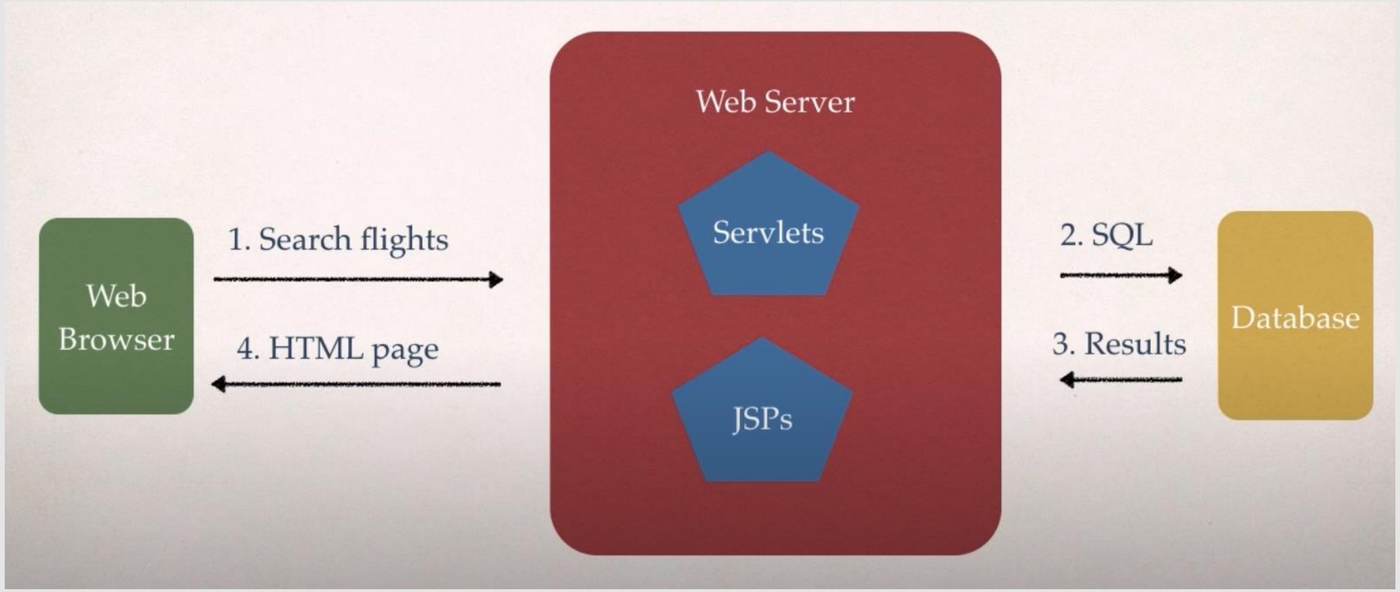
- A web site where the HTML pages are dynamically generated based on the user's actions.



What are JSPs and Servlets?

- Java Server Pages (JSP) is a technology which is used to develop web pages by inserting *Java* code into the HTML pages by making special JSP tags.
- The JSP tags which allow java code to be included into it are `<% ----java code----%>`.
- It can be used as HTML page, which can be used in forms and registration pages with the dynamic content into it.

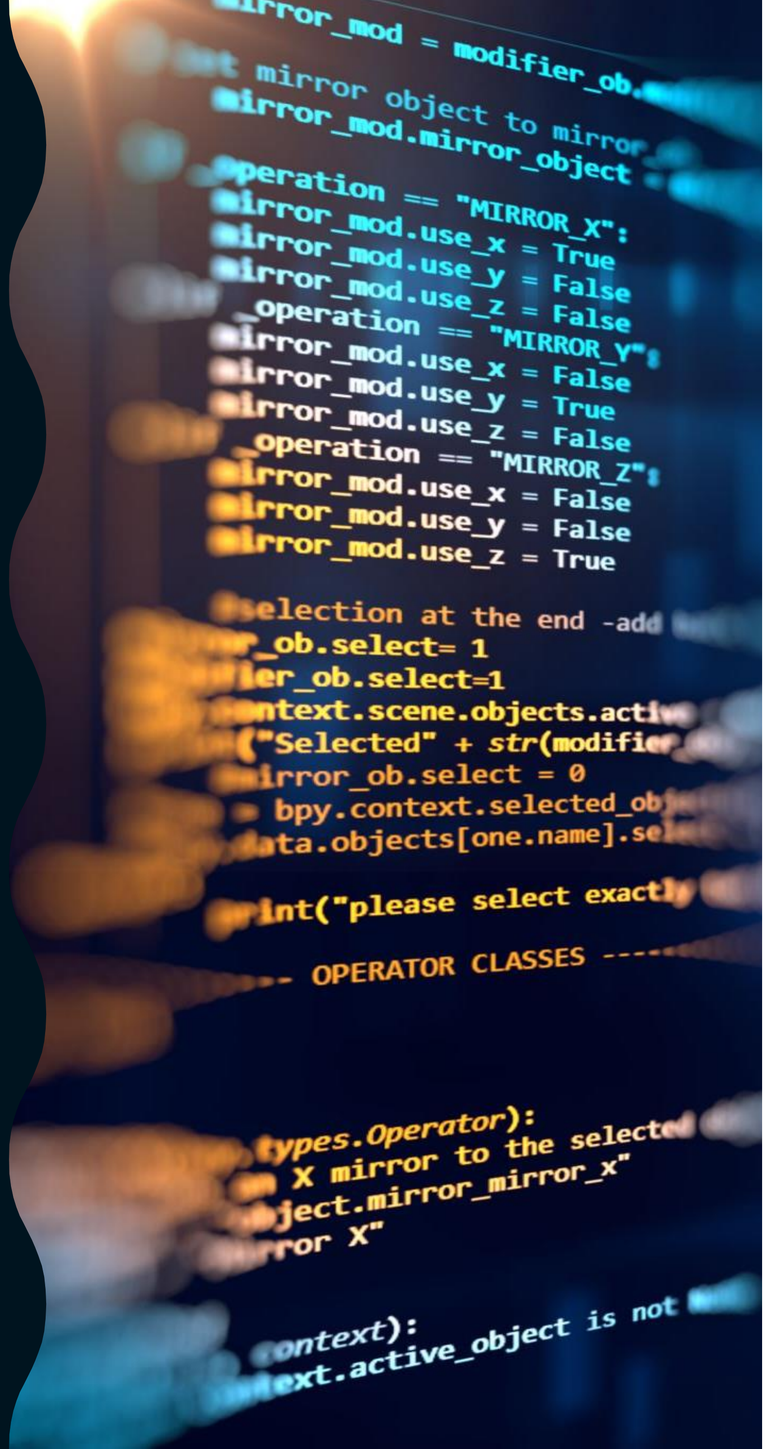
What are JSPs and Servlets?



