

UVM-Fag for Programmering - Java II UV-fag 6269 v.8

Indhold

UVM-Fag for Programmering - Java II UV-fag 6269 v.8.....	1
1. Installation af Apache Tomcat.....	3
Installer Java	3
Installer Tomcat.....	3
Yderligere opsætninger af Java og Tomcat	8
Start og Stop af Tomcat.....	11
JSP på Tomcat.....	14
Servlet på Tomcat.....	15
War-fil med den pakkede struktur	19
Windows og Java archives.....	20
Tomcats mappe-struktur.....	21
Installation af Eclipse	23
Ctrl-X	27
Opret en Servlet i Eclipse og deploy den på Tomcat.....	28
Deploy HelloServlet på Tomcat i Eclipse	35
Nyt projekt og Tomcat.....	38
JSP i Eclipse	39
Output fra JSP	39
En liste af indtastede navne opbevaret i session samt genkendelse af tryk på delete-knap for et navn i listen	40
Opgave 1.....	41
Opgave 2.....	41
Delete-knapper til at slette elementer i listen	42
1. Navn og delete-knap vises i hver sin form.....	42
2. Navne og tilhørende delete-knapper med fortløbende navne, vises i samme form (Eks. 1)	43
3. Navne og tilhørende delete-knapper med fortløbende navne, vises i samme form (Eks. 2)	43
4. Navne med tilhørende delete-knapper, som kalder javascript funktion, vises i samme form.	44

5. Alle navne med tilhørende delete-knapper, der hedder det samme, men med nummeret i knappens tekst, vist i samme form	46
Shoppingbasket projekt med session.....	47
JSTL i stedet for indlejret Java-kode i Products.jsp	54
Parametre	55
Session.....	62
Login-beskyttet web-side	64
Tjekke login på flere sider.....	66
Omdirigering til anden web-side.....	68
Klient-omdirigering.....	68
Server-omdirigering.....	72
Dette virker.....	74
NewName.jsp:	74
NewNameServlet.java:	74
ShowAll.jsp:	74
Java og databaser	77
Databaseeksempel	79
Noter.....	82

1. Installation af Apache Tomcat

Installer Java

Installer Java JDK, som både er Java runtime og udviklingsværktøjer.

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

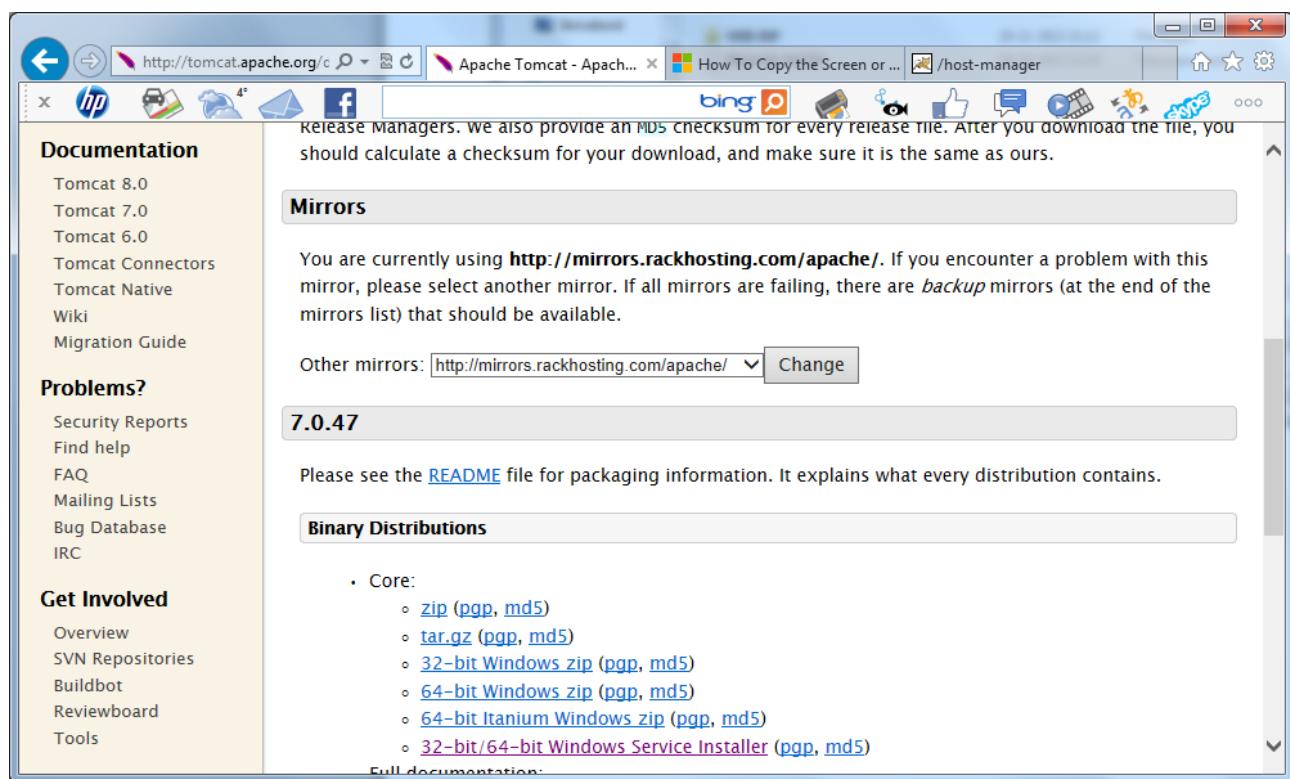
F.eks. jdk-8u101-windows-x64.exe

Installationen bliver som udgangspunkt lagt i

C:\Program Files (x86)\Java\jre7

Installer Tomcat

<http://tomcat.apache.org>

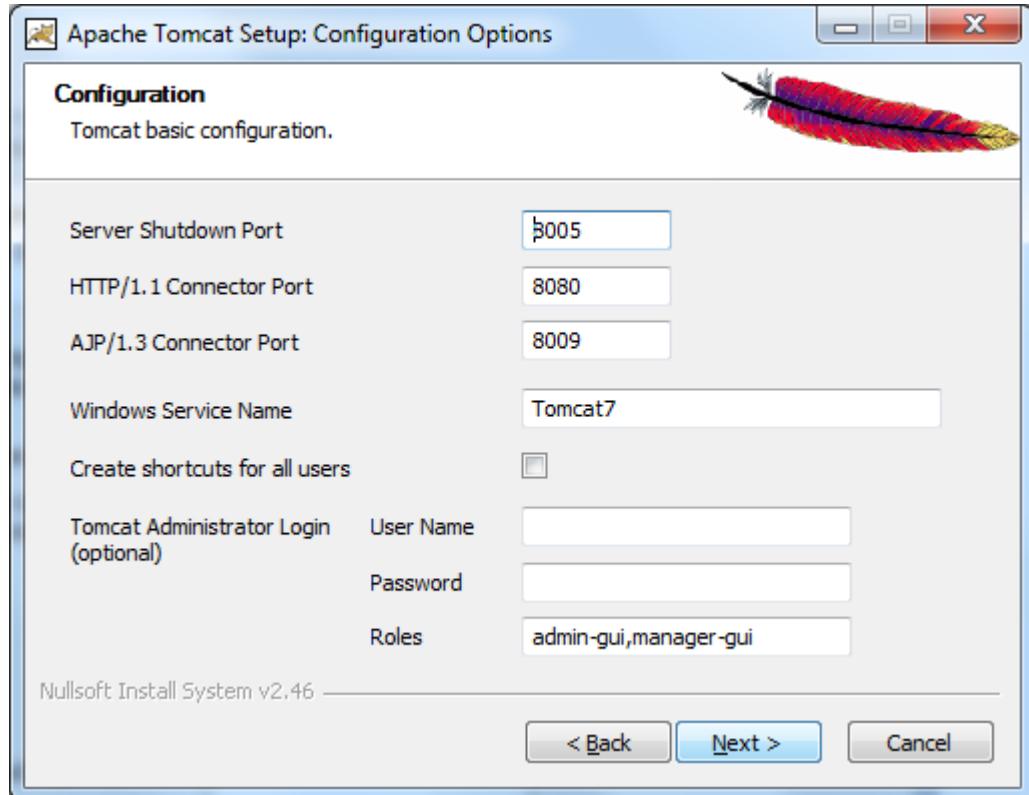
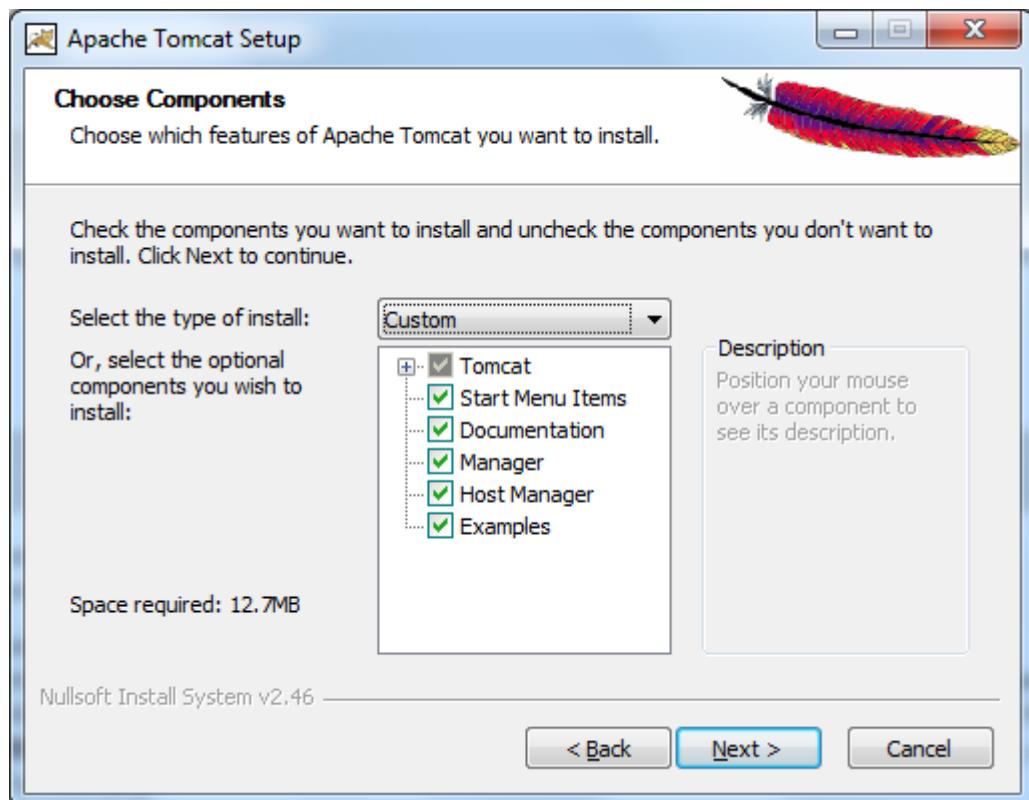


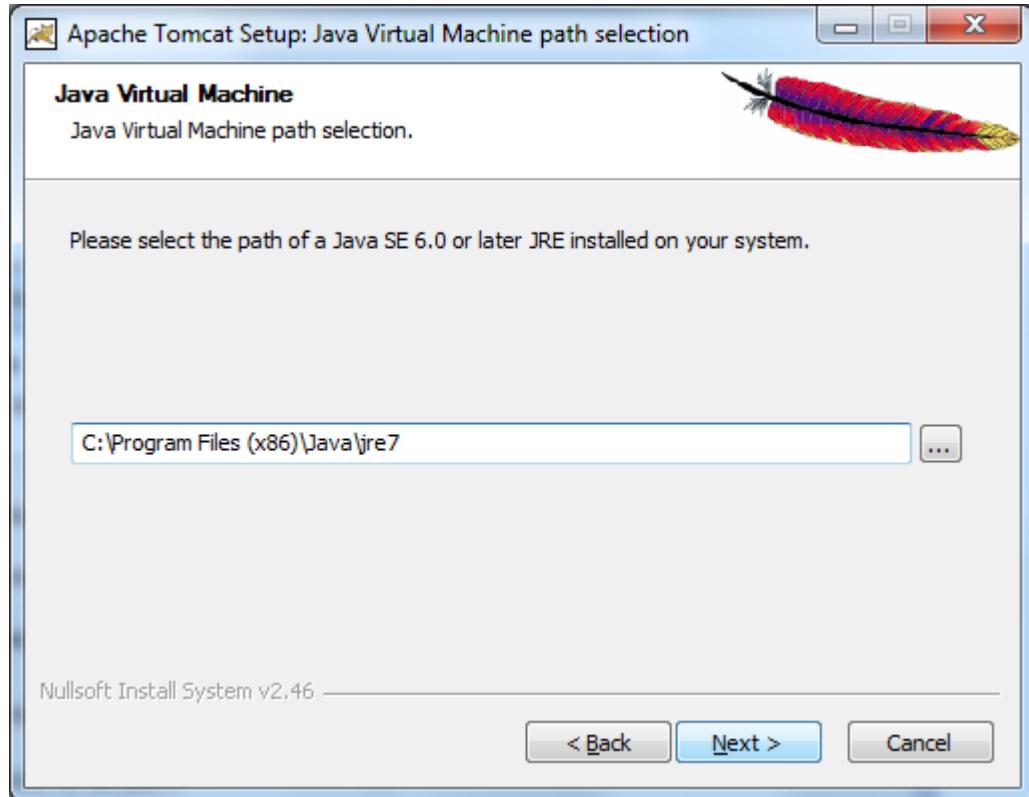
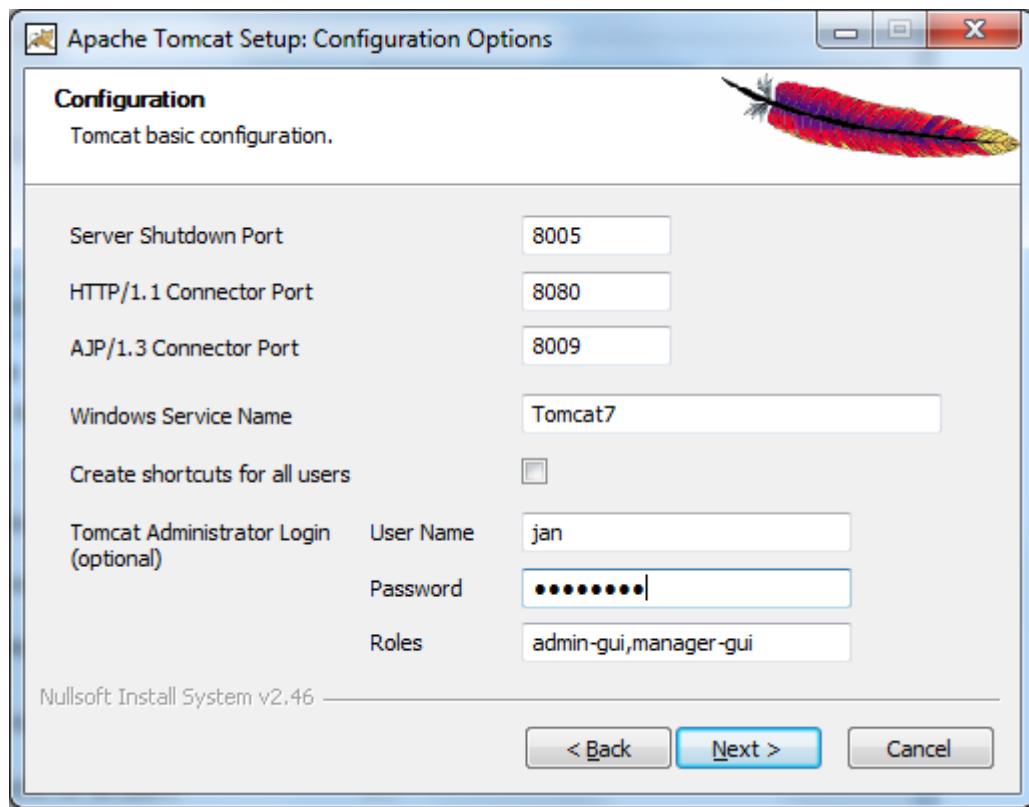
The screenshot shows a web browser window with the URL <http://tomcat.apache.org/>. The page content includes:

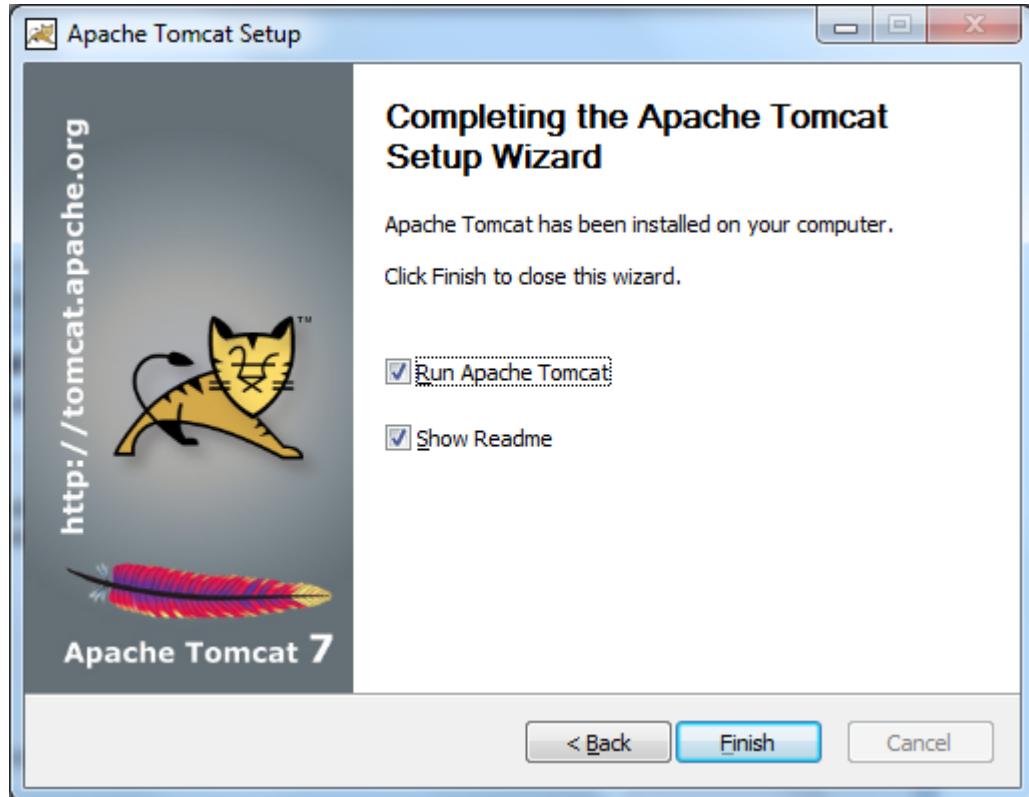
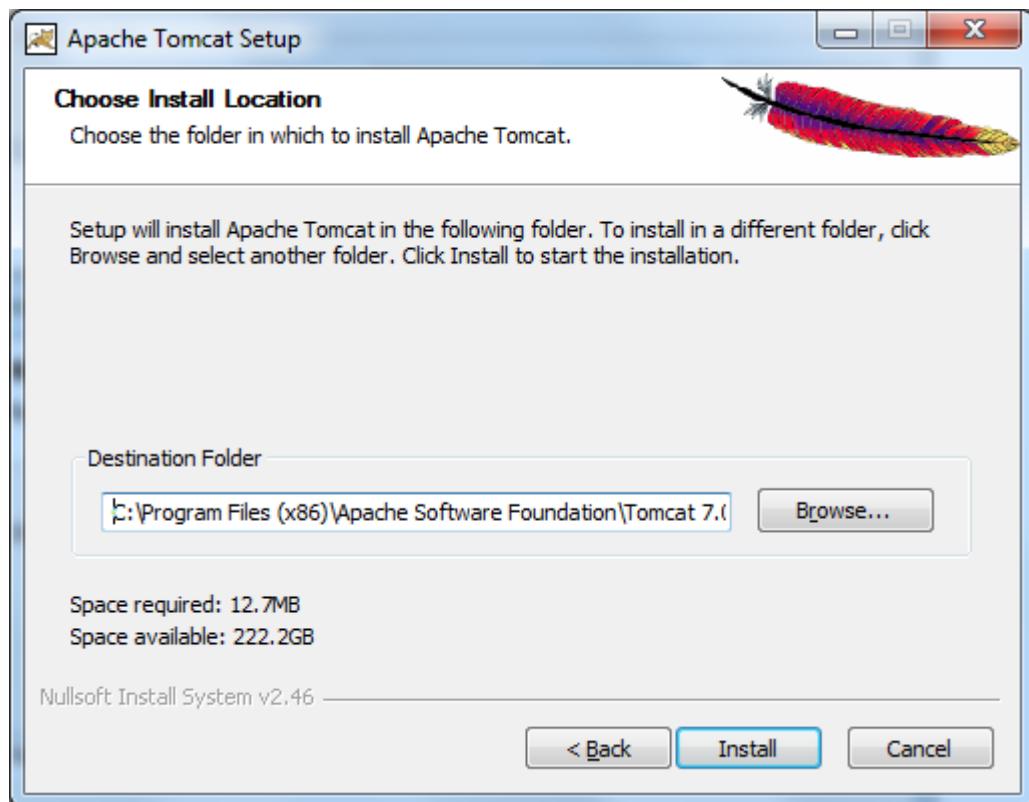
- Documentation:** Links to Tomcat 8.0, 7.0, 6.0, Connectors, Native, Wiki, and Migration Guide.
- Mirrors:** A note about MD5 checksums and a dropdown for other mirrors set to <http://mirrors.rackhosting.com/apache/>.
- 7.0.47:** A note to see the [README](#) file for packaging information.
- Binary Distributions:** A list of core distributions:
 - [zip \(pgp, md5\)](#)
 - [tar.gz \(pgp, md5\)](#)
 - [32-bit Windows zip \(pgp, md5\)](#)
 - [64-bit Windows zip \(pgp, md5\)](#)
 - [64-bit Itanium Windows zip \(pgp, md5\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, md5\)](#)