- Java code that runs on the web server (tomcat)
- Reads user’s action (from for instance a form)
- Performs work
- Returns a dynamically generated HTML page

What types of apps can you create?

- Any. ☺
- E-commerce
- Student / Employee tracking
- Restaurant / Hotel Reservations
- Social Media

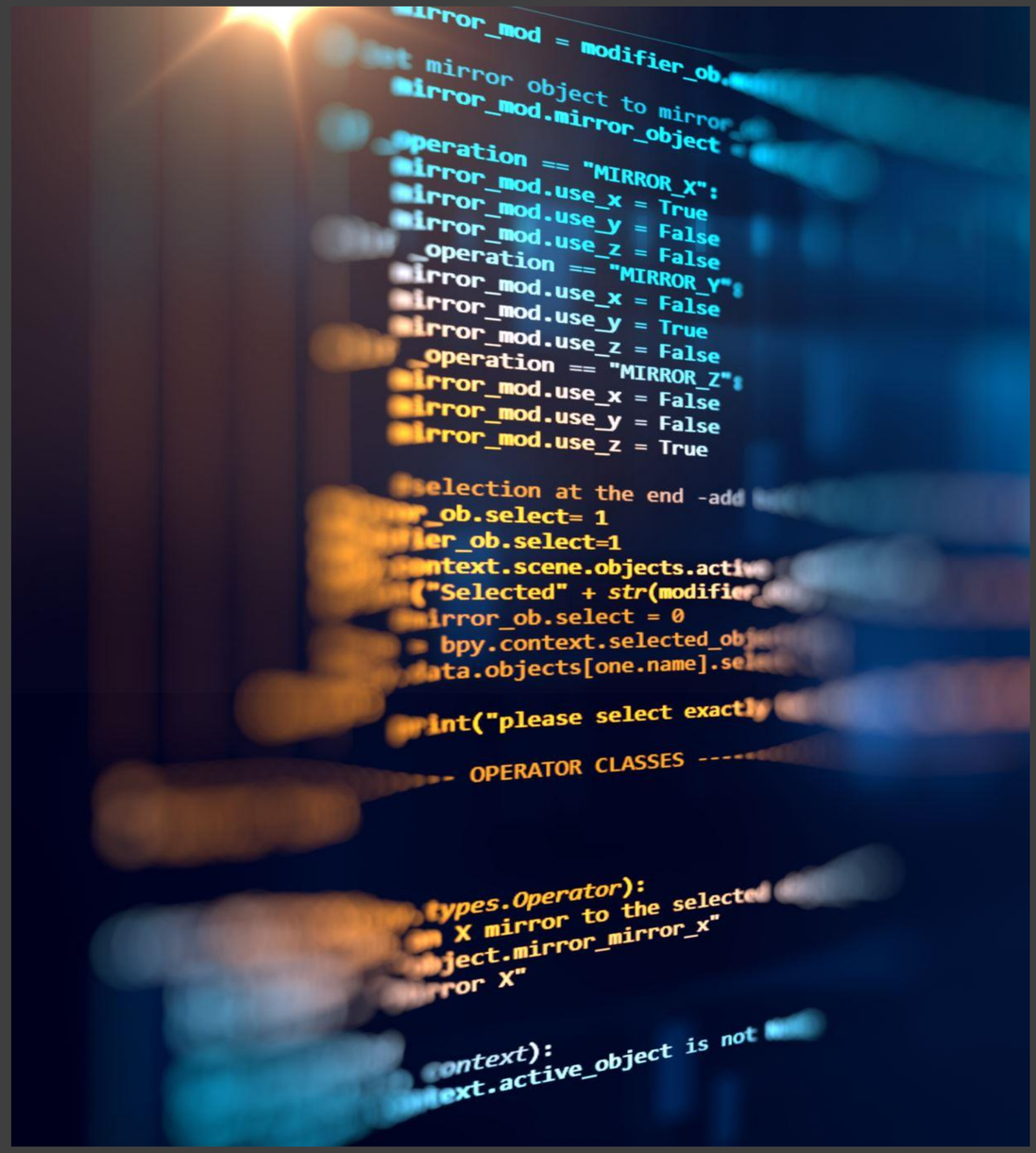