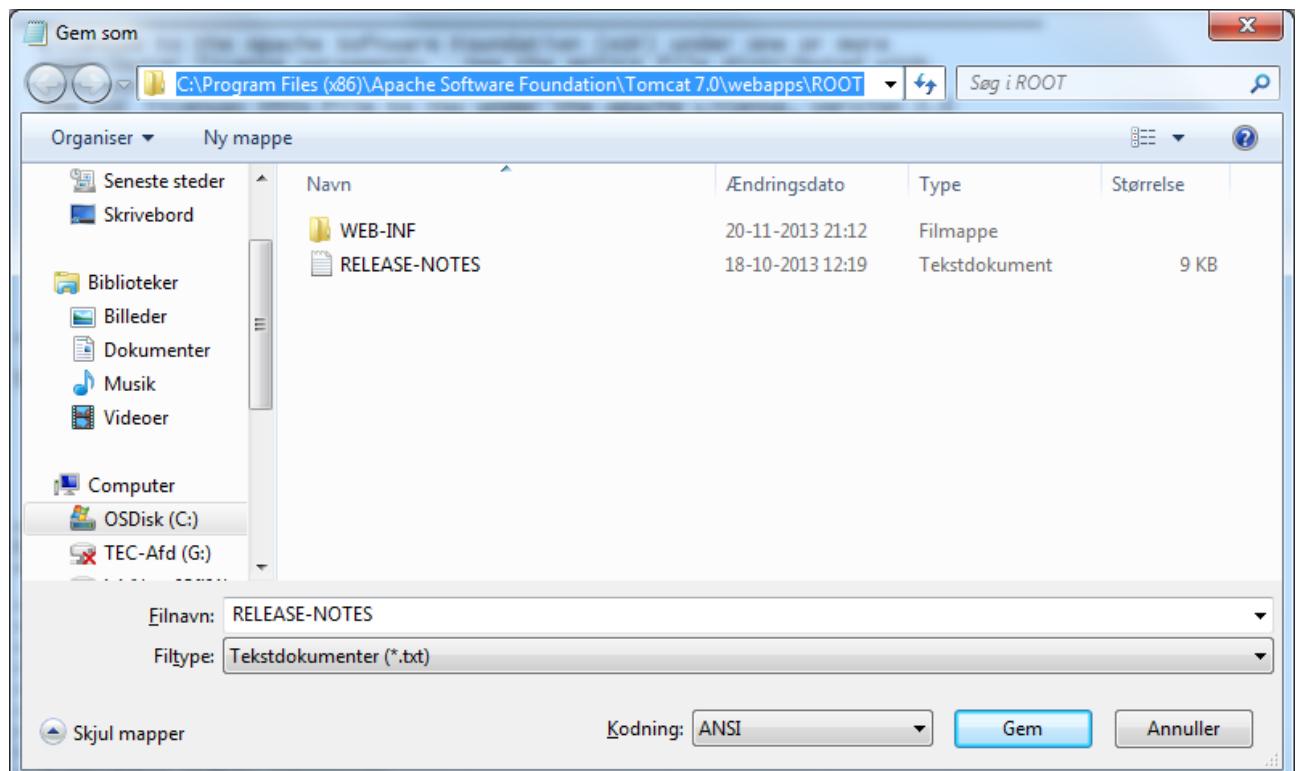
Klik på 32-bit/64-bit Windows Service Installer. Det starter installations-wizard af Apache Tomcat.

(Gem den evt. og kør den som administrator. Behøves ikke 😊)









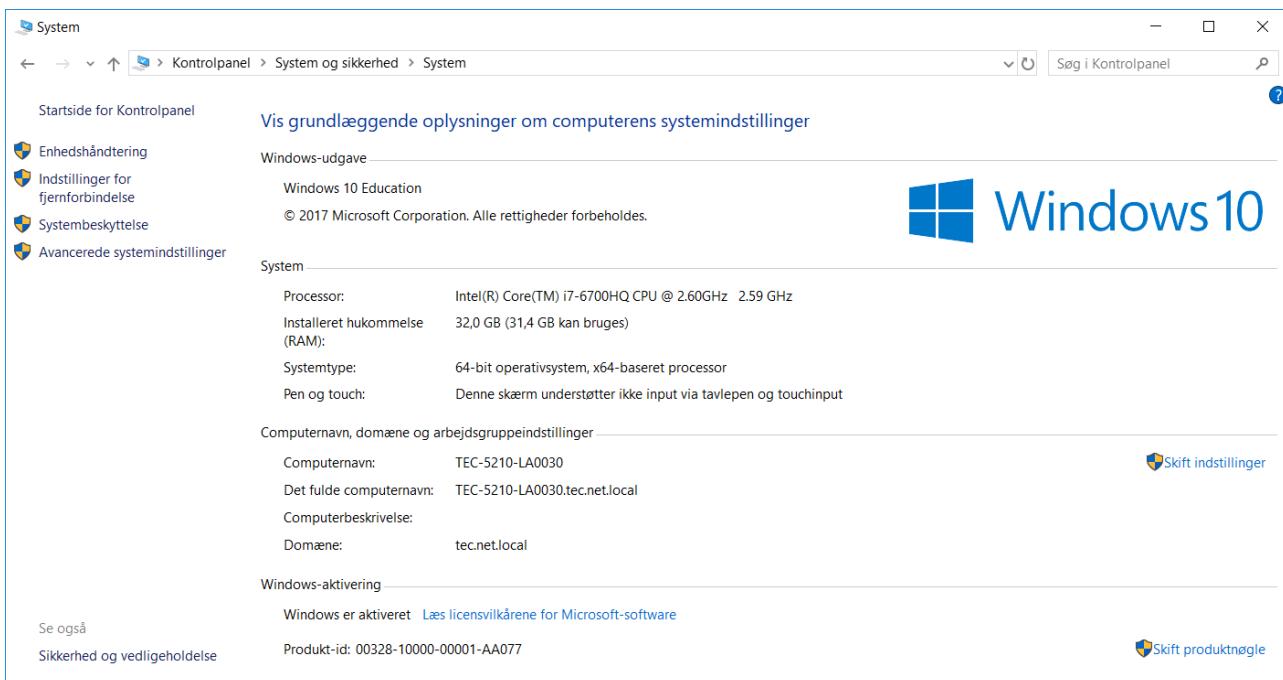
Nu er Tomcat installeret og hvis den ellers er startet, kan dens standard webside ses på følgende adresse.

<http://localhost:8080>

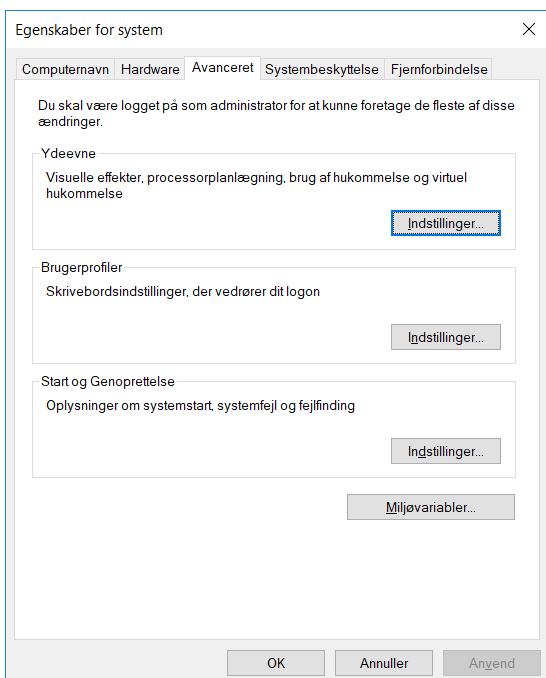
Hvis Tomcat ikke er startet, så gå alligevel videre. Det kommer om lidt.

Yderligere opsætninger af Java og Tomcat

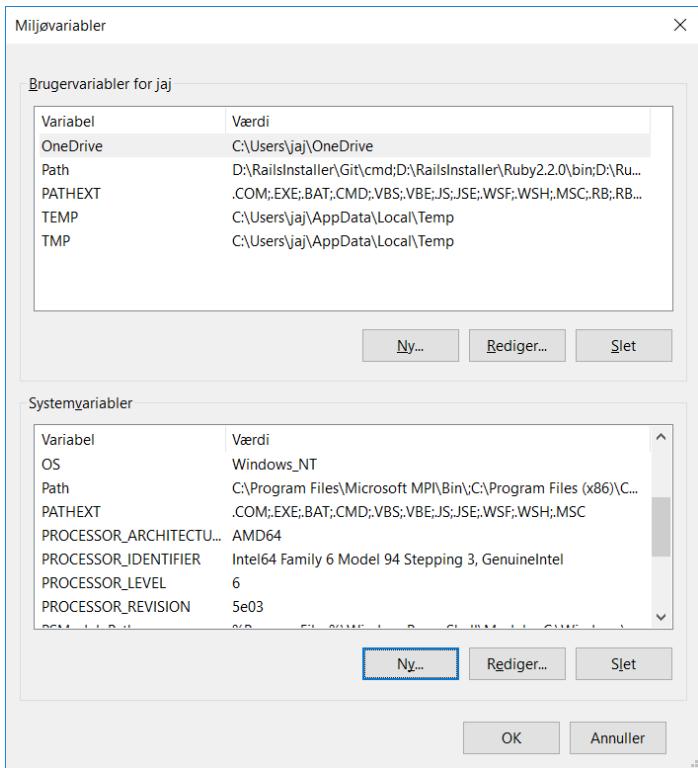
Gå i Kontrolpanel



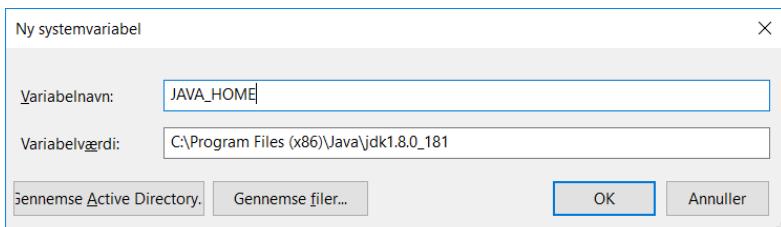
Vælg System og sikkerhed | System | Avancerede systemindstillingar.



Vælg Miljøvariabler.

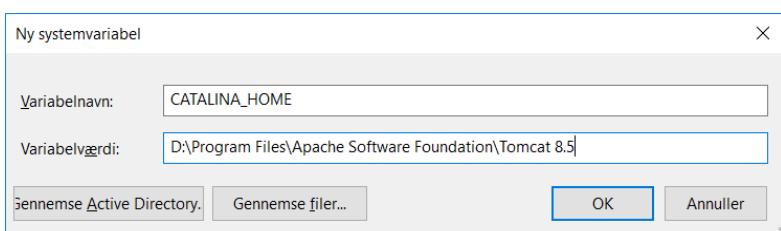


Vælg Ny og opret JAVA_HOME med Java installationens rod-mappe som vist.

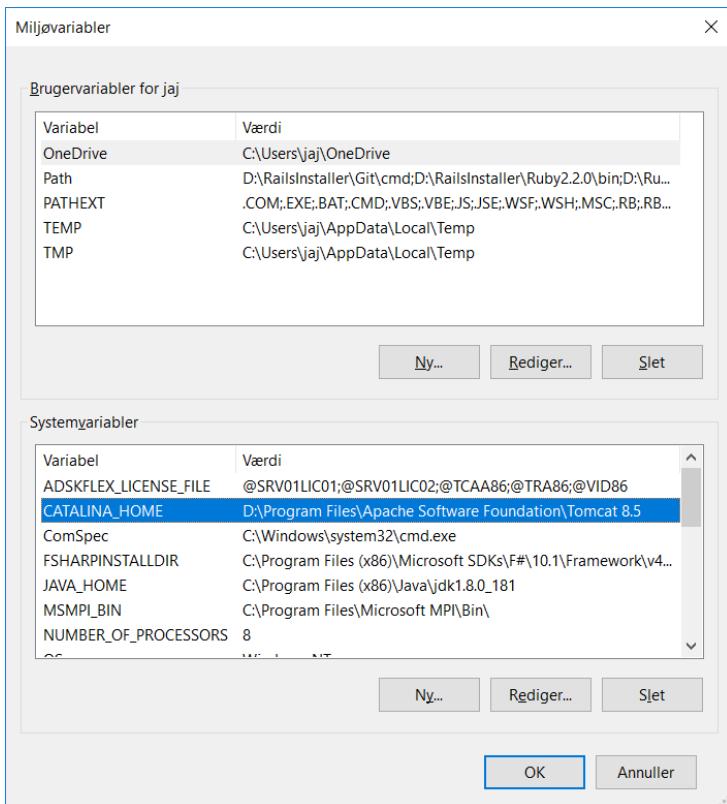


Skal være jdk-mappen for den nyinstallerede Java.

Opret CATALINA_HOME med Tomcat installationens rod-mappe.



Sådan.



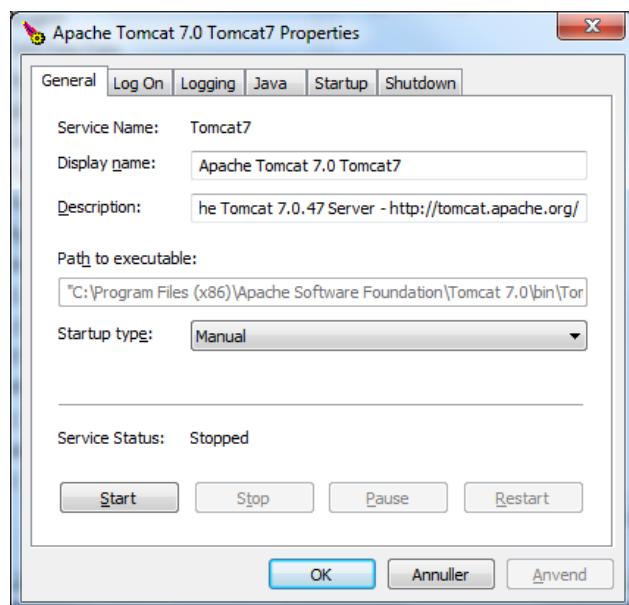
Start og Stop af Tomcat

Når Tomcat er installeret med Windows Service Installer findes der i mappen %CATALINA_HOME%\bin programmet tomcat7w.exe som kan udføres for at starte eller stoppe Tomcat.

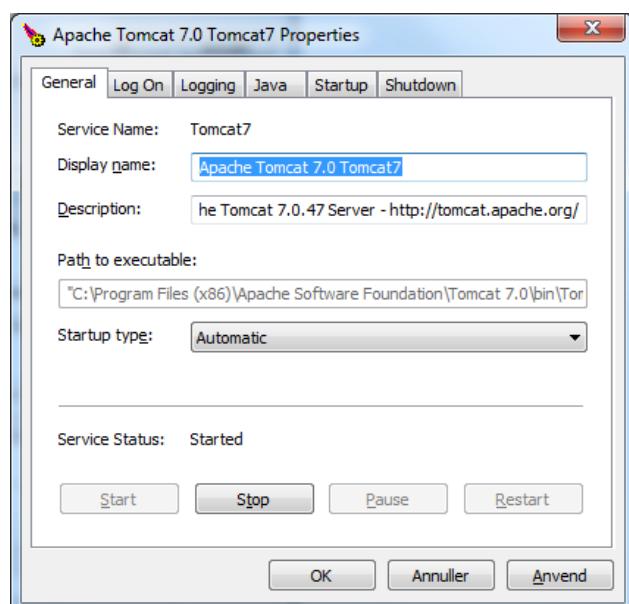
(Hvis Tomcat er installeret ved at downloade zip distributionen, findes i CATALINA_HOME\bin startup.bat og shutdown.bat samt catalina.bat, som kan udføres med argumenterne: start, stop, run, debug, version, configtest og jpda start)

Dobbeltklik på CATALINA_HOME\bin\ tomcat7w.exe

Sæt Startup Type til Automatic og tryk på Start.

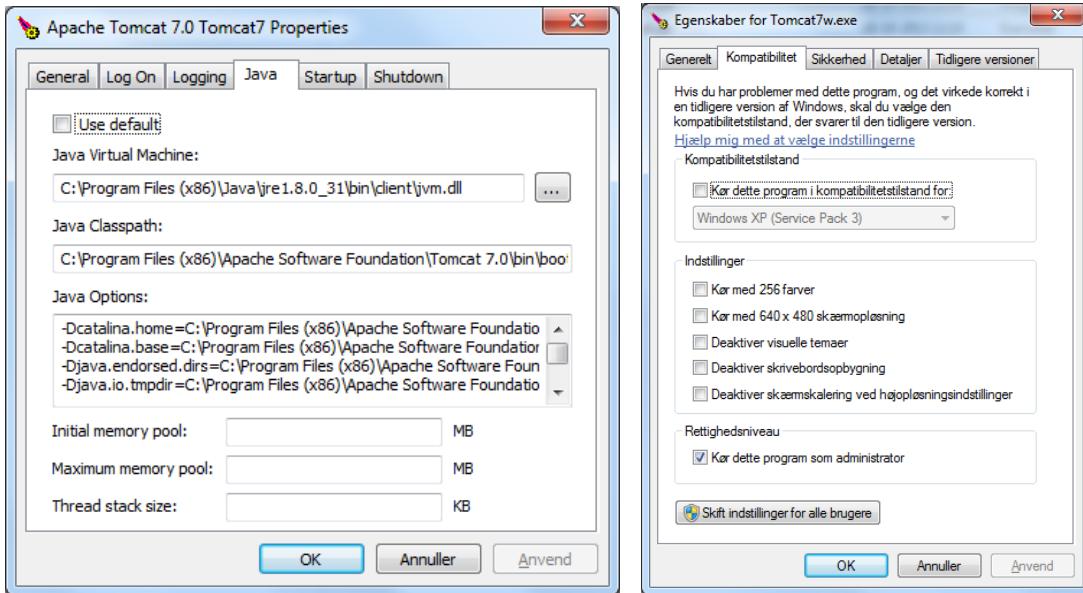


Så er Tomcat startet og kan stoppes her igen.



Der kan også i bin-mappen udføres startup.bat, som åbner en CMD, der er åben så længe serveren kører, hvor der kan udskrives fejlmeddelelser, og som kan lukkes igen, når der udføres shutdown.bat.

Hvis Tomcat7w.exe ikke kan starte Tomcat, så sørge for at Java Virtual Machine, under fanen Java, er sat rigtigt, og sæt Tomcatw.exe til altid at køres af administrator.



I browseren angives URL-adressen <http://localhost:8080> og Tomcats test-side vises.

The screenshot shows the Apache Tomcat 7.0.47 welcome page. At the top, there's a navigation bar with links to Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists. To the right of the navigation bar is a 'Find Help' button. Below the navigation bar, the title 'Apache Tomcat/7.0.47' is displayed next to the Apache Software Foundation logo. A green banner at the top of the main content area says 'If you're seeing this, you've successfully installed Tomcat. Congratulations!'. On the left side, there's a cartoon cat logo. In the center, under 'Recommended Reading:', there are links to 'Security Considerations HOW-TO', 'Manager Application HOW-TO', and 'Clustering/Session Replication HOW-TO'. On the right side, there are three buttons: 'Server Status', 'Manager App', and 'Host Manager'. Below the main content area, there's a 'Developer Quick Start' section with links to 'Tomcat Setup', 'First Web Application', 'Realms & AAA', 'JDBC DataSources', 'Examples', 'Servlet Specifications', and 'Tomcat Versions'.

Portnummeret 8080 er sat op i filen conf\server.xml.

The screenshot shows a Windows Notepad window titled 'server.xml - Notesblok'. The window displays the XML configuration file 'server.xml'. The XML code defines two connectors: one for port 8080 using the 'HTTP/1.1' protocol and another for port 8443 using the 'SSL' protocol. The code also includes comments explaining the purpose of each connector and the shared thread pool.

```

<!-- A "Connector" represents an endpoint by which requests are
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
<!-- Define a SSL HTTP/1.1 Connector on port 8443
    This connector uses the BIO implementation that requires the
    APR library installed on the system.
-->

```

JSP på Tomcat

I Tomcat webapps oprettes mappen HelloJSP og her lægges filen HelloJSP.jsp med følgende indhold.

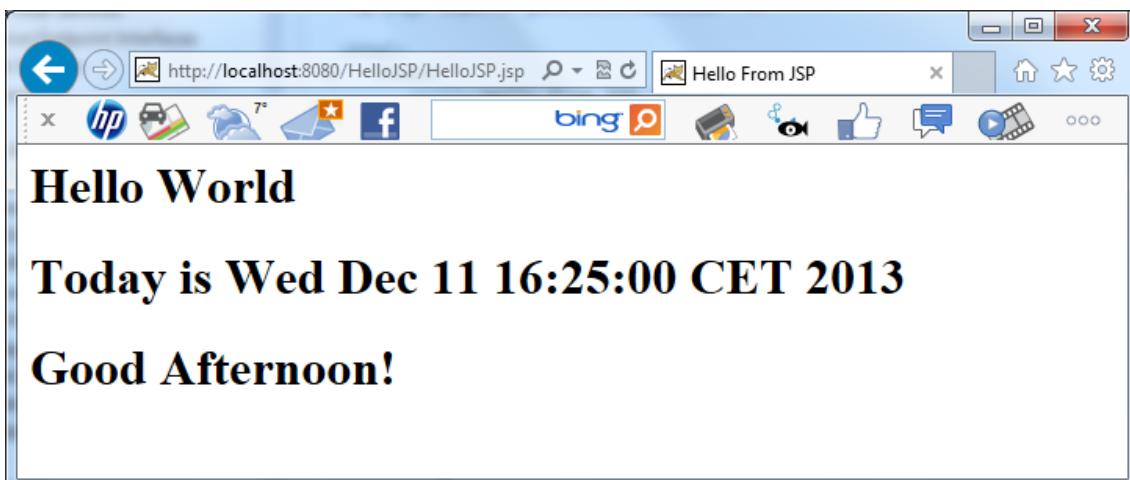
En JSP-fil er et script, der indeholder en blanding af HTML og Java, og som fortolkes af Tomcat.

Filen webapps\HelloJSP\HelloJSP.jsp ser sådan ud:

```
<%@ page import="java.util.Calendar" %>
<html>
    <head>
        <title>Hello From JSP</title>
    </head>
    <body>
        <h1>Hello World</h1>
        <%
            Calendar cal = Calendar.getInstance();

            String greeting;
            int hourOfDay = cal.get(Calendar.HOUR_OF_DAY);

            if (hourOfDay < 12)
            {
                greeting = "Good Morning!";
            }
            else if (hourOfDay >= 12 && hourOfDay < 18)
            {
                greeting = "Good Afternoon!";
            }
            else if (hourOfDay >= 18 && hourOfDay < 23)
            {
                greeting = "Good Evening!";
            }
            else
            {
                greeting = "Good Night!";
            }
        %>
        <h1>Today is <%=cal.getTime()%></h1>
        <h1><%=greeting%></h1>
    </body>
</html>
```



Servlet på Tomcat

En servlet er en ren Java-klasse, som compileres. Output, som svar på request, gives via et PrintWriter-objekt som hentes fra response-objektet.

Følgende fil HelloServlet.java kan placeres i Tomact i mappen webapps\HelloServlet.

```
package dk.tec.jaj;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException
    {
        Calendar cal = Calendar.getInstance();

        String greeting;
        int hourOfDay = cal.get(Calendar.HOUR_OF_DAY);

        if (hourOfDay < 12)
        {
            greeting = "Good Morning!";
        }
        else if (hourOfDay >= 12 && hourOfDay < 18)
        {
            greeting = "Good Afternoon!";
        }
        else if (hourOfDay >= 18 && hourOfDay < 23)
        {
            greeting = "Good Evening!";
        }
        else
        {
            greeting = "Good Night!";
        }

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("<h1>Today it is " + cal.getTime().toString() + "</h1>");
        out.println("<h1>" + greeting + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Denne java-fil kan kompileres til en class-fil i en Windows CMD med følgende kommando. For at få adgang til Servlet-ressourcerne benyttes –classpath option til servlet-api.jar i Tomcat.

```
C:\...\>javac -classpath "%CATALINA_HOME%\lib\servlet-api.jar" HelloServlet.java
```

Det kan også gøres i Power Shell, som i Stifinder kan åbnes i den aktuelle mappe ved at højreklikke i mappen samtidig med at Shift-tasten holdes nede, og så vælge Åben PowerShell-vinduet her.

```
javac -classpath "$env:CATALINA_HOME\lib\servlet-api.jar" HelloServlet.java
```

Følgende viser alle miljøvariabler i PowerShell

```
PS > Get-childItem env:
```

Og denne viser CATALINA_HOME

```
PS > Get-childItem env:CATALINA_HOME
```

Name	Value
---	---
CATALINA_HOME	D:\Program Files\Apache Software Foundation\Tomcat 8.5

Hvis javac.exe ikke kan findes af systemet, skal der oprettes sti til den i Windows-path. Gå ind i miljøvariabler og tilføj stien f.eks. C:\Program Files\Java\jdk1.7.0_45\bin til Path-variablen. Stierne adskilles af (;).

Pas på ikke at slette det eksisterende indhold. Tilføj i slutningen en (;) og derefter stien.

Hvis man kommer til at slette Path-indholdet så kan den findes i registreringsdatabasen.

Kør regedit.

Navn	Type	Data
ab\ (Standard)	REG_SZ	(værdien er ikke defineret)
ab\ ComSpec	REG_EXPAND_SZ	%SystemRoot%\system32\cmd.exe
ab\ FP_NO_HOST_C...	REG_SZ	NO
ab\ FPPUILang	REG_SZ	en-US
ab\ NUMBER_OF_PR...	REG_SZ	4
ab\ OnlineServices	REG_SZ	Online Services
ab\ OOBUILang	REG_SZ	da-DK
ab\ OS	REG_SZ	Windows_NT
ab\ Path	REG_EXPAND_SZ	C:\ProgramData\Oracle\Java\javapath;c:\Program...
ab\ PATHEXT	REG_SZ	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WS...
ab\ PCBRAND	REG_SZ	Pavilion
ab\ Platform	REG_SZ	BNB
ab\ PROCESSOR_AR...	REG_SZ	AMD64
ab\ PROCESSOR_IDE...	REG_SZ	Intel64 Family 6 Model 69 Stepping 1, GenuineIntel
ab\ PROCESSOR_LE...	REG_SZ	6
ab\ PROCESSOR_RE...	REG_SZ	4501
ab\ PSModulePath	REG_EXPAND_SZ	%SystemRoot%\system32\WindowsPowerShell\v1...
ab\ PTSMINSTALLP...	REG_SZ	c:\Program Files (x86)\Hewlett-Packard\HP Protec...
ab\ TEMP	REG_EXPAND_SZ	%SystemRoot%\TEMP
ab\ TMP	REG_EXPAND_SZ	%SystemRoot%\TEMP
ab\ USERNAME	REG_SZ	SYSTEM
ab\ VS110COMNTO...	REG_SZ	C:\Program Files (x86)\Microsoft Visual Studio 11.0...

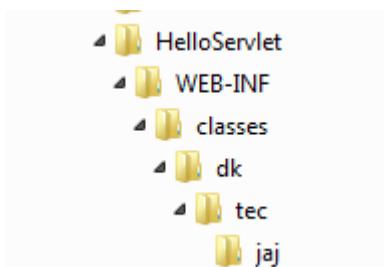
Kommandoen

C:\>javac -classpath "%CATALINA_HOME%\lib\servlet-api.jar" HelloServlet.java

resulterer i filen HelloServlet.class, som lægges i samme mappe som kommandoen udføres.

I webapps\HelloServlet skal der nu oprettes de nedenfor viste undermapper. Klassen kan nemlig kun udføres hvis den ligger i en mappestruktur svarende til den angivne package dk.tec.jaj, som er angivet øverst i filen.

I mappen jaj lægges den compilede klasse HelloServlet.class.



(Dette kan evt. også gøres med "-d ." i compilerkommandoen, som skal udføres i mappen classes. Så oprettes package-stien og HelloServlet.class lægges under dk.tec.jaj.)

I mappen WEB-INF lægges web.xml med følgende indhold.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    
```

```

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">

<display-name>
    Hello World
</display-name>
<servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>
        dk.tec.jaj.HelloServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>

</servlet>
<servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/Hello.html</url-pattern>
</servlet-mapping>

</web-app>

```

Der er to interessante ting i denne web.xml.

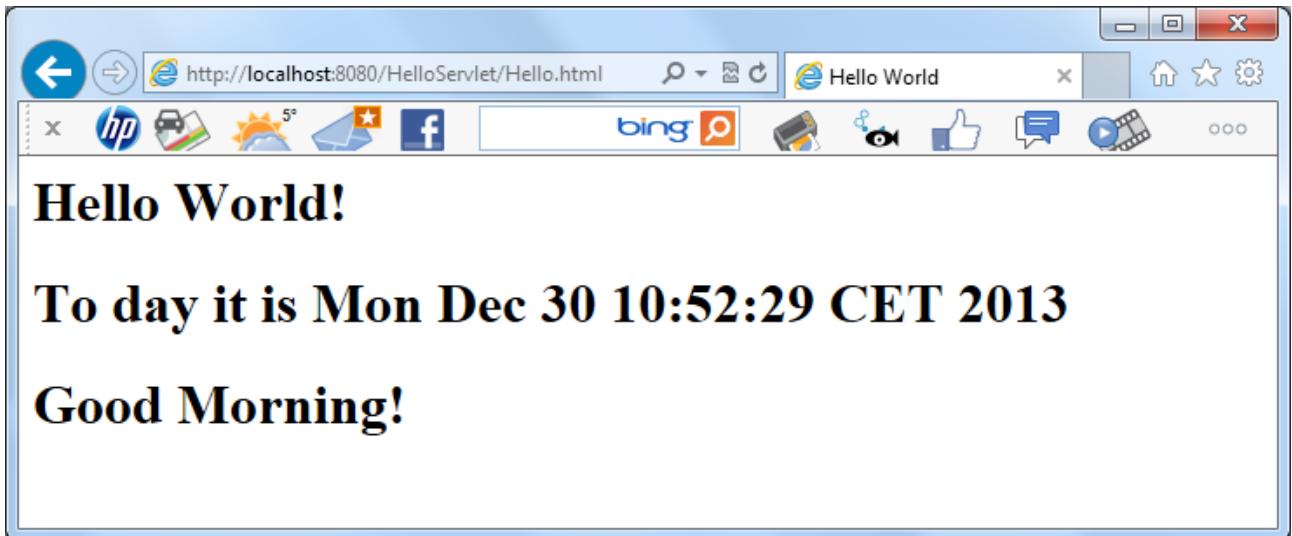
Tagget servlet-class angiver hvor klassen er placeret under mappen classes.

Tagget servlet-mapping med tagget url-pattern angiver en alias for hvad klienten skal requeste for at få fat i Servletten. Her Hello.html, som jo ikke eksisterer fysisk.

Den fås nu fat i med URL-adressen <http://localhost:8080>HelloServlet>Hello.html> i browseren.

I HelloServlet.java ses annotation `@WebServlet("/HelloServlet")`.

Dette gør at den også kan fås fat i på <http://localhost:8080>HelloServlet>HelloServlet>. Det vil så også virke uden web.xml.



War-fil med den pakkede struktur

Hele den ovenfor viste struktur kan lægges i en pakket Web-archive fil.

I en Windows CMD-prompt, i mappen webapps\HelloServlet kan skrives følgende kommando, som opretter en jar-fil men med extension .war. En jar-fil er en Java-archive og en War-fil er en Web-archive. Det er kun extension der er til forskel, og WinZip kan også udpakke en sådan.

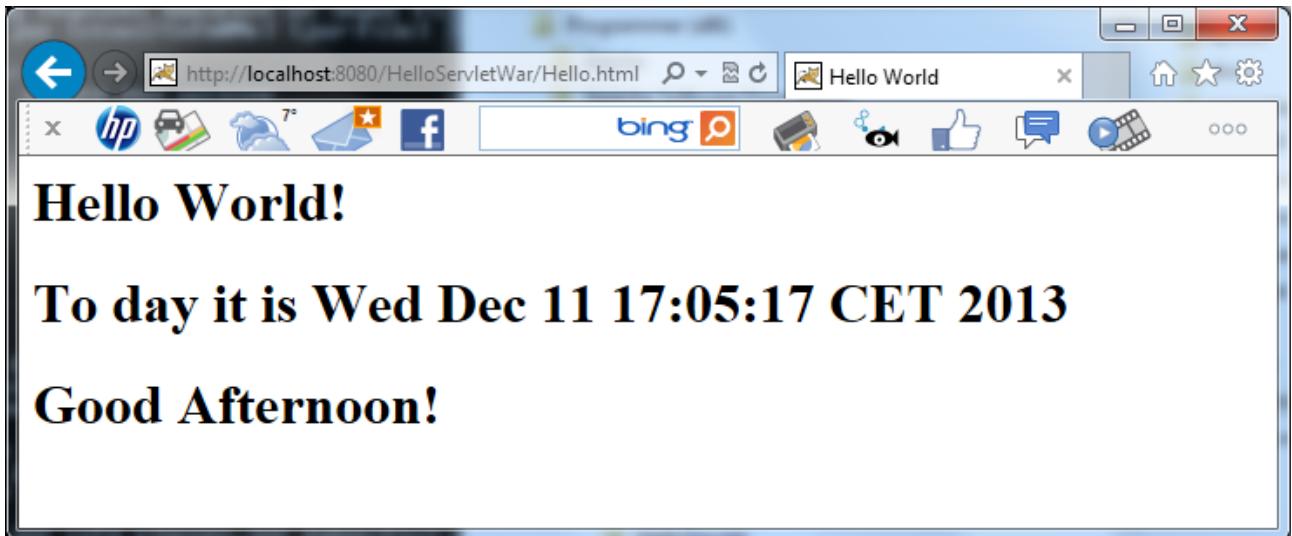
Det afsluttende punktum i kommandoen fortæller at det er alt indhold i current directory, der skal pakkes.

```
jar cvf HelloServletWar.war .
```

Dette kan også gøres i PowerShell.

Den resulterende fil HelloServletWar.war lægges i Tomcats mappe webapps og svarer nu til en mappe af navnet HelloServletWar og med det indhold der var i HelloServlet mappen. Det udpakkes faktisk af Tomcat så det hele ser ud som før, nu bare med mappenavnet HelloServletWar.

Den fås nu fat i med URL-adressen <http://localhost:8080>HelloServletWar/Hello.html> i browseren.



Windows og Java archives

Java applikationer pakkes ofte i .jar (Java ARchive) filer, Java web-applikationer pakkes i .war (Web ARchive) filer, og Java EE pakkes i .ear(Enterprise ARchive).

For at kunne se i disse i Windows, skal Windows lige have at vide at det er pakkede filer.

Følgende kommandoer udføres i en CMD-prompt:

```
assoc .jar=CompressedFolder  
assoc .war=CompressedFolder  
assoc .ear=CompressedFolder
```

En jar-fil (eller war-fil) kan i Windows-prompt udpakkes med kommandoen

```
C:\>jar -xf <jarfil>.jar
```

Tomcats mappe-struktur

Tomcats mappestruktur, altså mapperne under CATALINA_HOME, er som følger.

Directory	Contents
/bin	Contains the startup and shutdown scripts for both Windows and Linux. Jar files with classes required for tomcat to start are also stored here.
/conf	Contains the main configuration files for Tomcat. The two most important are server.xml and the global web.xml. Server.xml: PortNo
/lib	Contains the Tomcat Java Archive (jar) files, shared across all Tomcat components. All web applications deployed to Tomcat can access the libraries stored here. This includes the Servlet API and JSP API libraries.
/logs	Contains Tomcat's log files.
/temp	Temporary file system storage.
/webapps	The directory where all web applications are deployed, and where you place your WAR file when it is ready for deployment.
/work	Tomcat's working directory where Tomcat places all servlets that are generated from JSPs. If you want to see exactly how a particular JSP is interpreted, look in this directory.

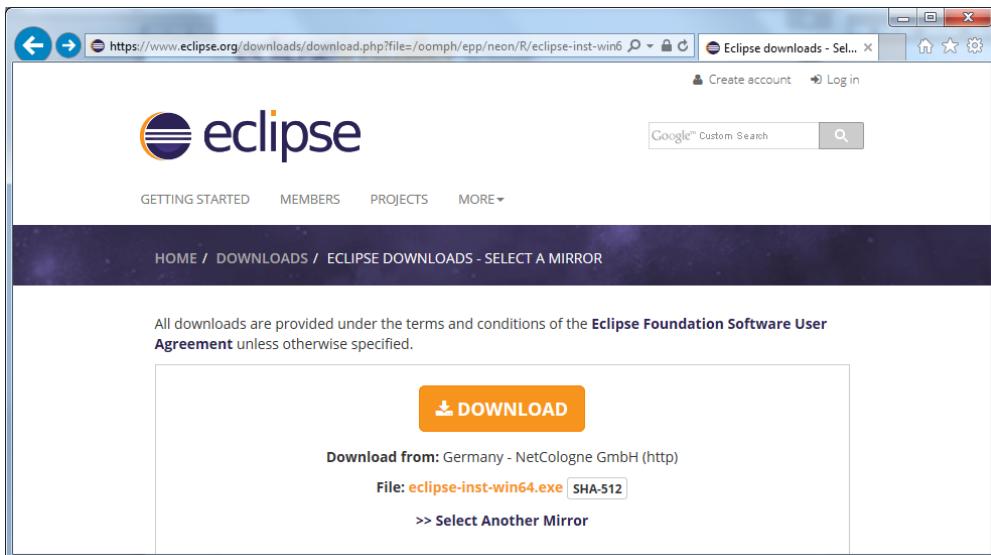
Web Application Directory Structure

Mappestrukturen i en web-applikation er som følger. Her med et projekt af navnet MyWeb. Mappen MyWeb ligger i Tomcats mappe webapps.