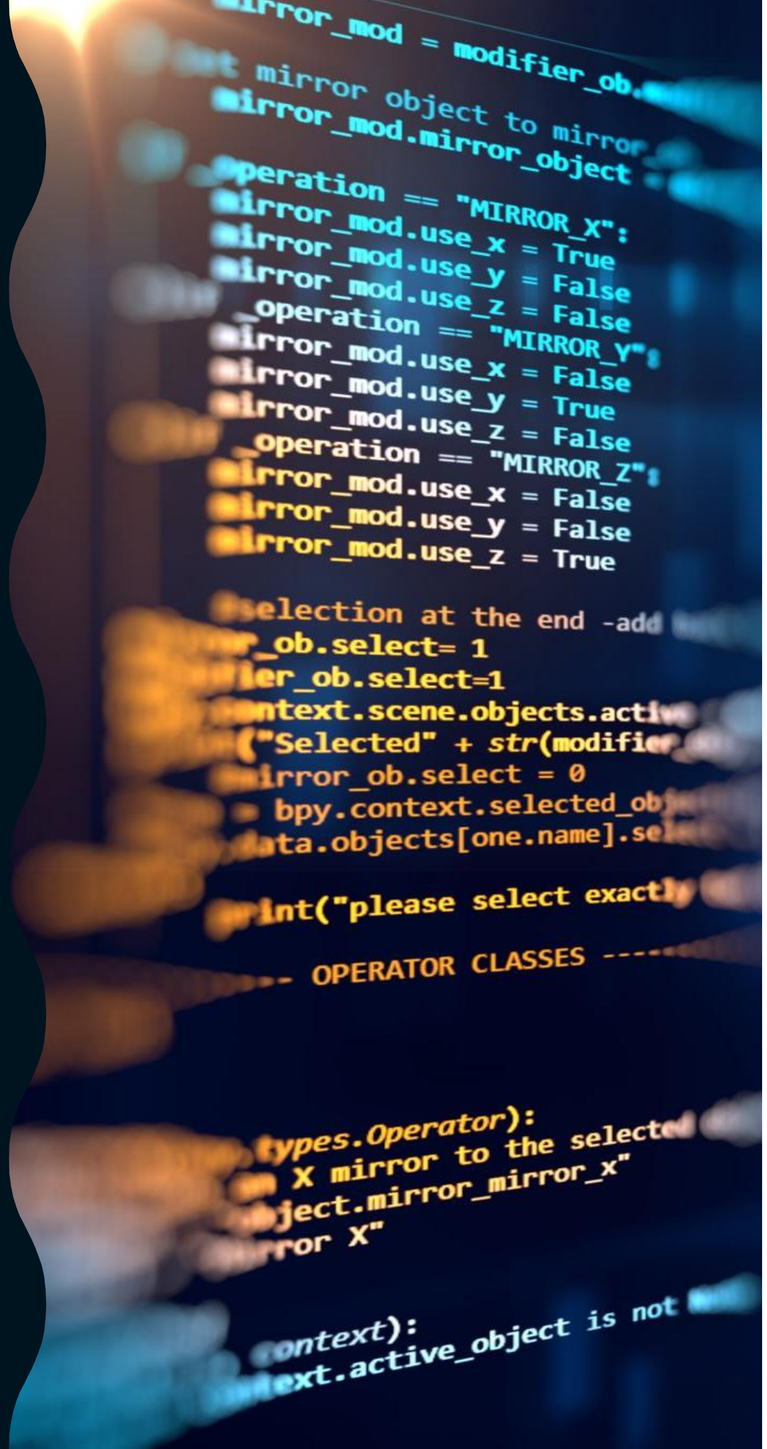
Why use JSP?

- In Java server pages JSP, the execution is much faster compared to other dynamic languages.
- Java server pages JSP are always compiled before its processed by the server as it reduces the effort of the server to create process.

Why use JSP?