Directory	Description
/MyWeb/	The root directory of the web application. All JSP and HTML files should be stored here. Usually each type of static content is stored in a separate subdirectory (images/, styles/, js/).
/MyWeb/WebContent/WEB-INF	Contains all resources related to the application that are not in the document root of the application. This is where your web application deployment descriptor, web.xml, is located. Note that the WEB-INF directory is not part of the public document. No files contained in this directory can be requested directly by a client.
/MyWeb/WebContent/WEB-INF/classes	Where servlet and utility classes are located.
/MyWeb/WebContent/WEB-INF/lib	Contains Java archive files (jar libraries) that the web application is dependent upon. For example, this is where you would place a jar file that contained a JDBC driver or JSP tag library.
/MyWeb/WebContent/jsp	
/MyWeb/WebContent/htmls	
/MyWeb/WebContent/images	
/MyWeb/WebContent/scripts	
/MyWeb/WebContent/styles	

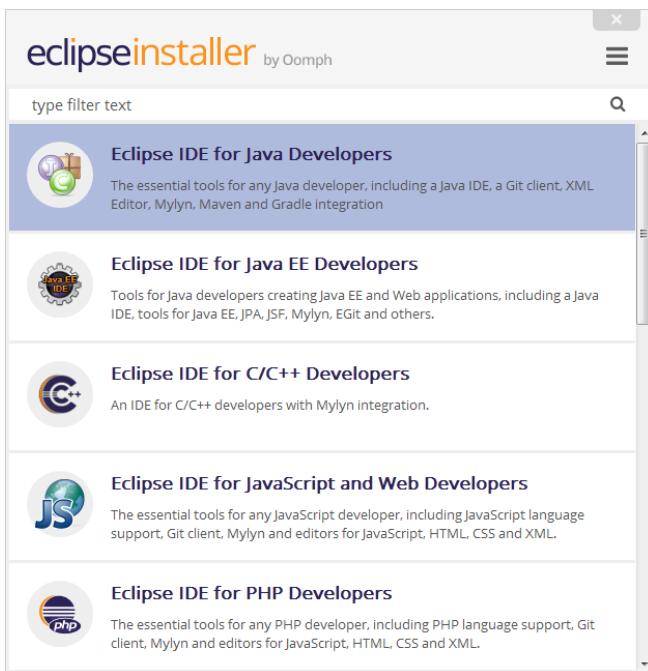
Installation af Eclipse

Hent installer fra eclipse.org.



The screenshot shows the Eclipse download page at <https://www.eclipse.org/downloads/download.php?file=/oomph/epp/neon/R/eclipse-inst-win64.exe>. The page features the Eclipse logo and navigation links for 'GETTING STARTED', 'MEMBERS', 'PROJECTS', and 'MORE'. A banner at the top says 'HOME / DOWNLOADS / ECLIPSE DOWNLOADS - SELECT A MIRROR'. Below the banner, a note states: 'All downloads are provided under the terms and conditions of the [Eclipse Foundation Software User Agreement](#) unless otherwise specified.' A large orange 'DOWNLOAD' button is centered, with the text 'Download from: Germany - NetCologne GmbH (http)' and 'File: eclipse-inst-win64.exe SHA-512' below it. A link '»> Select Another Mirror' is also present.

Udfør den og vælg Eclipse IDE for Java Developers.



The screenshot shows the Eclipse Installer interface titled 'eclipseinstaller by Oomph'. It lists several development environments:

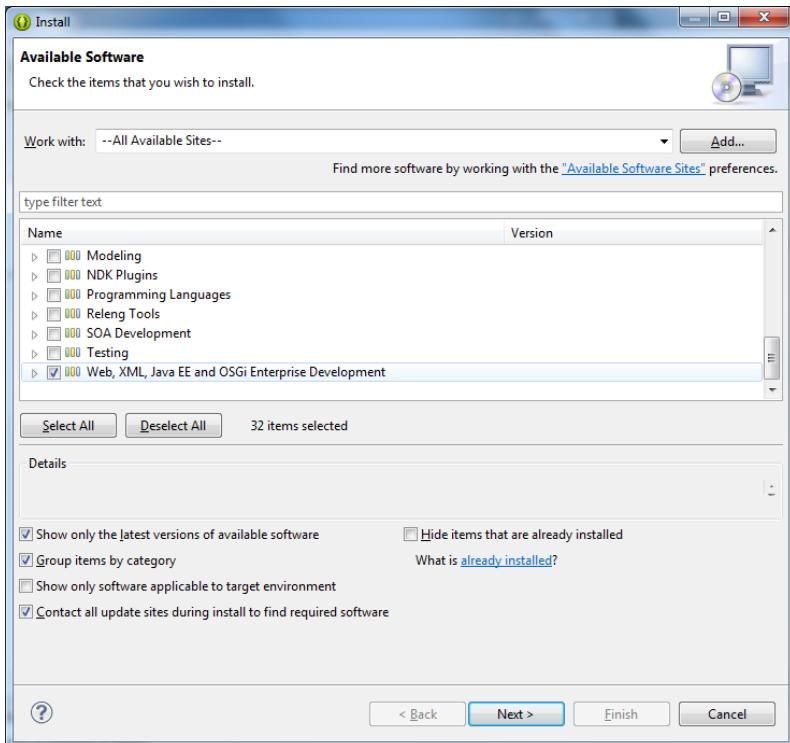
- Eclipse IDE for Java Developers**: The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration.
- Eclipse IDE for Java EE Developers**: Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.
- Eclipse IDE for C/C++ Developers**: An IDE for C/C++ developers with Mylyn integration.
- Eclipse IDE for JavaScript and Web Developers**: The essential tools for any JavaScript developer, including JavaScript language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.
- Eclipse IDE for PHP Developers**: The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Start Eclipse med Eclipse.exe (opret evt. en genvej på skrivebordet)

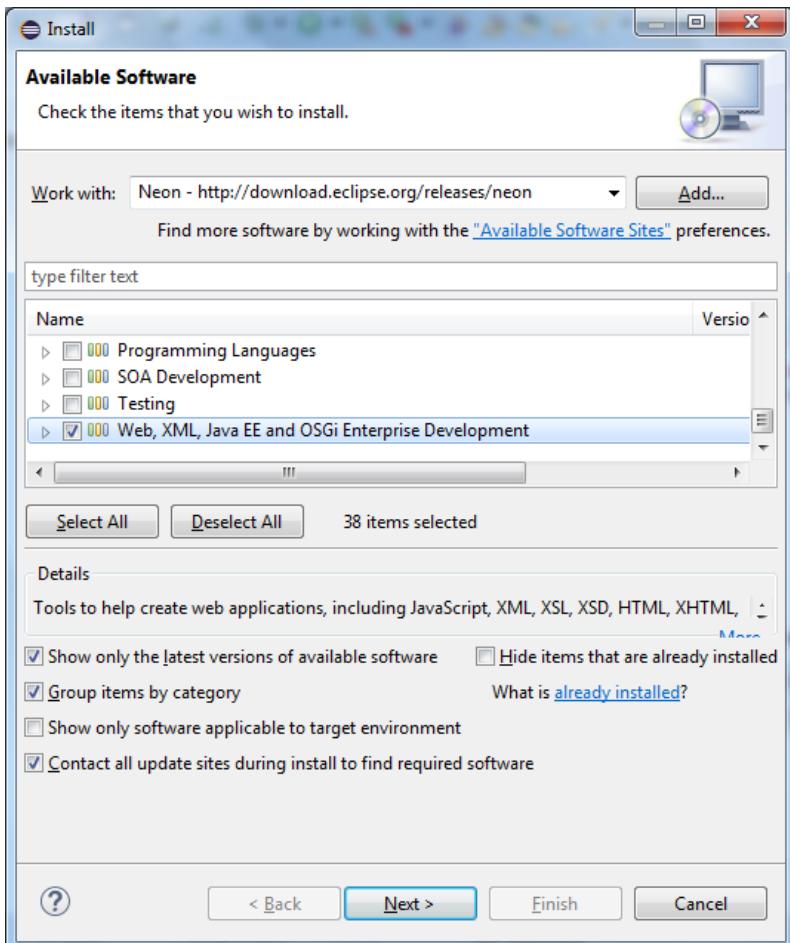
I Eclipse menuen Help vælg Install New Software...

Vælg Work with: --All Available Sites—

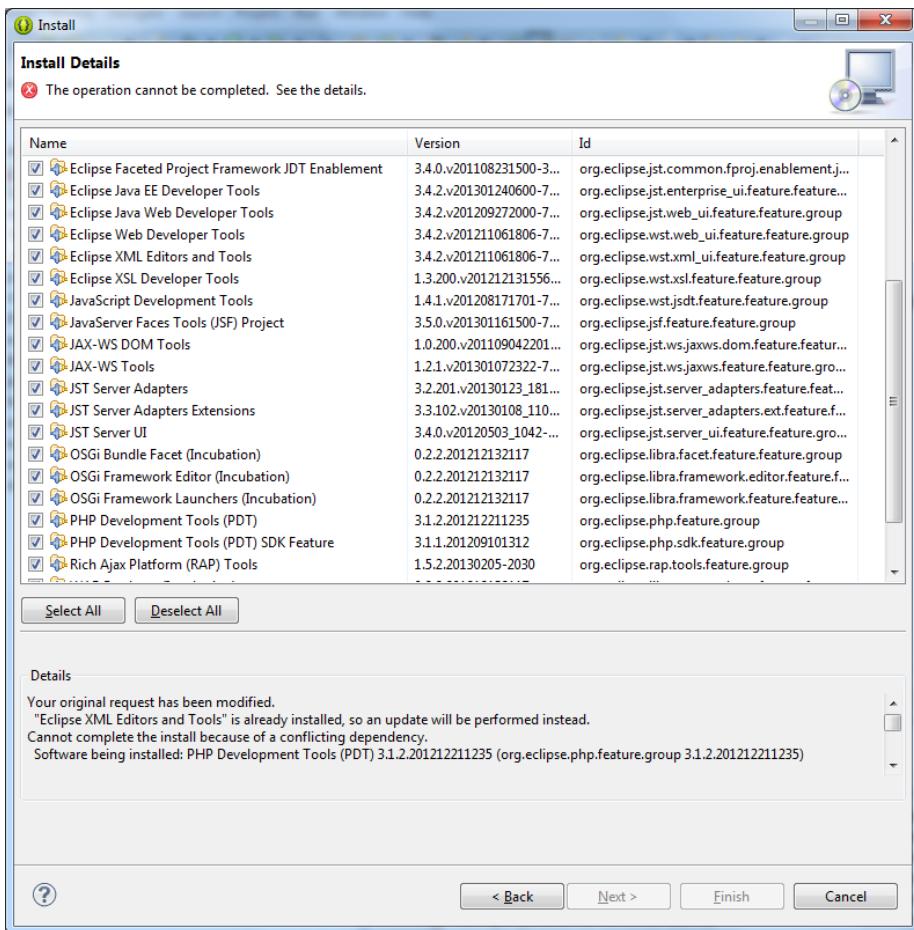
Og klik Web, XML, Java EE and OSG



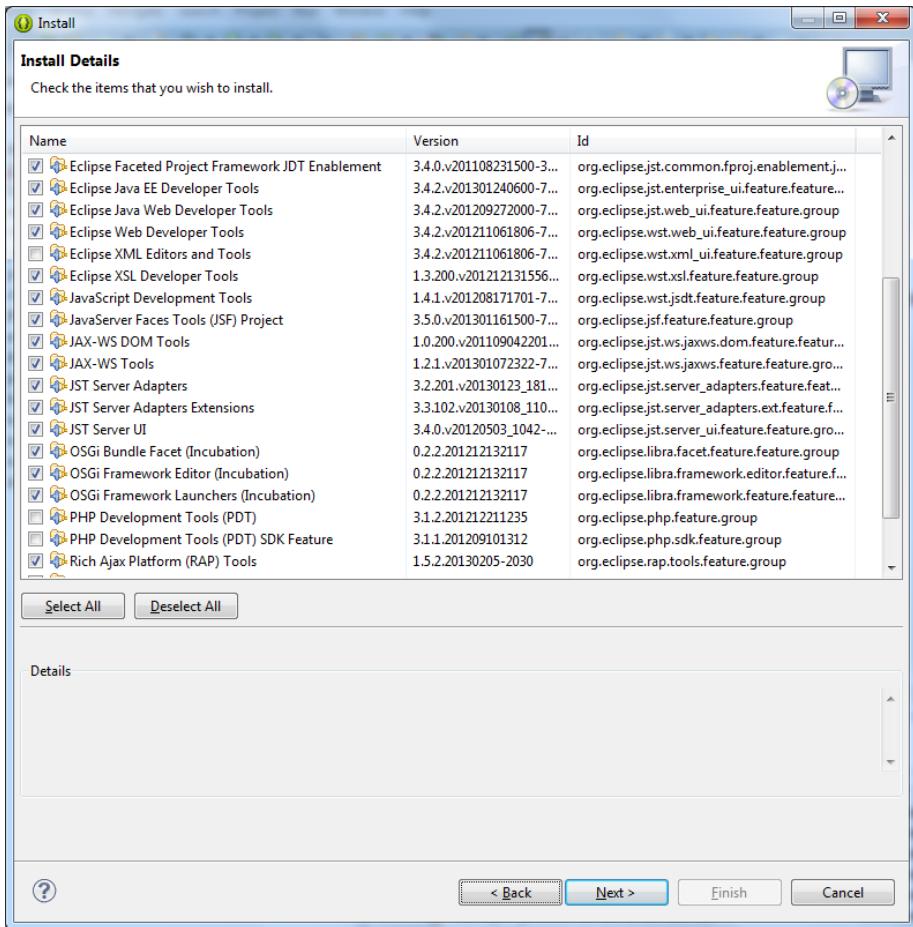
Eller i Neon vælge denne.



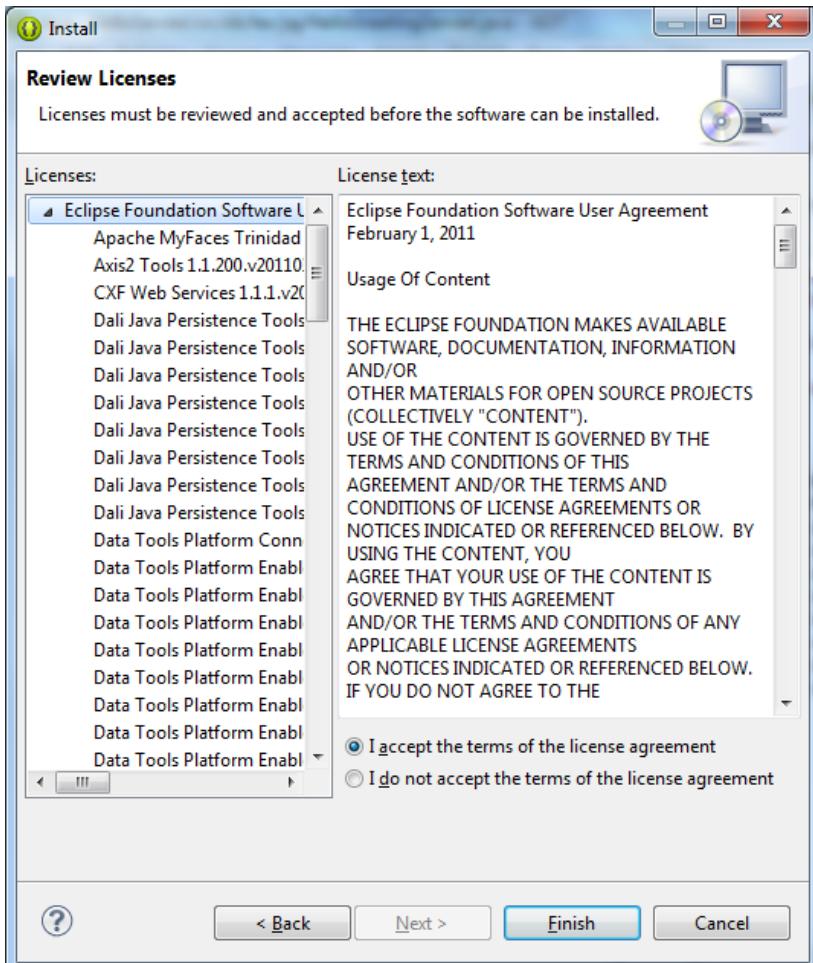
Tryk Next.



Hvis der kommer en fejl pga. at noget er installeret i forvejen, så af-klik dette fra listen.



Tryk Next og Next.



Jeg måtte trykke Cancel og starte forfra for at Finish knappen blev aktiv!!??!!!!!!!

Genstart Eclipse.

Ctrl-X

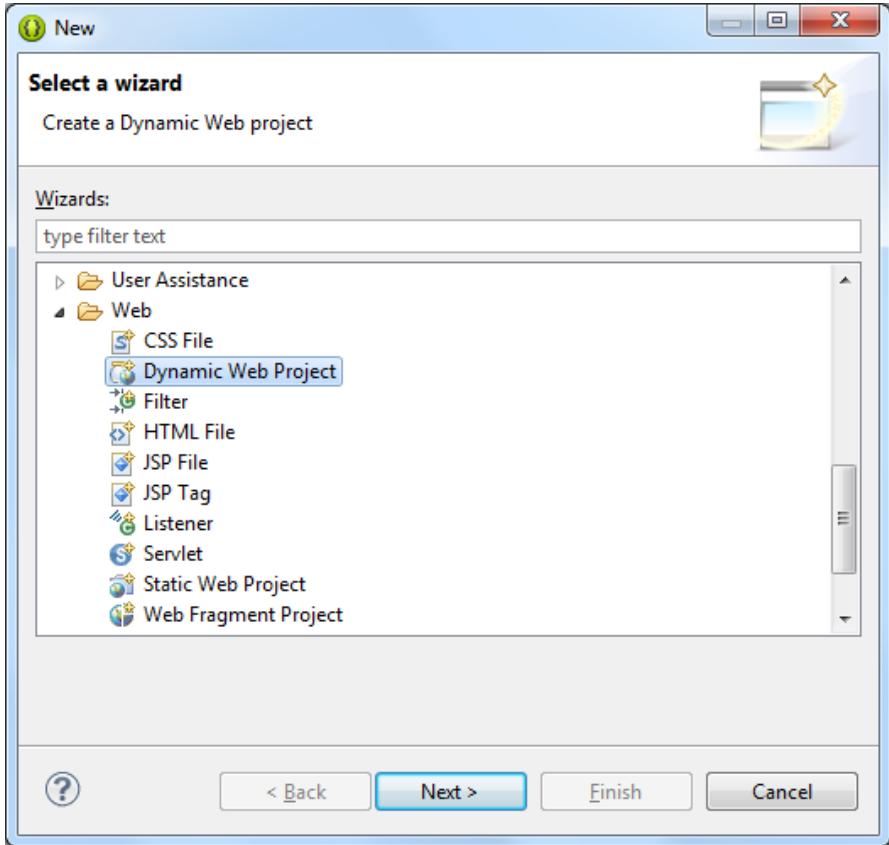
Hvis ctrl-X lukker Eclipse i stedet for at klippe en tekst ud så gør følgende

Gå ind i Window -> Preferences -> General -> Keys.