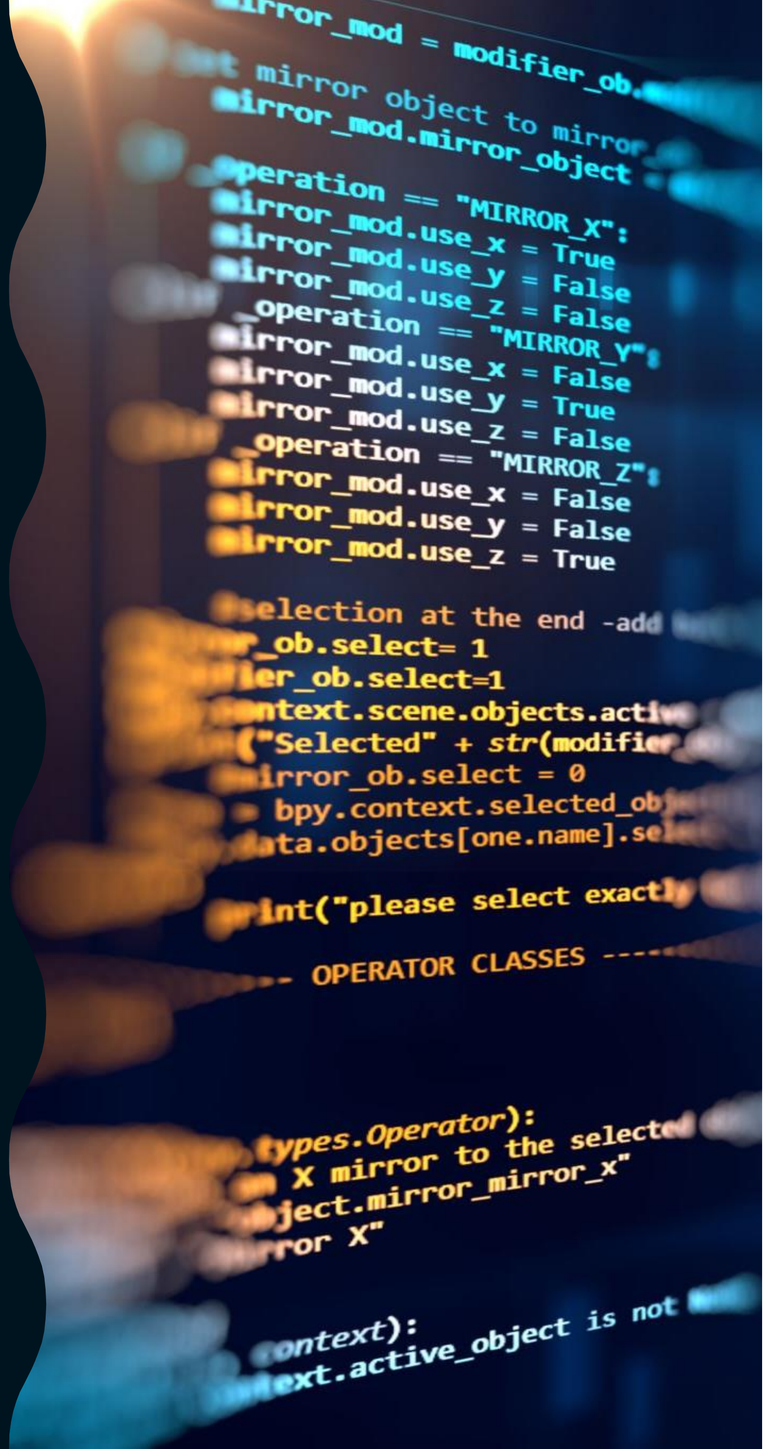
- Java server pages JSP are built over Java Servlets API. Hence, it has access to all Java APIs, even it has access to JNDI, JDBC EJB and other components of java.
- JSP are used in MVC architecture as view layer.
- The request is processed by a view layer which is JSP and then to servlet layer which is java servlet and then finally to a model layer class which interacts with the database.
- JSP is an important part of Java EE, which is a platform for enterprise level applications.





Advantages of using JSP

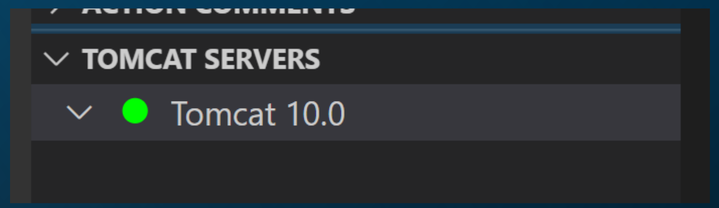
- The advantage of JSP is that the programming language used is JAVA, which is a dynamic language and easily portable to other operating systems.
- It is very much convenient to modify the regular HTML. We can write the servlet code into the JSP.
- It is only intended for simple inclusions which can use form data and make connections.
- JSP can also include the database connections into it. It can contain all type of java objects.
- It is very easy to maintain



Advantages of using JSP

- Performance and scalability of JSP are very good because JSP allows embedding of dynamic elements in HTML pages.
- As it is built on Java technology, hence it is platform independent and not depending on any operating systems.
- Also, it includes the feature of multithreading of java into it.
- We can also make use of exception handling of java into JSP.
- It enables to separate presentation layer with the business logic layer in the web application.
- It is easy for developers to show as well as process the information.