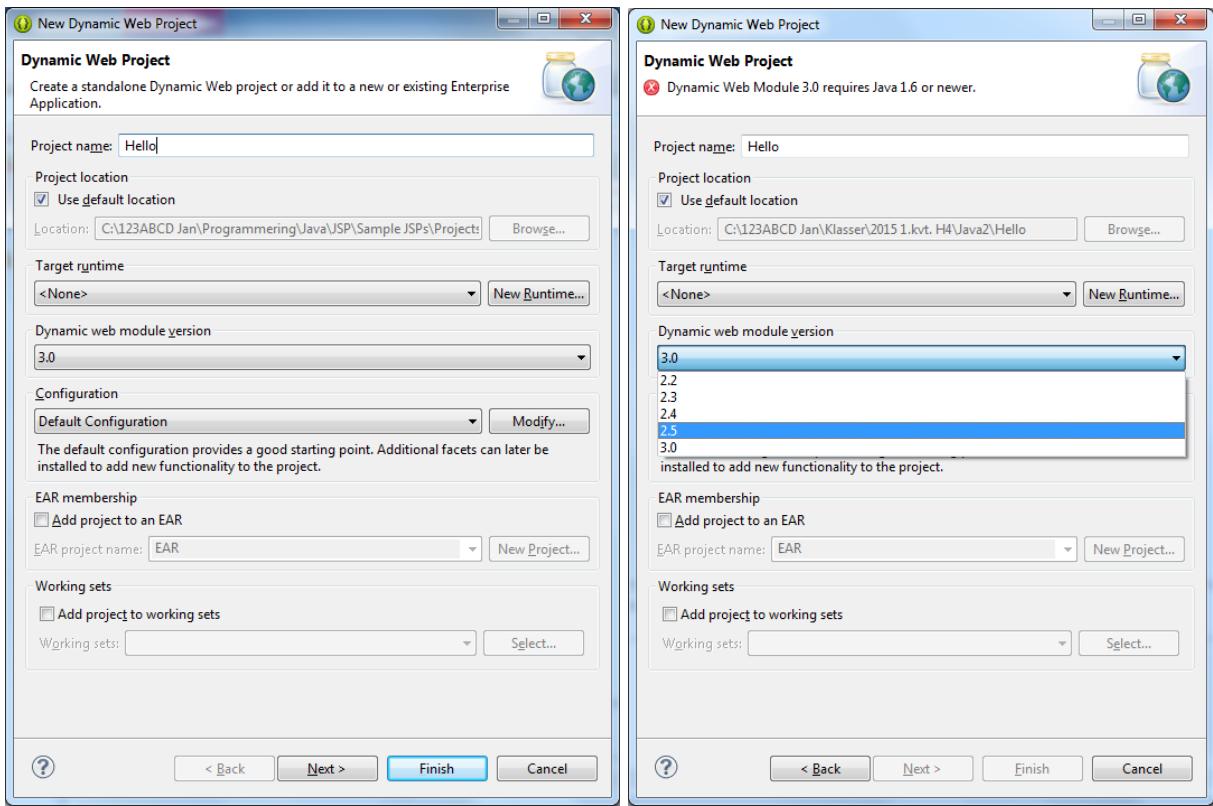
Søg efter 'Cut' command. Binding option skulle vise 'Ctrl+X'. When option ændres fra 'In Dialogs and Windows' til 'Editing Text'.

Opret en Servlet i Eclipse og deploy den på Tomcat

Vælg File | New | Other... | Web | Dynamic Web Project.



og tryk Next.



Indtast Project name: Hello.

Man kan komme ud for at få en fejlmeddeelse Dynamic Web Module 3.0 requires Java 1.6 or newer, selv om man f.eks. har Java 1.8 installeret.

Det kunne løses ved at skifte Dynamic web module version til 2.5 og tilbage til 3.0 igen!?!

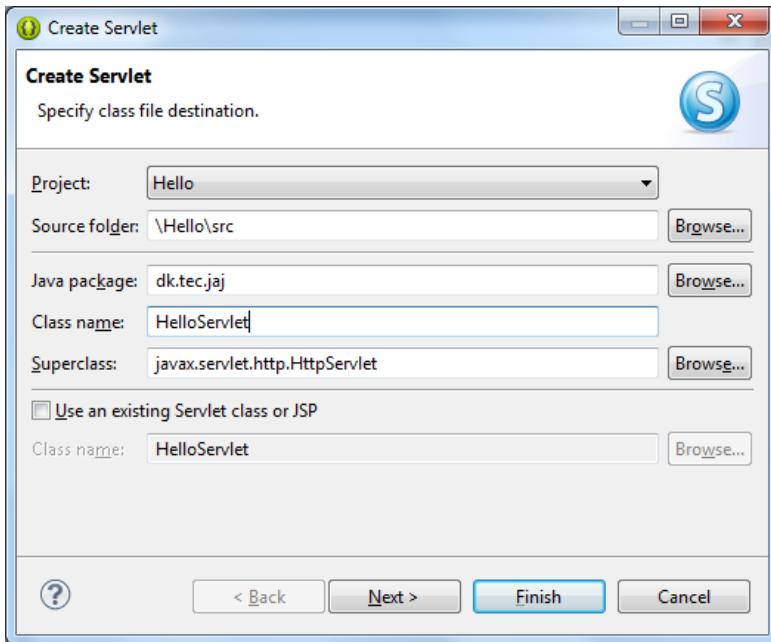
Tryk Finish.

Svar Ja til at åbne Java EE Perspective.

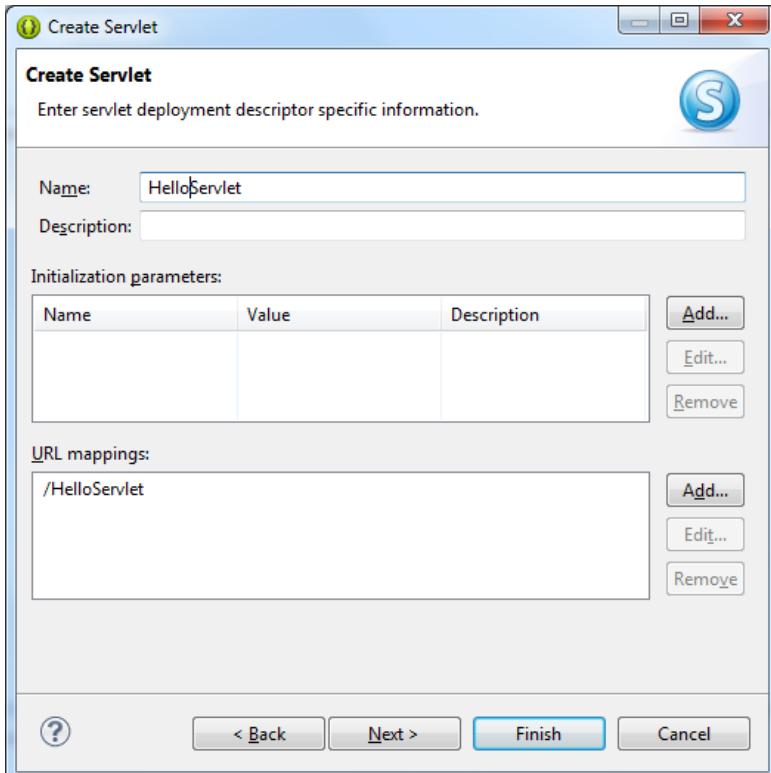
Projektet er oprettet.

Der skal nu tilføjes en Servlet.

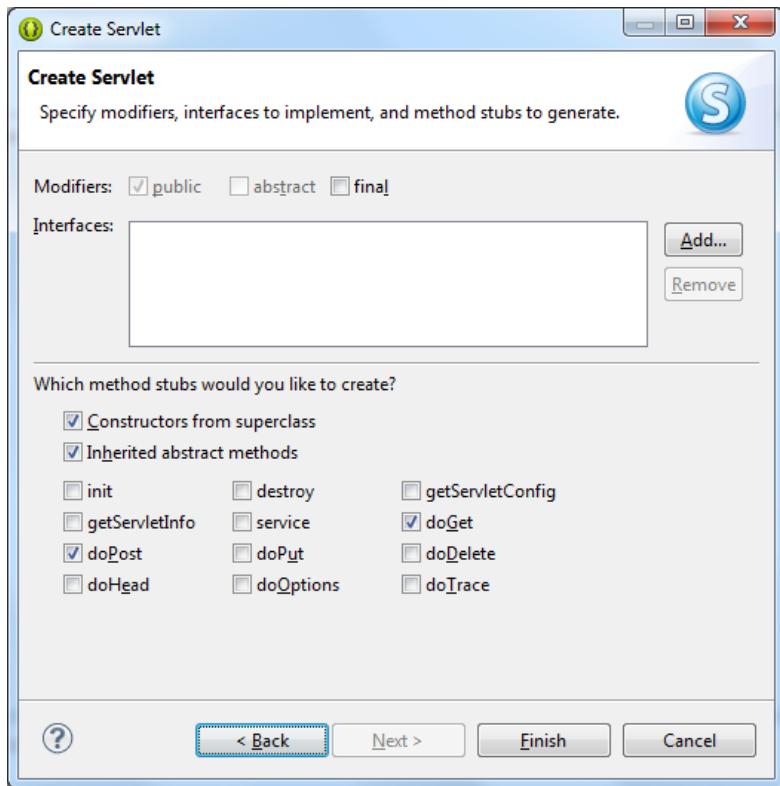
Højre-klik på projektet i Project Explorer og vælg New | Other... | Web | Servlet



Indtast Java package og Class name, og tryk Next.



Her kan der oprettes initialiserings-parametre. Tryk Next.



Her kan vælges hvilke metoder Servletten skal indeholde. Tryk Finish.

Den nye klasses kode kan ses her under.

Der er fjernet en tom constructor og en tom doPost(). Metoden doGet skal have fyldt noget kode i sig.

Der er en masse fejl i koden, da ressourcerne for Servlet ikke er tilføjet Eclipse.

The screenshot shows the Eclipse IDE interface. The Project Explorer view on the left displays a project named 'Hello'. Inside the 'src' folder, there is a package 'dk.tec.jaj' containing a file 'HelloServlet.java'. The Java EE Editor on the right shows the code for 'HelloServlet.java'. The code defines a servlet named 'HelloServlet' that extends 'HttpServlet'. It includes imports for various Java.io and javax.servlet packages, and annotations like @WebServlet. A doGet method is partially implemented.

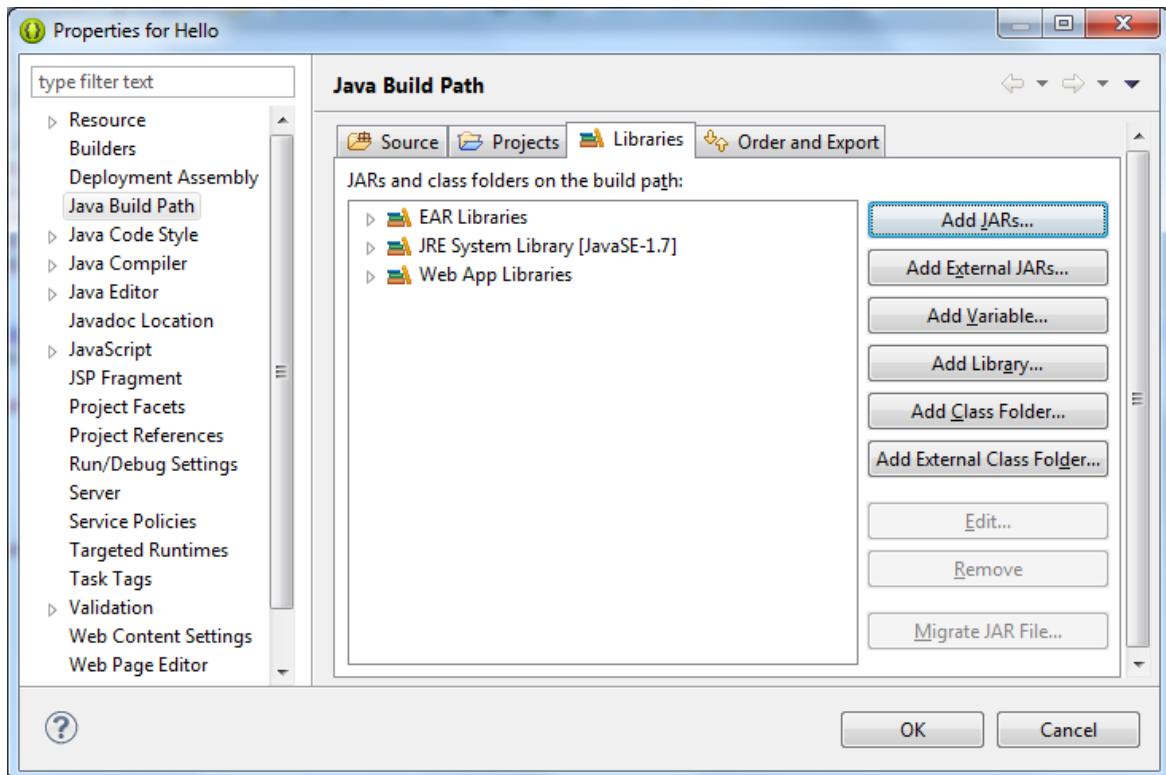
```
package dk.tec.jaj;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;

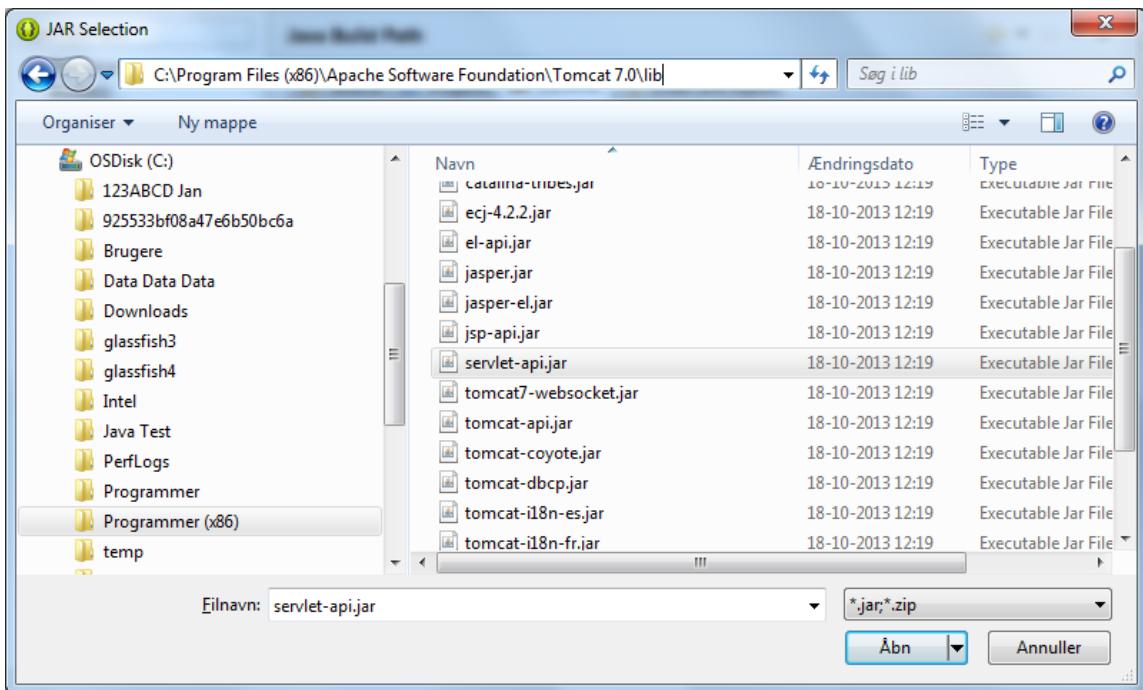
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
        throws ServletException, IOException
    {
    }
}
```

Højre-klik på projektet Hello øverst i Project Explorer og vælg Properties.

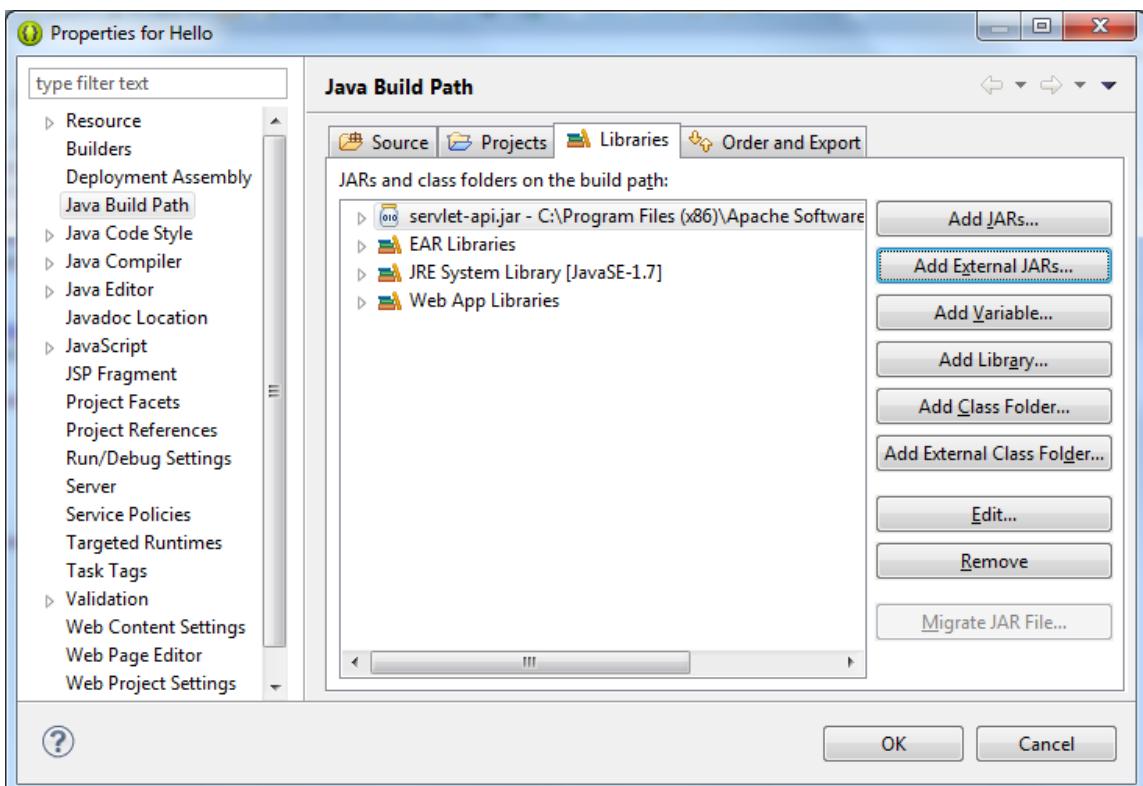


Vælg Java Build Path og Libraries og tryk Add External JARs...

Browse til
C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\lib



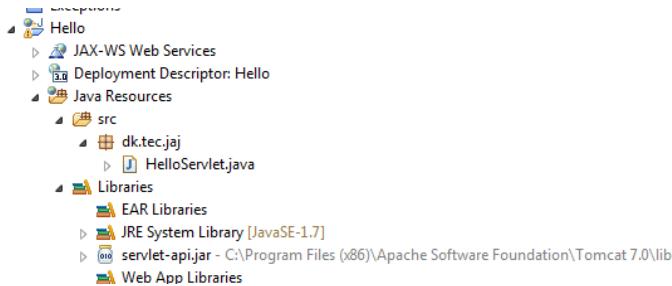
Vælg servlet-api.jar og tryk Åbn



Og tryk OK.

Så er alle compiler-fejlene forsvundet.

I Project Explorer kan servlet-api.jar ses under projektets Java Resources | Libraries.



Tilføj Java-kode sådan at HelloServlet.java kommer til at se ud som vist herunder.

(Det er samme kode som tidligere)

```
package dk.tec.jaj;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
                    throws ServletException, IOException
    {
        Calendar cal = Calendar.getInstance();

        String greeting;
        int hourOfDay = cal.get(Calendar.HOUR_OF_DAY);

        if (hourOfDay < 12)
        {
            greeting = "Good Morning!";
        }
        else if (hourOfDay >= 12 && hourOfDay < 18)
        {
            greeting = "Good Afternoon!";
        }
        else if (hourOfDay >= 18 && hourOfDay < 23)
        {
            greeting = "Good Evening!";
        }
        else
        {
            greeting = "Good Night!";
        }

        response.setContentType("text/html");
    }
}
```

```

    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Hello World</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Hello World!</h1>");
    out.println("<h1>Today it is " + cal.getTime().toString() + "</h1>");
    out.println("<h1>" + greeting + "</h1>");
    out.println("</body>");
    out.println("</html>");
}
}

```

Tilføj under WebContent – WEB-INF filen web.xml med følgende indhold.

```

<?xml version="1.0" encoding="ISO-8859-1"?>

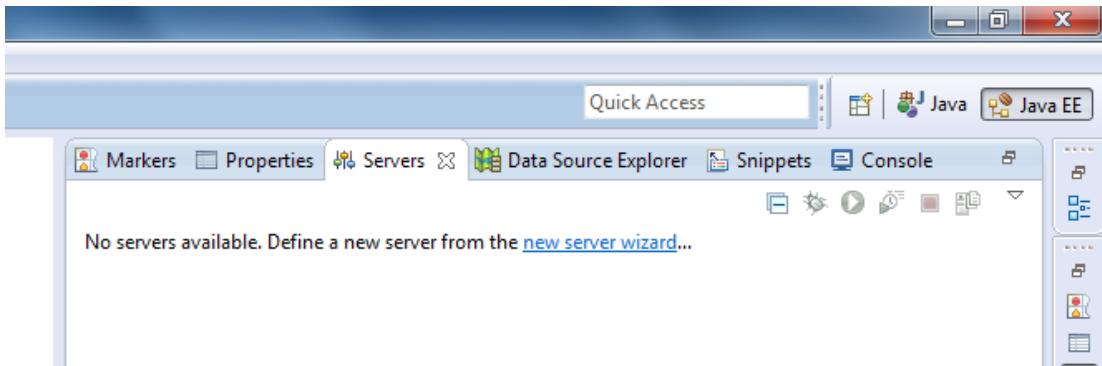
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">

  <display-name>
    Hello World
  </display-name>
  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>
      dk.tec.jaj.HelloServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/Hello.html</url-pattern>
  </servlet-mapping>
</web-app>

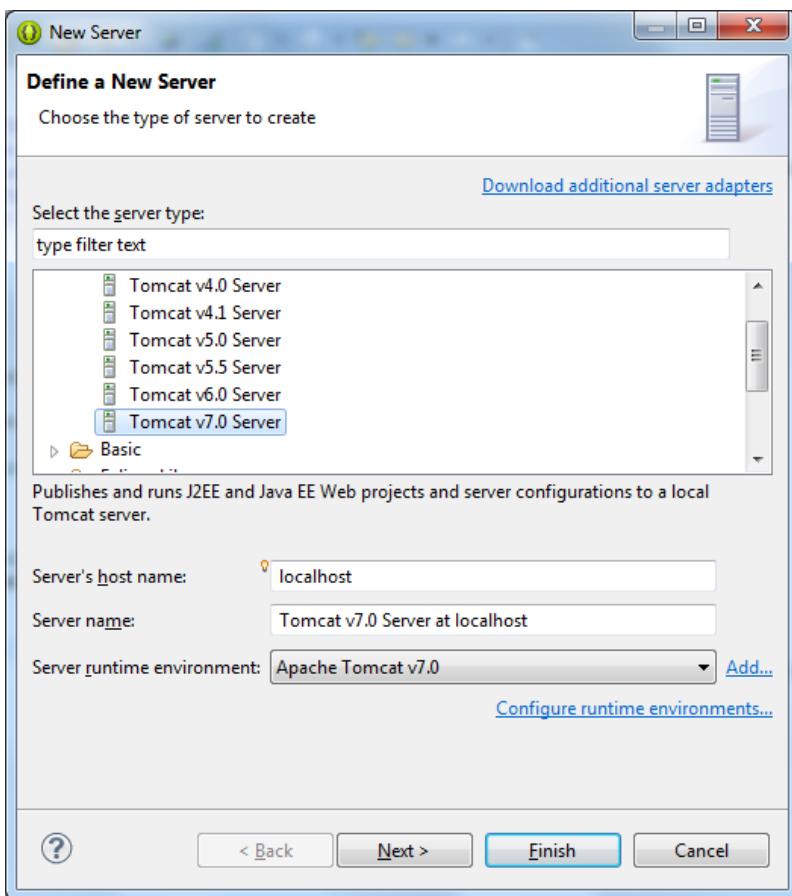
```

Deploy HelloServlet på Tomcat i Eclipse

Vælg i Eclipse Window – Show View – Servers og følgende fremkommer.



Klik på: new server wizard...

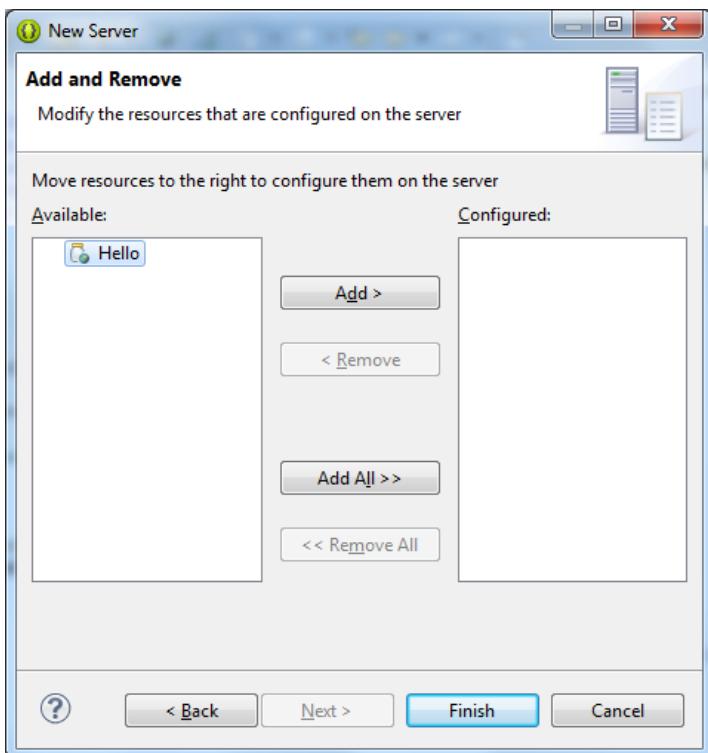


Vælg Apache - Tomcat v7.0 Server eller nyere og tryk Next.

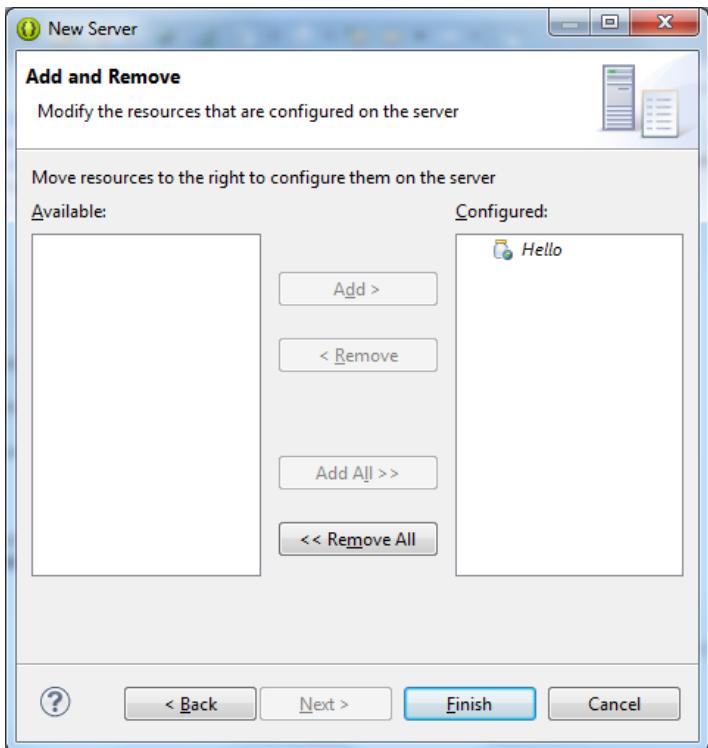
Vælg Tomcat installation directory f.eks.

C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0

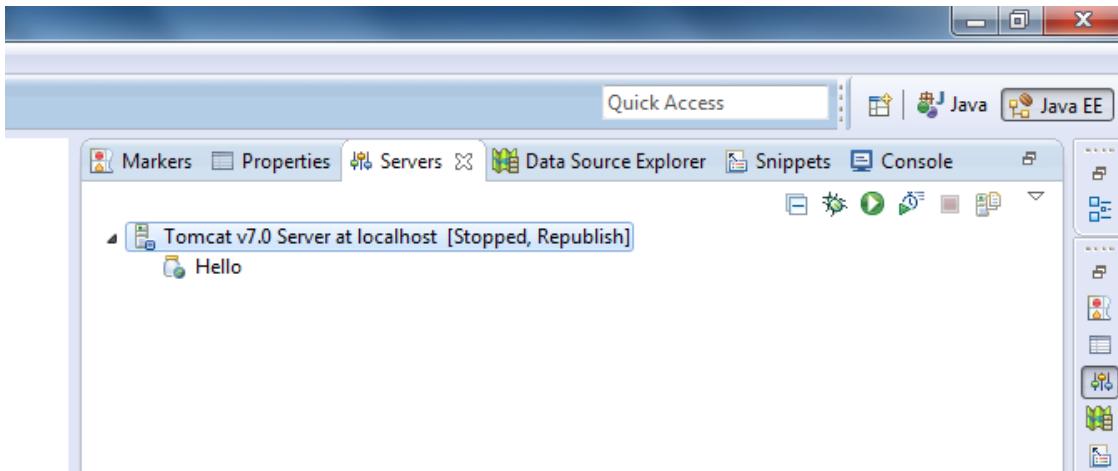
og tryk Next.



Marker projektet Hello i ruden Available og Tryk Add.



Tryk Finsish.



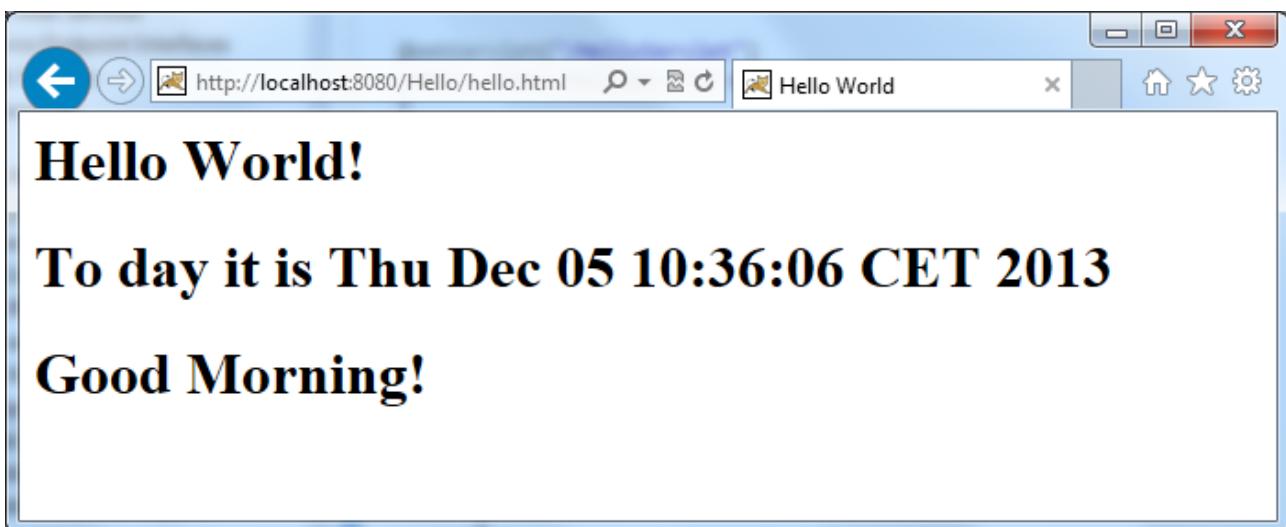
Marker serveren Tomcat v7.0 og tryk på den grøn-hvide start-knap.

Hvis Tomcat kører i forvejen, kommer der en fejlmeddeelse om, at der allerede er en service, der benytter port 8080. Så Tomcat skal stoppes.

Når Tomcat startes her, virker det som om at det er en anden instans af Tomcat, der bliver startet, da den oprindelige Tomcat som startes og stoppes med Tomcat7w.exe, ikke bliver startet når Tomcat startes i Eclipse. Passer fint med fejlmeddelelsen som nævnt ovenfor.

Gå i browseren og angiv URL <http://localhost:8080>Hello/Hello.html>

Altså projektets navn Hello efterfulgt af alias i web.xml /Hello.html.



Nyt projekt og Tomcat

For hvet nyt projekt, der skal afprøves i Eclipse, skal ovenstående gentages.

Højre-klik i Server-vinduet og vælg New – Server, vælg Tomcat v7.0 og tilføj projektet, der skal udføres.

JSP i Eclipse

Det er lidt farligt at rename JSP-filer i Eclipse, da de jo bliver oversat til Java-Servlet-klasser. Hvis dette ikke sker igen efter en rename af en JSP-fil, kan den tilhørende klasse ikke findes.

En løsning kan være bare at ændre en detalje i JSP-filen, så bliver den nemlig automatisk oversat igen.

Output fra JSP

De to følgende metoder write() og print() gør det samme, nemlig at udskrive som det allerførste til browseren. Dvs. at uanset hvor de står i HTML-koden vil udskrifterne komme før alt HTML.

```
<%
    response.getWriter().write("Hallo!");
%>
<%
    response.getWriter().print("Hej!");
%>
```

Metoden out.print() indsætter udskriften der hvor den kaldes i HTML-koden.

```
<%
    out.print("Hejsa!");
%>
```

Kortform udskriften <%= indsætter indholdet af det efterfølgende der hvor konstruktionen er nævnt i HTML-koden. Der kan angives værdier, variable eller metodekald.

```
<%"Hejsa!"%>
```

En liste af indtastede navne opbevaret i session samt genkendelse af tryk på delete-knap for et navn i listen

I det følgende vises hvordan en liste af navne overlever roundtrip ved at knytte listen til en sessions-attribut.

Derefter vises flere metoder til at generere en delete-knap for hvert navn i listen og genkende denne, når der postes til serveren.

Der kan indtastes fornavn og efternavn, og trykkes OK, hvorved navnet indsættes i listen, som gemmes som sessions-attributten Names. Herunder vist resultatet af koden uden Delete-knapper.

The screenshot shows a web browser window with the URL <http://localhost:8080/Names/GetNames.jsp?txtFirstName=Jan&txtLastName=Johansen&btnOK=OK>. The form has fields for 'Fornavn' (Søren) and 'Efternavn' (Rosenbed), with 'OK' and 'Cancel' buttons. Below the form, a table titled 'Udskriv alle navne:' contains three rows: 'Hans Hansen', 'Niels Nielsen', and 'Jan Johansen'. The entire table is highlighted with a blue border.

Der oprettes et nyt Dynamic Web Project og der tilføjes en JSP File, GetNames.jsp. Der skal stadig tilføjes servlet-api.jar, da JSP-filen oversættes til en servlet.

Højre-klik i Server-vinduet og vælg New – Server, vælg Tomcat v7.0 og tilføj projektet, der skal udføres.

Først vises koden for at indtaste nye navne og udskrive listen af navne, men uden Delete-knapper, som vist ovenfor.

Der spørges om session-attributten findes. Hvis ikke, oprettes den med en ArrayList.

Under alle omstændigheder hentes den og der indsættes et nyt navn i den, hvis der er tastet et navn. Arraylisten behøver ikke at blive lagt tilbage i sessionen, da det jo er en kopi af referencen, der benyttes.

Til sidst udskrives alle navnene fra Arraylisten.

GetNames.jsp:

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-8">
    <title>Insert title here</title>
    <style>
        td {color:blue; padding:10px;}
    </style>
```

```

</head>

<body style="font-family:Arial">
<%
if(session.getAttribute("Names") == null)
{
    session.setAttribute("Names", new ArrayList<String>());
}
ArrayList<String> names = (ArrayList<String>)session.getAttribute("Names");

if(request.getParameter("btnOK") != null)
{
    String firstName = request.getParameter("txtFirstName");
    String lastName = request.getParameter("txtLastName");
    if(firstName != null && !firstName.equals(""))
        && lastName != null && !lastName.equals(""))
    {
        names.add(firstName + " " + lastName);
    }
}
%>

<form method="GET" action="GetNames.jsp" >
    Fornavn: &ampnbsp<input type="text" name="txtFirstName" />&ampnbsp
    Efternavn: &ampnbsp<input type="text" name="txtLastName" />&ampnbsp
    <input type="submit" name="btnOK" value="OK" style="width:80px" />&ampnbsp
    <input type="reset" name="btnCancel" value="Cancel" style="width:80px"/>
</form>

<br />

Udskriv alle navne:

<table border="1">

<%
for(int i = 0; i < names.size(); i++ )
{%
    <tr>
        <td><%=names.get(i)%></td>
    </tr>
%}>
</table>

</body>
</html>

```

Opgave 1

Skriv et program hvor brugeren kan indtaste et tal. Lad alle indtastede tal blive akkumuleret (summen af alle tal) og denne værdi hele tiden udskrevet.

Opgave 2

Skriv et program med to indtastningsfelter hvor der kan skrives et tal x, der skal opløftes i et andet tal y.

F.eks. x = 5 og y = 3 så er $x^y = 125$

Når der trykkes på knappen "=" skal det beregnes og resultatet udskrives. Samtidig skal de to indtastede tal blive stående.

Delete-knapper til at slette elementer i listen

Der vendes nu tilbage til projektet med indtastning af navne.

I det følgende tilføjes en Delete-knap til hvert navn i udskriften, så det aktuelle navn kan slettes.

Der vises 5 forskellige måder at gøre dette på.

De først 4 måder resulterer i dette udseende.

The screenshot shows a web page with the URL http://localhost:8080/Names/GetNames.jsp?txtFirstName=S%F8ren&txtLastName=Rosenbed&btnOK=OK. At the top, there are input fields for 'Fornavn' and 'Efternavn' with 'OK' and 'Cancel' buttons. Below this, a table titled 'Udskriv alle navne:' lists four names: Hans Hansen, Niels Nielsen, Jan Johansen, and Søren Rosenbed. Each name has a 'Delete' button next to it. The table has a border of 1.

Udskriv alle navne:	
Hans Hansen	<input type="button" value="Delete"/>
Niels Nielsen	<input type="button" value="Delete"/>
Jan Johansen	<input type="button" value="Delete"/>
Søren Rosenbed	<input type="button" value="Delete"/>

1. Navn og delete-knap vises i hver sin form

Når der oprettes en form til hvert navn og knap, kan der tilføjes en hidden tag, med samme navn i alle forme, og med en value der består af det nummer som elementet har i listen.

Når der submittes ved at trykke på delete-knappen, sendes parameteren "delNr" med og dens værdi kan læses og bruges til at udpege nummeret der skal slettes.

Koden efter "Udskriv alle navne:" ændres/tilføjes så det ser ud som følger.

GetNames.jsp:

```
...
Udskriv alle navne:  

<%  

    if(request.getParameter("btnDelete") != null)  

    {  

        int delNr = Integer.parseInt(request.getParameter("delNr"));  

        names.remove(delNr);  

        response.sendRedirect("GetNames.jsp");  

    }  

%>  





```

```

<td>
    <form style="margin:0px">
        <input type="submit" name="btnDelete" value="Delete"/>
        <input type="hidden" name="delNr" value="<%>i%<%>"/>
    </form>
</td>
</tr>
<%}>
</table>
</body>
</html>

```

2. Navne og tilhørende delete-knapper med fortløbende navne, vises i samme form (Eks. 1)

Navnene på delete-knapperne tilføjes nummeret svarende til elementet i listen. Knapperne kommer altså til at hedde btnDelete0, btnDelete1 osv.

Ved modtagelsen af submit, løbes alle navnemulighederne for btnDeleteNN igennem, og der spørges for hvert navn, om der er en parameter af dette navn. Hvis det er tilfældet bruges det aktuelle nummer til at slette det tilsvarende element.

GetNames.jsp:

```

...
    Udskriv alle navne:

<%
for(int delNr = 0; delNr < names.size(); delNr++)
{
    if(request.getParameter("btnDelete" + delNr) != null)
    {
        names.remove(delNr);
        response.sendRedirect("GetNames.jsp");
    }
}
%>

<form style="margin:0px">
    <table border="1">
        <%
        for(int i = 0; i < names.size(); i++ )
        {%
        <tr>
            <td><%=names.get(i)%></td>
            <td>
                <input type="submit" name="btnDelete<%=i%>" value="Delete"/>
            </td>
        </tr>
        <%}>
    </table>
</form>
</body>
</html>

```

3. Navne og tilhørende delete-knapper med fortløbende navne, vises i samme form (Eks. 2)

Denne er magen til den forrige, blot at bestemmelsen af den trykkede delete-knap er anderledes.

Navnene på delete-knapperne tilføjes nummeret svarende til elementet i listen. Knapperne kommer altså til at hedde btnDelete0, btnDelete1 osv.

Ved modtagelsen af submit, løbes alle modtagne parametre igennem, og der spørges for hver parameter, om dens navn begynder med "btnDelete". Hvis dette er tilfældet, hentes resten af navnet, som er nummeret på elementet der slettes.

GetNames.jsp:

```
...
Udskriv alle navne:  
  
<%  
    java.util.Enumeration<String> parNames = request.getParameterNames();  
    while(parNames.hasMoreElements())  
    {   //Er der en parameter der hedder "btnDeleteNN" så isoler NN og slet denne  
        String parName = parNames.nextElement();  
        if(parName.length() > "btnDelete".length()  
            && parName.substring(0, "btnDelete".length()).equals("btnDelete"))  
        {  
            int delNr = Integer.parseInt(parName.substring("btnDelete".length()));  
            names.remove(delNr);  
        }  
    }  
%>  
  
<form style="margin:0px">  
    <table border="1">  
        <%  
        for(int i = 0; i < names.size(); i++ )  
        {  
            <tr>  
                <td><%=names.get(i)%></td>  
                <td>  
                    <input type="submit" name="btnDelete<%=i%>" value="Delete"/>  
                </td>  
            </tr>  
        <% }%>  
    </table>  
    </form>  
</body>  
</html>
```

4. Navne med tilhørende delete-knapper, som kalder javascript funktion, vises i samme form.

Delete-knapperne kalder hver den samme javascript funktion onMySubmit() og sender det aktuelle nummer som parameter til funktionen. Funktionen genererer et nyt form-element med elementnummeret, som derved sendes som parameter ved postningen.

Hvis GetNames.jsp ligger i mappen WebContent\jpsps og MyJavaScript.js ligger i mappen WebContent\scripts, vil script-tagget se sådan ud, og den placeres i head-tagget i GetNames.jsp.