Setting up VS Code - Tomcat for Java



- Add Tomcat Server from Tomcat Install Path
- Start/Restart Tomcat Server from VSCode
- Run war package on Tomcat Server
- Debug war package on Tomcat Server
- Run exploded war on Tomcat Server
- Debug exploded war on Tomcat Server
- Open server homepage in browser to check all deployed war packages
- View all deployed war packages in Tomcat Explorer
- Open war package homepage in browser
- Stop Tomcat Server
- Rename Tomcat Server
- Customize JVM Options when starting Tomcat Server
- Reveal deployed war packages in file explorer
- Delete deployed war package
  - https://marketplace.visualstudio.com/items?itemName=adashen.vscode-tomcat
  - https://github.com/adashen/vscode-tomcat

Hello World!

```
<html>

<head>
    <title>Demo JSP</title>
</head>

<body>
    Time on server is: <%= new
java.util.Date() %>

</body>

</html>
```

JSP Expressions

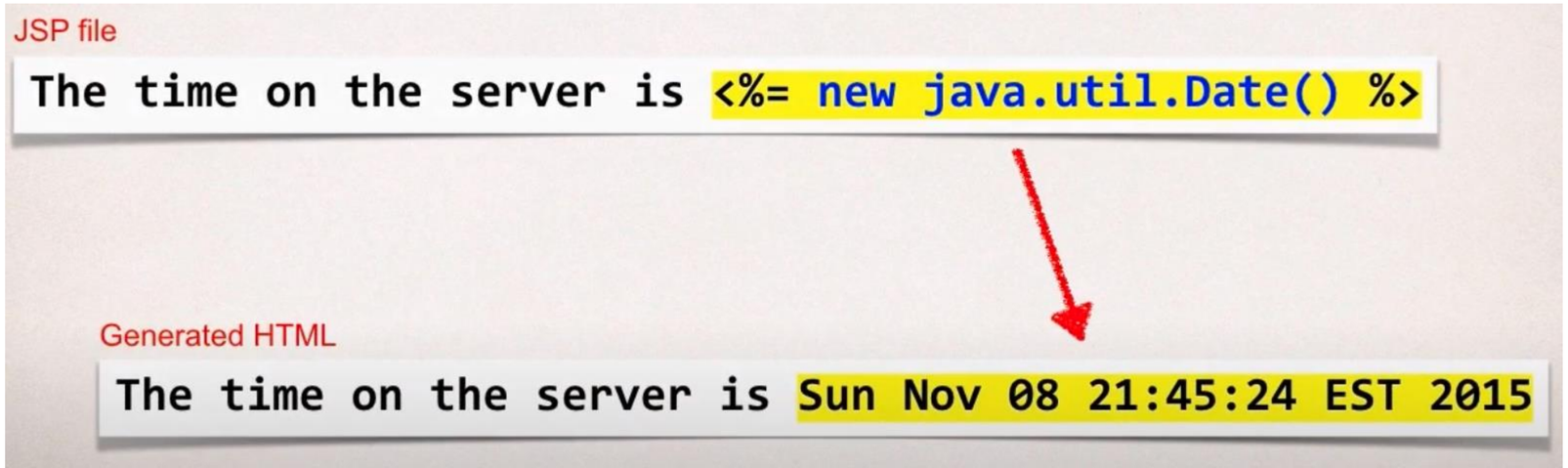
JSP Scripting Elements

Element	Syntax
JSP Expression	`<%= some Java expression %>`
JSP Scriptlet	`<% some Java code: 1 to many lines %>`
JSP Declaration	`<%! variable or method declaration %>`

JSP Expression

- Compute an expression
- Result is included (printed) in the HTML returned to the browser

```
<%= some Java expression %>
```



JSP Expression – More examples

```
Converting a string to uppercase: <%= new String("Hello World").toUpperCase() %>

<br/><br/>

25 multiplied by 4 equals: <%= 25*4 %>

<br/><br/>

Is 75 less than 69? <%= 75 < 69 %>
```

```
<%
   // some lines of Java code
%>
```

JSP Scriptlets

- Insert 1 to many lines of Java code
- To include content in the page use `out.println(…)`

JSP Scriptlet - Example

```
<h3>Hello world!</h3>

    <% for (int i=1; i <= 5; i++) {
out.println("Line " + i + " <br>");

        }

        %>
```

JSP Scriptlet – Best practice

- Minimize the amount of code in a JSP
- Refactor code into a separate Java class

```
<%!
    // declare a method
%>
```