```
<script src="../scripts/MyJavaScript.js" type="text/javascript" ></script>
```

Filen MyJavaScript.js:

```
function onMySubmit(delElementNr)  
{  
    //alert("onMyClick called");  
  
    var f = document.forms["deleteForm"];
```

```

var i = document.createElement("input");
i.setAttribute('type','text');
i.setAttribute('name','delNr');
i.setAttribute('value', delElementNr);

f.appendChild(i);

f.submit();
return false;
}

```

GetNames.jsp:

...

Udskriv alle navne:

```

<%
String strDelNr = request.getParameter("delNr");
if(strDelNr !=null)
{
    int delNr = Integer.parseInt(strDelNr);
    names.remove(delNr);
}
%>

<form id="deleteForm">
<table border="1">
<%
for(int i = 0; i < names.size(); i++)
{%
<tr>
    <td> <%=names.get(i) %> </td>
    <td>
        <input type="submit" name="btnDel" onclick="return onMySubmit(<%=i%>)" value="Delete"/>
    </td>
</tr>
%}
</table>
</form>
</body>
</html>

```

5. Alle navne med tilhørende delete-knapper, der hedder det samme, men med nummeret i knappens tekst, vist i samme form

The screenshot shows a web browser window with the URL <http://localhost:8080/Names/GetNames.jsp?txtFirstName=Jan&txtLastName=Johansen&btnOK=OK>. At the top, there are back and forward navigation buttons, a search icon, and the URL. Below the address bar, there are two input fields: 'Fornavn:' and 'Efternavn:', both containing the text 'Jan'. To the right of these fields are 'OK' and 'Cancel' buttons. The main content area is titled 'Udskriv alle navne:' and contains a table with three rows. Each row has two columns: a name and a delete button. The names are 'Hans Hansen', 'Flemming Sørensen', and 'Jan Johansen'. The delete buttons are labeled 'Delete (0)', 'Delete (1)', and 'Delete (2)' respectively.

Alle knapper hedder btnDelete, men de har hver deres tekst med nummeret på elementet, der skal slettes.

Ved modtagelsen spørges om der er trykket på en btnDelete, og i givet fald hentes nummeret ud af dens tekst, og det tilsvarende element slettes.

GetNames.jsp:

...

Udskriv alle navne:

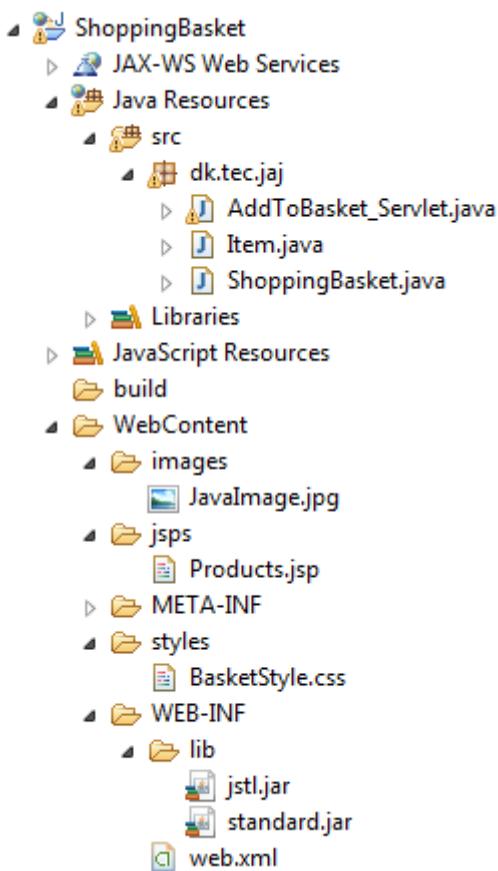
```
<%  
String delete = request.getParameter("btnDelete");  
if(delete != null)  
{  
    int delNr = Integer.parseInt(delete.substring(8, delete.length() - 1));  
    names.remove(delNr);  
}  
%>  
  
<form style="margin:0px">  
    <table border="1">  
        <%  
        for(int i = 0; i < names.size(); i++ )  
        {  
            <tr>  
                <td><%=names.get(i)%></td>  
                <td>  
                    <input type="submit" name="btnDelete" value="Delete (<%=i%>)"/>  
                </td>  
            </tr>  
        }%>  
    </table>  
</form>  
</body>  
</html>
```

Shoppingbasket projekt med session

The screenshot shows a Java application window titled "Java Book Store". The URL in the address bar is "http://localhost:8080/ShoppingBasket/jsp/Products.jsp". The window contains a logo for "Java CORE JAVA TRAINING" and a heading "Welcome to the Java Book Store". Below this, there is a table titled "Select product:" listing three items:

Product Name	Price	Actions
Apache Tomcat 7	\$32.50	<input type="button" value="Add to basket"/>
Java Server Pages	\$17.85	<input type="button" value="Add to basket"/>
Android Development for Beginners	\$24.90	<input type="button" value="Add to basket"/>

At the bottom of the window, a message says "Your basket is empty".



Højreklik på projektet i Project Explorer og tilføj en ny klasse til projektet af navnet Item og dermed filen Item.java.

Lad den have følgende indhold.

```
package dk.tec.jaj;

import java.io.Serializable;

public class Item implements Serializable
{
    private static final long serialVersionUID = 1L;

    private String name;
    private double price;

    public Item(String name, double price)
    {
        this.name = name;
        this.price = price;
    }

    public String getName()
    {
        return this.name;
    }

    public double getPrice()
    {
        return this.price;
    }
}
```

```
    }  
}
```

Hvis package dk.tec.jaj ikke er oprettet, kommer der en fejl, hvor det foreslås at oprette denne package og flytte filen hen i denne.

Tilføj klassen ShoppingBasket ved at højre-klikke på package-mappen dk.tec.jaj i Projekt Explorer.

Giv den følgende indhold.

```
package dk.tec.jaj;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
  
public class ShoppingBasket implements Serializable  
{  
    private static final long serialVersionUID = 1L;  
  
    private List<Item> items = new ArrayList<Item>();  
    private double totalValue = 0.0;  
  
    public void addToBasket(Item item)  
    {  
        items.add(item);  
        totalValue += item.getPrice();  
    }  
  
    public List<Item> getItems()  
    {  
        return items;  
    }  
  
    public double getTotalValue()  
    {  
        return totalValue;  
    }  
}
```

Tilføj en Servlet til projektet ved at højre-klikke på package dk.tec.jaj og vælge New – Servlet af navnet AddToBasket_Servlet.java og giv den følgende indhold.

```
package dk.tec.jaj;  
  
import java.io.IOException;  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
  
@WebServlet("/AddToBasket")  
public class AddToBasket_Servlet extends HttpServlet  
{  
    private static final long serialVersionUID = 1L;
```

```

protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException
{
    HttpSession session = request.getSession(true);
    ShoppingBasket basket = (ShoppingBasket)session.getAttribute("SHOPPING_BASKET");
    if(basket == null)
    {
        basket = new ShoppingBasket();
        session.setAttribute("SHOPPING_BASKET", basket);
    }

    String name = request.getParameter("productName");
    double price = Double.parseDouble(request.getParameter("price"));
    Item item = new Item(name, price);
    basket.addToBasket(item);

    response.sendRedirect("/ShoppingBasket/jsp/Products.jsp");
}
}

```

Læg mærke til at session trækkes ud af request'et for at have samme session som er startet i Products.jsp.

Annotation

@WebServlet("/AddToBasket")
gør at servetten kan findes uden at det er angivet i web.xml, og at web.xml slet ikke får virkning mht. til den aktuelle servlet.

Der kan altså vælges imellem den annotation eller følgende web.xml.

Tilføj filen web.xml ved at højre-klikke på WebContent\WEB-INF og vælge New – XML File og giv den følgende indhold.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
          version="3.0">
    <display-name>Shopping Basket</display-name>
    <description>Http Session Demo</description>

    <servlet>
        <servlet-name>AddToBasket</servlet-name>
        <servlet-class>
            dk.tec.jaj.AddToBasket_Servlet
        </servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>AddToBasket</servlet-name>
        <url-pattern>/AddToBasket</url-pattern>
    </servlet-mapping>
</web-app>

```

Hvis web.xml ikke er korrekt, kan det ske at Apache Tomcat ikke kan starte.

Tilføj mappen styles ved at højre-klikke på WebContent og vælge New – Folder.

Tilføj filen BasketStyle.css ved at højre-klikke på mappen styles og vælge New - Web - CSS File med følgende indhold.

```
@CHARSET "ISO-8859-1";  
  
td{padding-left:20px}
```

Tilføj mappen images til WebContent og læg her et billede af navnet *JavaImage.jpg*

Tilføj mappen jsps til WebContent.

Højreklik på denne og tilføj JSP-filen Products.jsp med følgende indhold.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
  
<%@page import="dk.tec.jaj.Item"%>  
<%@page import="dk.tec.jaj.ShoppingBasket"%>  
  
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">  
  
<html>  
  <head>  
    <title>Java Book Store</title>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>  
    <link rel="stylesheet" href="/ShoppingBasket/styles/BasketStyle.css"  
          type="text/css"/>  
  </head>  
  
  <body>  
      
    <br/><br/>  
    <h2>Welcome to the Java Book Store </h2>  
    <br/>  
    <b>Select product: </b><br/><br/>  
    <table style="border-spacing: 10px">  
      <tr>  
        <td><b>Product Name</b></td>  
        <td><b>Price</b></td>  
        <td><b>Actions</b></td>  
      </tr>  
      <tr>  
        <td>Apache Tomcat 7</td>  
        <td>$32.50</td>  
        <td>  
          <form action="/ShoppingBasket/AddToBasket" method="POST">  
            <input type="hidden" name="productName" value="Apache Tomcat 7"/>  
            <input type="hidden" name="price" value="32.50"/>  
            <input type="submit" value="Add to basket" />  
          </form>  
        </td>  
      </tr>  
      <tr>  
        <td>Java Server Pages</td>  
        <td>$17.85</td>  
        <td>
```

```

        <form action="/ShoppingBasket/AddToBasket" method="POST">
            <input type="hidden" name="productName" value="Java Server Pages"/>
            <input type="hidden" name="price" value="17.85"/>
            <input type="submit" value="Add to basket" />
        </form>
    </td>
</tr>
<tr>
    <td>Android Development for Beginners</td>
    <td>$24.90</td>
    <td>
        <form action="/ShoppingBasket/AddToBasket" method="POST">
            <input type="hidden" name="productName"
                   value="Android Development for Beginners"/>
            <input type="hidden" name="price" value="24.90"/>
            <input type="submit" value="Add to basket" />
        </form>
    </td>
</tr>
</table>

<!-- Vis Shopping Basket med <u>indlejret Java-kode -->

<%
ShoppingBasket basket = (ShoppingBasket)session.getAttribute("SHOPPING_BASKET");
if (basket != null && basket.getItems().size() > 0)
{%
<br/>
<b>Your shopping cart:</b><br /><br />
<table>
    <tr>
        <td><b>Product</b></td>
        <td><b>Price</b></td>
    </tr>

    <%
for(Item i : basket.getItems())
{
    String price = (new DecimalFormat("#.00")).format(i.getPrice());
%>
    <tr>
        <td><%=i.getName()%></td>
        <td>$<%=price%></td>
    </tr>
}%
    <%} %>

    <tr>
        <td style="padding-top:10px"><b>Total:</b></td>
        <td style="padding-top:10px">
            $<%(new DecimalFormat("#.00")).format(basket.getTotalValue())%>
        </td>
    </tr>
</table>
%}
else
{%
    <b>Your basket is empty</b>
%}%
</body>

```

```
</html>
```

De to øverste linier er til brug for JSTL, som brugt nedenfor til at udskifte noget af koden.

Disse vil give fejlmeddelelser.

Download jstl-1.2.jar fra

<http://download.java.net/maven/1/jstl/jars/jstl-1.2.jar>

og læg den i WEB-INF\lib

The screenshot shows a web browser window with the URL `http://localhost:8080/ShoppingBasket/jsp/Products.jsp` in the address bar. The title bar says "Java Book Store". The page content includes the Java logo and the text "CORE JAVA TRAINING". Below this, a heading "Welcome to the Java Book Store" is displayed. A section titled "Select product:" lists four items with their prices and "Add to basket" buttons:

Product Name	Price	Actions
Apache Tomcat 7	\$32.50	Add to basket
Java Server Pages	\$17.85	Add to basket
Android Development for Beginners	\$24.90	Add to basket

Below this, a section titled "Your shopping cart:" shows the same four items with their prices, indicating they have been added to the cart:

Product	Price
Apache Tomcat 7	\$32,50
Java Server Pages	\$17,85
Apache Tomcat 7	\$32,50
Android Development for Beginners	\$24,90
Total:	\$107,75

JSTL i stedet for indlejret Java-kode i Products.jsp

```
<!-- Vis Shopping Basket med JSTL -->

<c:set var="basket" value="${sessionScope.SHOPPING_BASKET}" />
<c:if test="${basket != null && not empty basket.items}">
    <br/>
    <b>Your shopping cart:</b><br /><br />
    <table>
        <tr>
            <td><b>Product</b></td>
            <td><b>Price</b></td>
        </tr>
        <c:forEach items="${basket.items}" var="item">
            <tr>
                <td><c:out value="${item.name}" /></td>
                <td>$<fmt:formatNumber pattern="0.00" value="${item.price}" /></td>
            </tr>
        </c:forEach>
        <tr>
            <td style="padding-top:10px"><b>Total:</b></td>
            <td style="padding-top:10px">
                $<fmt:formatNumber pattern="0.00" value="${basket.totalValue}" />
            </td>
        </tr>
    </table>
</c:if>
<c:if test="${basket ==null || empty basket.items }">
    <b>Your basket is empty</b>
</c:if>
</body>
</html>
```

Parametre



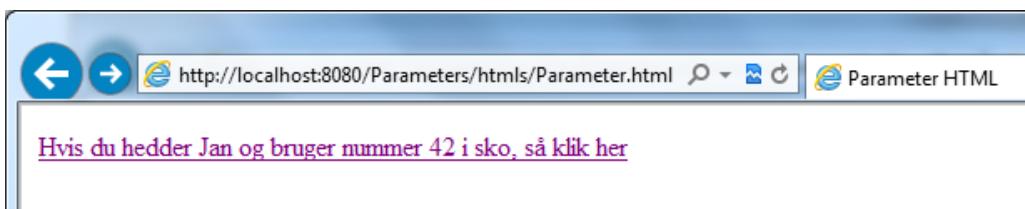
Filen ShowParameter.jsp:

```
<html>
  <head>
    <title>Parameters</title>
  </head>
  <body>
    <%
      String name = request.getParameter("name");
      String shoe = request.getParameter("shoe");
      out.print("Hej " + name + "! <br />Dit skonummer er " + shoe);
    %>
  </body>
</html>
```



Filen Parameter.html:

```
<html>
  <head>
    <title>Parameter HTML</title>
  </head>
  <body>
    <a href="http://localhost:8080/Parameters/jsp/ShowParameter.jsp?name=Jan&shoe=42">
      Hvis du hedder Jan og bruger nummer 42 i sko, så klik her
    </a>
  </body>
</html>
```

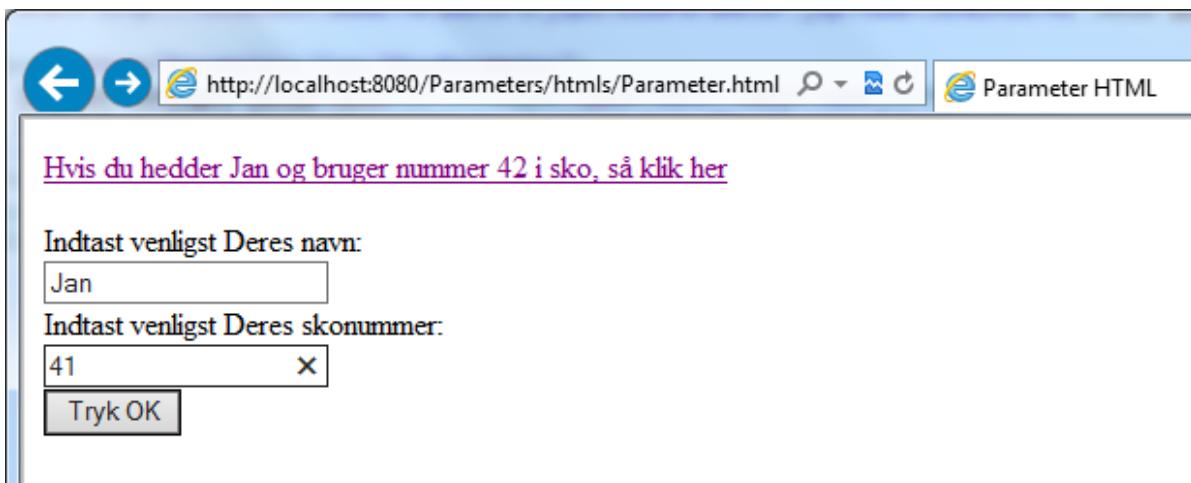


Og resultatet er det samme som før.

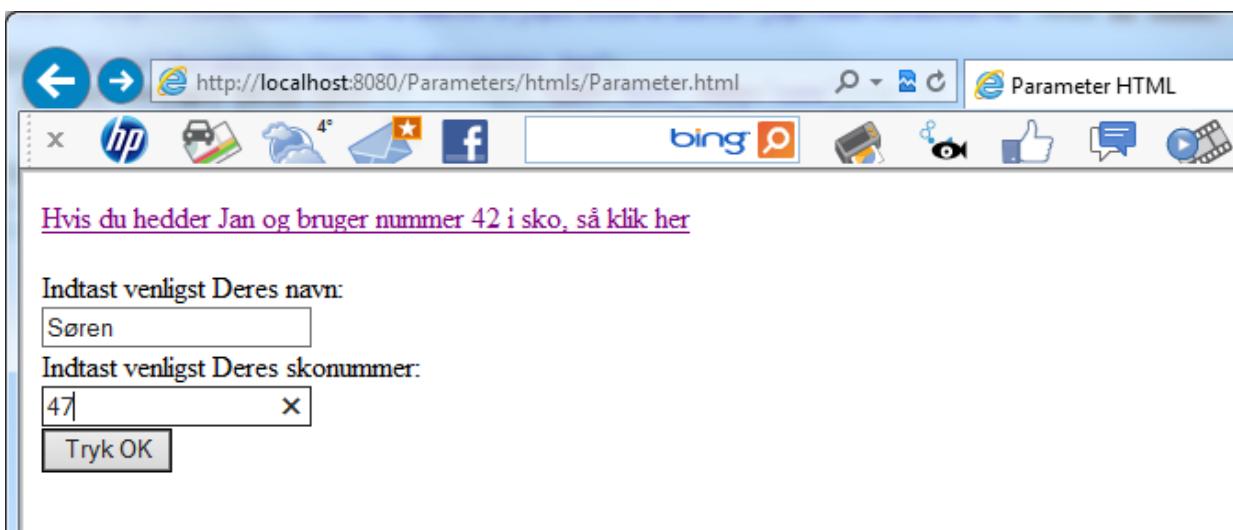