JSP Declarations

- Declare a method in the JSP page
- Call the method in the same JSP page

JSP Declaration - Example

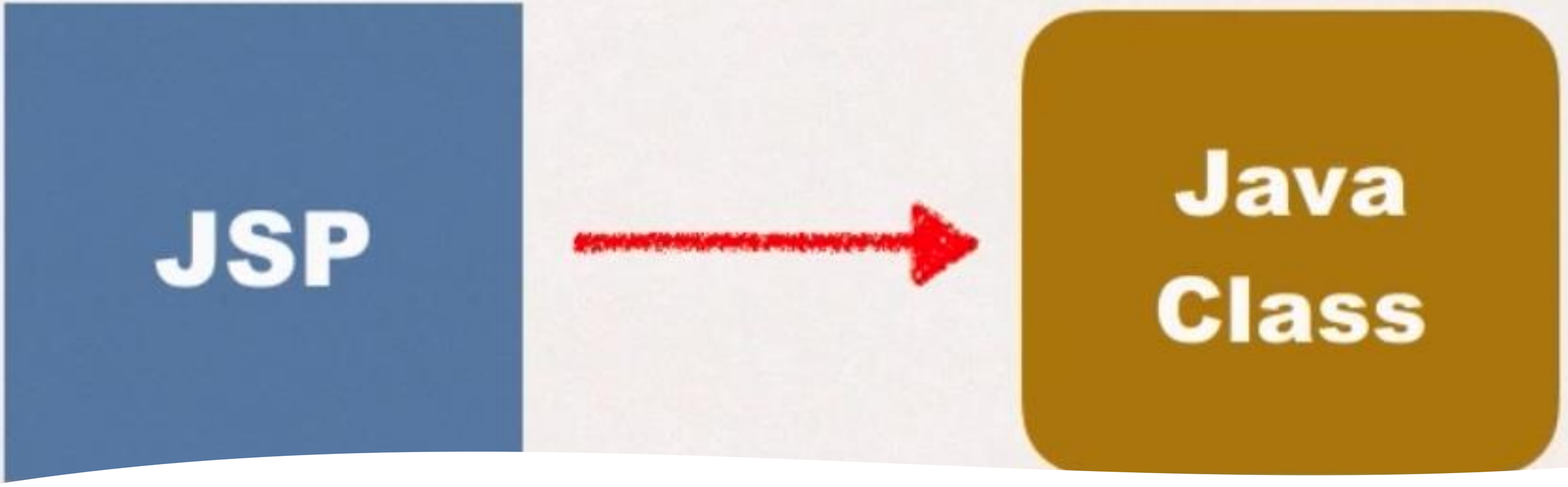
```
<%!
  String makeItLower(String data) {
    return data.toLowerCase();
  }
%>

Lower case "Hello World": <%= makeItLower("Hello World") %>
```

JSP Declarations – Best practice

- Minimize the number of declarations in a JSP
- Refactor into a separate Java class

Exercise – Vorsicht!

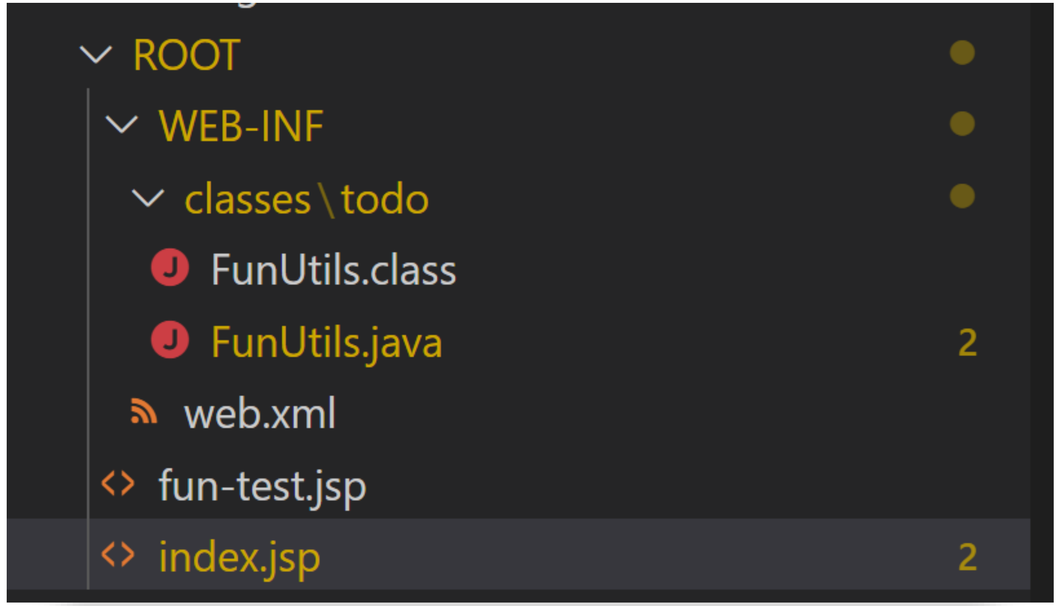
- ”Vorsicht” is a german game. You count from 1 to 50 and if the number contains 3 (like 13 or 32) or if it’s evenly dividable by 3 (like 3, 6, 9) you say ”Vorsicht!” (”look out!”), otherwise you say the number.
- Declare a method that takes a parameter (int) and returns the number (you may convert it into a String) or ”Vorsicht!”, depending on conditions above.
- Make a loop that calls the method 50 times and prints the returned value with a line break in a JSP page.
- Use the parts we’ve been looking at – expressions (if needed), scriptlets and declarations.



JSP – Best practice

- We want to move as much of our code into classes as possible.
- The classes will contain our business logic – the rules that state who may see what etc.

```java
package todo;

public class FunUtils {
    public static String makeItLower(String data) {
        return data.toLowerCase();
    }
}
```



Todo-list

- Create Java class
- Call Java class from JSP
- Now, something is off in my compiler and maybe in your's too. If needed, manually compile the .java file using `javac file.java`

Exercise

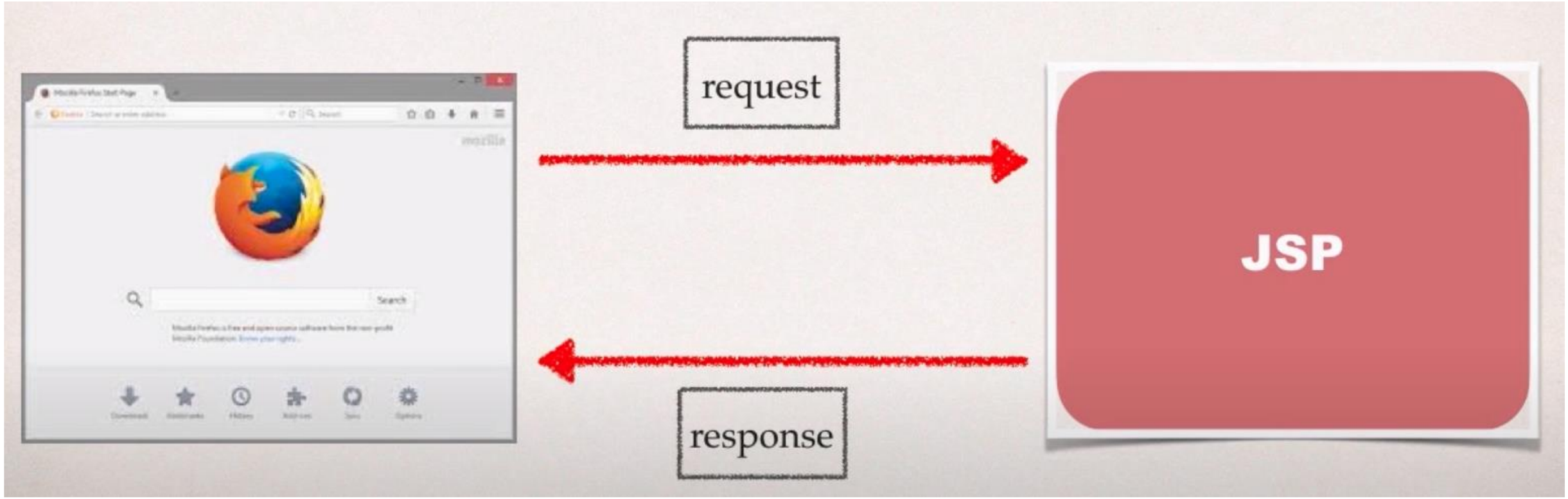
- Turn your ”Vorsicht!” game into a class.
- Let the class contain one method for checking a number, like before, and one for counting and calling the first method.
- Let the second method take a parameter that decides how many numbers to count.
  - If the user sends 100, then let the program count from 1 to 100.
- Import your class into your JSP page and call the method.

Object	Description
request	Contains HTTP request headers and form data
response	Provides HTTP support for sending response
out	JspWriter for including content in HTML page
session	Unique session for each user of the web application
application	Shared data for all users of the web application

Built-In Server Objects

Available directly in your JSP page.

HTTP Request / Response



```
<html>
2
3  <body>
4
5  <h3>JSP Built-In Objects</h3>
6
7  Request user agent: <%= request.getHeader("User-Agent") %>
8
9  <br/><br/>
10
11 Request language: <%= request.getLocale() %>
12
13 </body>
```

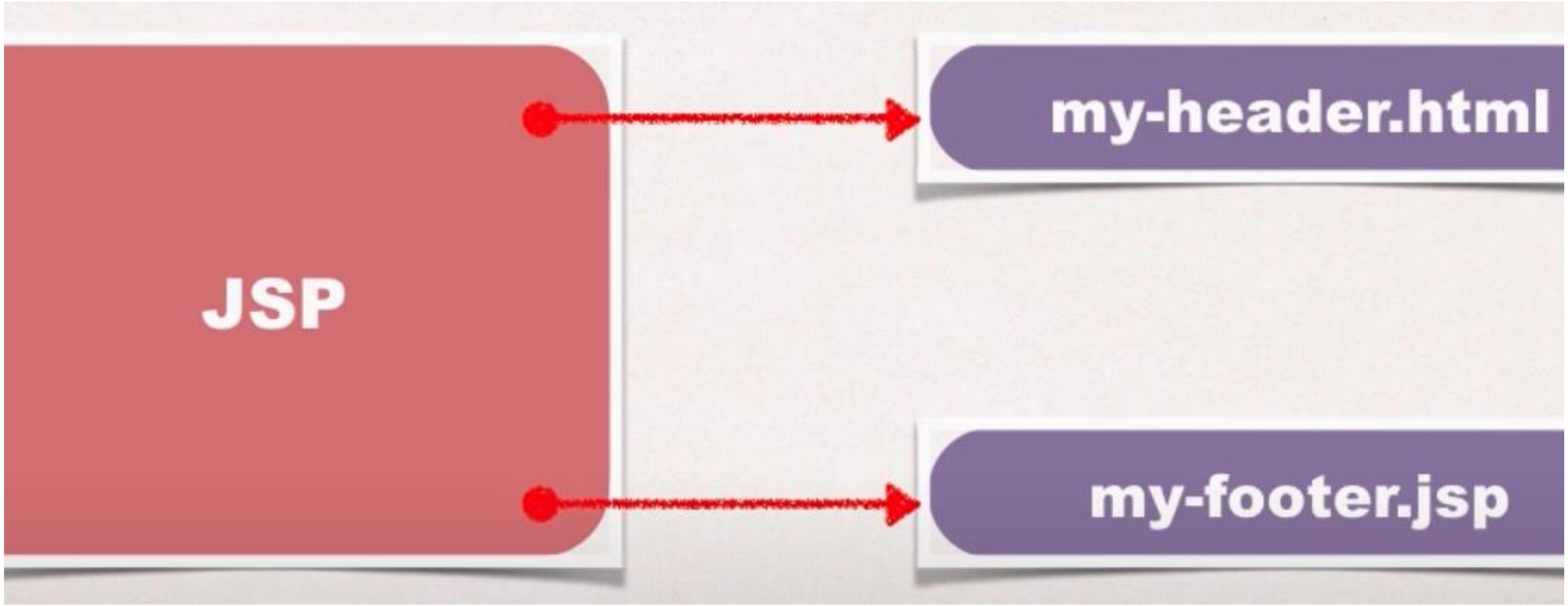
Request

Exercise

Write different content depending on the user agent. Try it out in different browsers.

List some properties you find interesting from the request object.

- https://www.decodejava.com/jsp-request-object.htm



Including files

Including files is common for common components, such as header, footer, navigation…

Including files

```
<jsp:include page="my-header.html" />
```

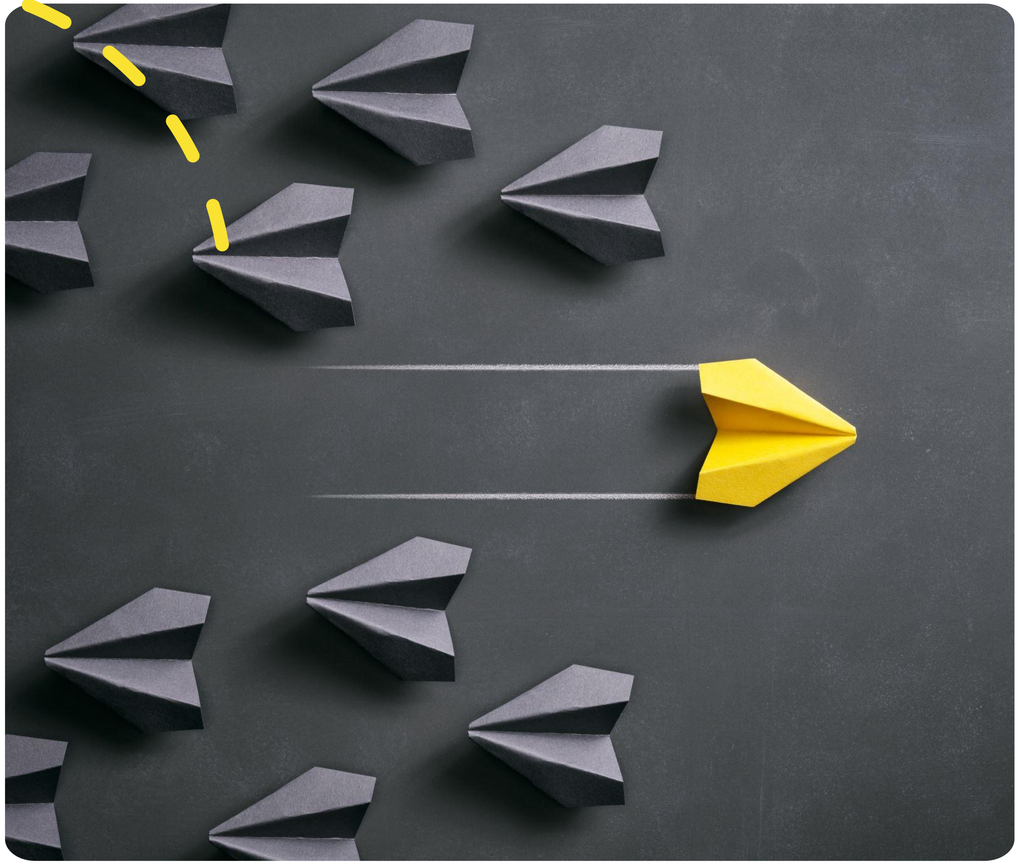
```
<jsp:include page="my-footer.jsp" />
```

Including files

```
1  <html>
2
3  <body>
4
5  <jsp:include page="my-header.html" />
6
7  Blah blah blah .... <br/> <br/>
8  Blah blah blah .... <br/> <br/>
9  Blah blah blah .... <br/> <br/>
10
11 <jsp:include page="my-footer.jsp" />
```

Exercises for today

- In Kalles gatukök, break out headers, footers and navigation into separate files and include them where they belong.
- Add a class for menu items.
  - Save a couple of menu items in a serialized file and make a method that lists them. Call the method on a suiting page.
- Print the session id somewhere. (Session is a built-in object.)
- Add a contact form to one of the pages.
- Add buttons to the menu items to put the item in a cart. (It doesn’t need to do anything yet.)
- Build a login page. (It doesn’t need to do anything yet.)
- Build a review page with a form where visitors can review the food (doesn’t need to do anything yet) and some hard coded entries.



Wait a minute! We can’t do all of this today!?
That’s fine, do as much as you can. The tasks are listed in order of priority.

Wait a minute! We need more tasks!
That’s fine, use your imagination. ;) Build a chat, maybe?