```
<html>
<head>
<title>Parameter HTML</title>
</head>
<body>
    <a href="http://localhost:8080/Parameters/jsp/ShowParameter.jsp?name=Jan&shoe=42">
        Hvis du hedder Jan og bruger nummer 42 i sko, så klik her
    </a>

    <form action="/Parameters/jsp/ShowParameter.jsp">
        Indtast venligst Deres navn:<br /><input type="text" name="name"/><br />
        Indtast venligst Deres skonummer:<br /><input type="text" name="shoe"/><br />
        <input type="submit" name="btnOK" value="Tryk OK" />
    </form>
</body>
</html>
```



Læg mærke til at knappen også bliver sendt som parameteren btnOK og med værdien Tryk+OK, som er dens value og teksten på den. Der må ikke sendes spaces i en URL, derfor er skiftes disse ud med tegnet +.



Læg mærke til at ø bliver kodet til %F8.



Hvis tegnet + indgår i URL, bliver denne kodet også. Således bliver 47+2 kodet til 47%2B2.

Typer af formularfelter

Indtast dit navn: (Input Text)

Indtast din kode: (Input Password)

Skriv en historie: (TextArea)

Jeg er verdensmester i at
 banke et program sammen.

Hvad foretrækker du at programmere i? (RadioButtons)

Java C++ PHP

Hvad kan du programmere i? (Checkboxes)

Java C++ PHP

Hvilken bil har du? (Select)

Rover

Hvilke biler kunne du tænke dig? (Select multiple)

Audi
BMW
 Rover
 Folkevogn



```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head><title>Typer af formularfelter</title></head>
  <body>
    <h1>Typer af formularfelter</h1>
    <form action="/Parameters/jsp/ShowParameters.jsp" method="get">

      <p>Indtast dit navn: (Input Text)
      <input type="text" name="name" value="Hans Peter Jensen" size="20"/>
      <br>Indtast din kode: (Input Password)
      <input type="password" name="code" value="Abcdef123" size="20"/>
      <input type="hidden" name="id" value="7913"/>
    </p>

      <p>Skriv en historie: (TextArea) <br />
      <textarea name="history" rows="2" cols="30">

```

```

    Jeg er verdensmester i at banke et program sammen.
</textarea>
</p>

<p>Hvad foretrækker du at programmere i? (RadioButtons)<br />
<input type="radio" name="preferProg" value="Java"/>Java
<input type="radio" name="preferProg" value="C++"/>C++
<input type="radio" name="preferProg" value="PHP" checked="checked"/>PHP
</p>

<p>Hvad kan du programmere i? (Checkboxes)<br />
<input type="checkbox" name="canProg" value="Java" checked="checked"/>Java
<input type="checkbox" name="canProg" value="C++"/>C+
<input type="checkbox" name="canProg" value="PHP" checked="checked"/>PHP
</p>

<p>Hvilken bil har du? (Select)<br />
<select name="haveCar">
    <option value="Audi">Audi</option>
    <option value="BMW" selected="selected">BMW</option>
    <option value="Rover" >Rover</option>
    <option value="Folkevogn">Folkevogn</option>
</select>
</p>

<p>Hvilke biler kunne du tænke dig? (Select multiple)<br />
<select name="wishCar" size="4" multiple="multiple">
    <option value="Audi">Audi</option>
    <option value="BMW" selected="selected">BMW</option>
    <option value="Rover" selected="selected">Rover</option>
    <option value="VolksWagen">Folkevogn</option>
</select>
</p>

<p>
<input type="reset" value="Clear"/>
<input type="submit" name="btnSend" value="Indsend data"/>
<input type="image" src="/Parameters/images/JavaImage.jpg"
       name="imgSubmit" value="Java"/>
</p>
</form>
</body>
</html>

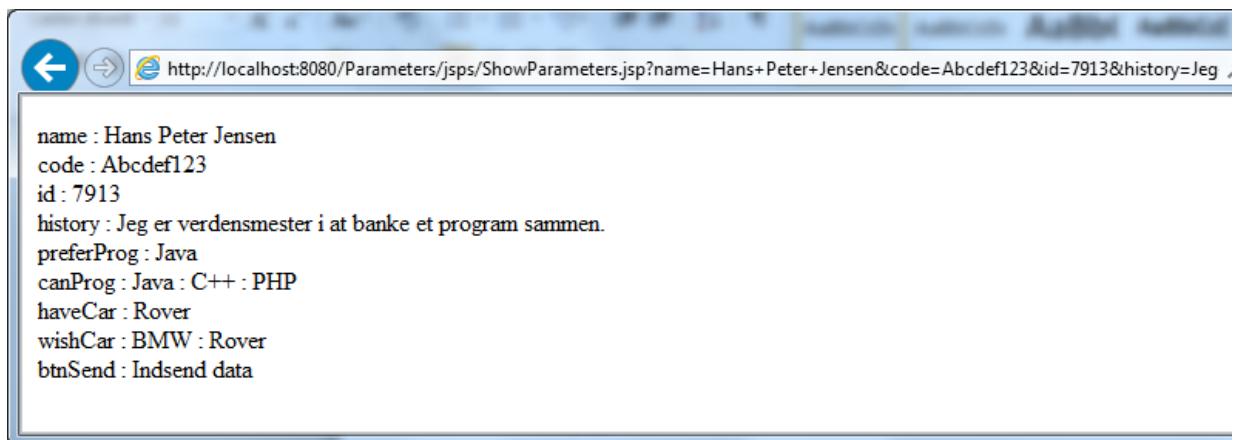
```

```

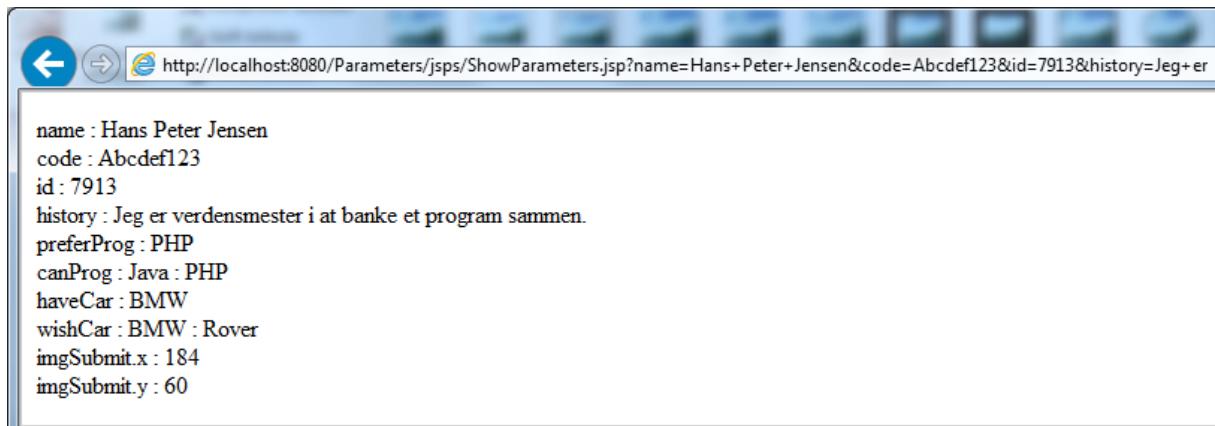
<%@page import="java.util.Enumeration"%>

<html>
  <head>
    <title>Insert title here</title>
  </head>
  <body>
    <%
      Enumeration<String> parameterNames = request.getParameterNames();
      while(parameterNames.hasMoreElements())
      {
        String parameterName = parameterNames.nextElement();
        String[] parameterValues = request.getParameterValues(parameterName);
        out.println(parameterName);
        for(String value : parameterValues)
          out.print(" : " + value);
        out.print("<br />");
      }
    %>
  </body>
</html>

```



Hvis der trykkes på billedet, som også er en submit, vil dennes parametre vises med x og y koordinater.



Form-taggets method-attribut kan undværes. I så fald defaulter den til metoden GET.

```
<form action="/Parameters/jsp/ShowParameters.jsp">
```

Er altså det samme som

```
<form action="/Parameters/jsp/ShowParameters.jsp" method="get">
```

Med metoden POST vil parametrene ikke være hægtet på URL'en, men være gemt i HTML-headeren.

```
<form action="/Parameters/jsp/ShowParameters.jsp" method="post">
```

Vil så give dette resultat i adresselinien.



Når der angives en URL i browserens adresselinie, vil det altid være en GET.

Det samme er tilfældet med en HTML-link. Den vil også altid være GET.

Efter forfatterens mening ville det være fornuftigt at bruge POST på en HTML-form, da der jo postes data til WEB-serveren.

Så ville brugerens første request af en WEB-side være GET og efterfølgende requests i sessionen være POST, og derved kan det afgøres om det er starten på en ny session og svares ud fra dette til brugeren.

Session

A screenshot of a web browser window titled "Shopping List". The URL in the address bar is "http://localhost:8080/SessionDemo/jsp/ShoppingList.jsp". The page content includes a text input field with the placeholder "Indtast vare der skal købes:" and a button labeled "Tilføj". Below this, a section titled "Indkøbslisten indeholder:" displays the text "Listen er tom". At the bottom, it says "Requestmetoden var GET".

Ved første request ses at listen er tom og metoden er GET.

Der skrives noget i tekstfeltet og trykkes på Tilføj.

A screenshot of a web browser window titled "Shopping List". The URL in the address bar is "http://localhost:8080/SessionDemo/jsp/ShoppingList.jsp". The page content includes a text input field with the placeholder "Indtast vare der skal købes:" and a button labeled "Tilføj". Below this, a section titled "Indkøbslisten indeholder:" displays the text "2 kg Mel". At the bottom, it says "Requestmetoden var POST".

Nu er der noget i listen og metoden er POST.

Osv....

The screenshot shows a web browser window with the following details:

- Address Bar:** http://localhost:8080/SessionDemo/jsp/ShoppingList.jsp
- Title Bar:** Shopping List
- Content Area:**
 - A text input field containing "1 liter Juice" with a clear ("X") button and a "Tilføj" (Add) button.
 - A section titled "Indkøbslisten indeholder:" (The shopping list contains:) listing the following items:
 - 2 kg Mel
 - 3 liter Mælk
 - 1 pakke Tyggegummi
 - 1 kasse Sodavand

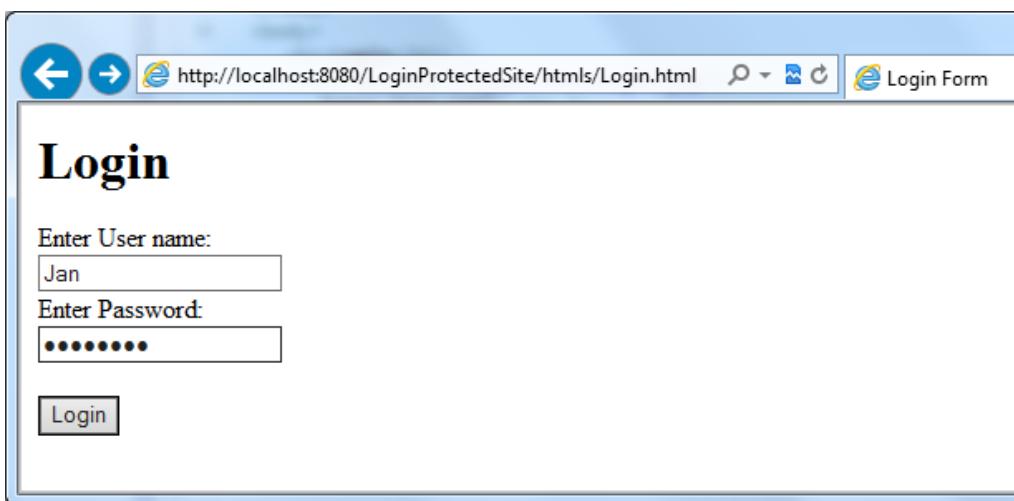
Login-beskyttet web-side

Session-objektet bruges også til at holde styr på om der er logget ind. Og eventuelt hvem der er logget ind.

Dette demonstreres her med en side, hvor brugernavn og password kan indtastes, en side, der logger ind, hvis alt er korrekt, og en side der kun kan vises hvis der er logget ind.

Filen Login.html:

```
<html>
  <head>
    <title>Login Form</title>
  </head>
  <body>
    <h1>Login</h1>
    <form action=".//jsps/Login.jsp">
      Enter User name: <br />
      <input type="text" name="LoginName"/><br />
      Enter Password: <br />
      <input type="password" name="password"/><br /><br />
      <input type="submit" name="btnLogin" value="Login"/>
    </form>
  </body>
</html>
```



Filen Login.jsp:

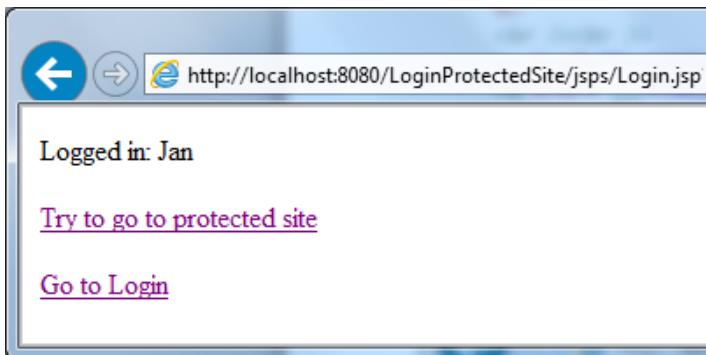
```
<html>
  <head>
    <title>Login JSP</title>
  </head>
  <body>
    <%
      String name = request.getParameter("loginName");
      String passw = request.getParameter("password");

      if(name.equals("Jan") && passw.equals("P@ssw0rd"))
      {
        session.setAttribute("LoggedIn", name);
        out.println("Logged in: " + name);
      }
    else
```

```

{
    session.removeAttribute("LoggedIn");
    out.println("Login failed");
}
%>
<br /><br />
<a href="ProtectedSite.jsp">Try to go to protected site</a><br /><br />
<a href="../htmls/Login.html">Go to Login</a>
</body>
</html>

```



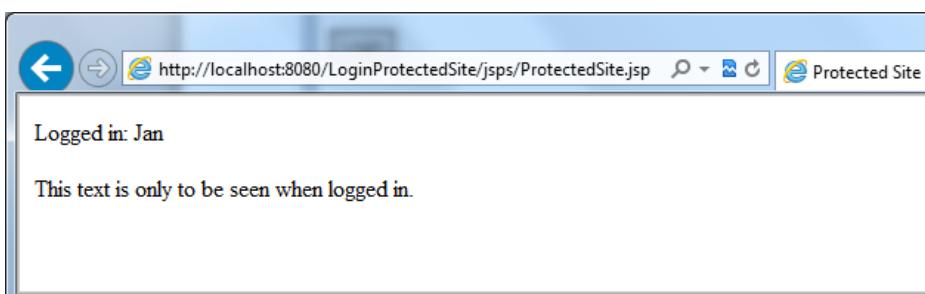
Filen ProtectedSite.jsp:

```

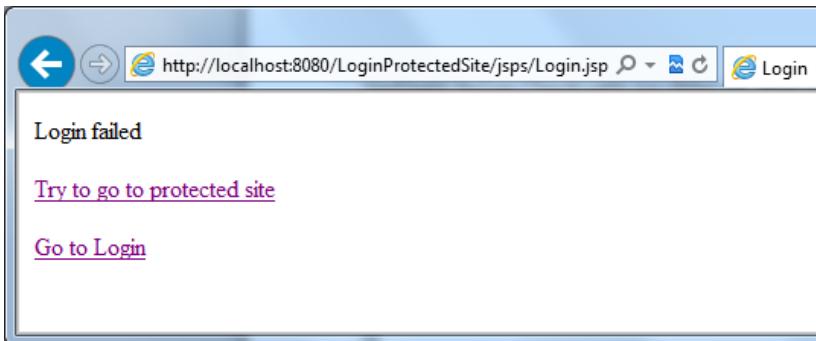
<%
if(session.getAttribute("LoggedIn") == null)
{
    response.sendRedirect("../htmls/Login.html");
}
%>

<html>
<head>
    <title>Protected Site</title>
</head>
<body>
    Logged in: <%=session.getAttribute("LoggedIn") %><br /><br />
    This text is only to be seen when logged in.
</body>
</html>

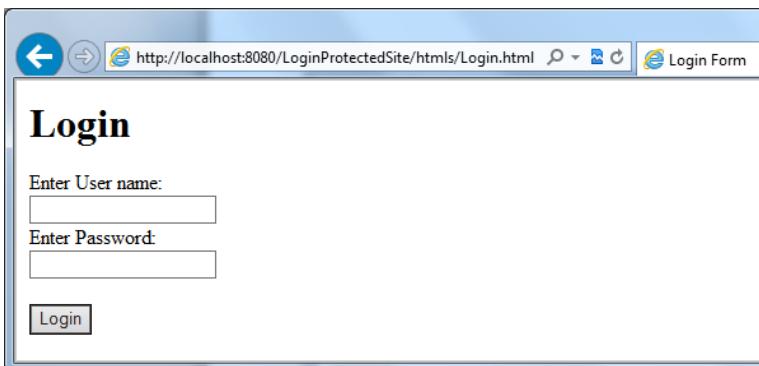
```



Ved login-fejl ser LogIn.jsp sådan ud,



og et tryk på linket Try to go to protected site, kalder Login.html.



Tjekke login på flere sider.

For at gøre det enklere at tjekke for login på hver side, hvor dette måtte ønskes, flyttes de første linier i filen ProtectedSite.jsp

```
<%
    if(session.getAttribute("LoggedIn") == null)
    {
        response.sendRedirect("../htmls/Login.html");
    }
%>

<html>
    <head>
        <title>Protected Site</title>
    </head>
    <body>
        Logged in: <%=session.getAttribute("LoggedIn") %><br /><br />
        This text is only to be seen when logged in.
    </body>
</html>
```

ud i en nyoprettet fil, filen CheckLoggedIn.jsp:

```
<%
    if(session.getAttribute("LoggedIn") == null)
    {
        response.sendRedirect("../htmls/Login.html");
    }
%>
```

Og der indsættes et kald, i form af et include direktiv i

Filen ProtectedSite.jsp:

```
<%@ include file="CheckLoggedIn.jsp" %>

<html>
  <head>
    <title>Protected Site</title>
  </head>
  <body>
    Logged in: <%=session.getAttribute("LoggedIn") %><br /><br />
    This text is only to be seen when logged in.
  </body>
</html>
```

Nu vil alle filer der indeholder denne include lede brugeren til login-siden, hvis der ikke er logget ind.

En include svarer til at koden i den inkluderede fil lægges ind hvor include-direktivet står.

Det vil sige at programmet udføres videre, når det inkluderede er udført.

Dette sker her når der er logget ind, men det vil ikke ske hvis der ikke er logget ind, da der jo så udføres en omdirigering i den inkluderede fil, CheckLoggedIn.jsp.

Omdirigering til anden web-side

Omdirigering kan principielt foregå på to måder:

- Klient-omdirigering - der sendes en response til browseren, hvor den bedes om at requeste en anden side.
- Server-omdirigering – browseren mærker ikke at der skiftes side, men modtager bare noget andet indhold i response.

Omdirigering skal ske inden der sendes noget som helst response tilbage til browseren. I en JSP-side er der en response-buffer, som ikke må løbe fuld inden der omdiriges. I en Servlet er der ingen buffer, så omdirigering skal ske før der outputtes noget fra koden.

Klient-omdirigering

I forrige eksempel med en beskyttet side er brugt Java-koden

```
response.sendRedirect("../htmls/Login.html");
```

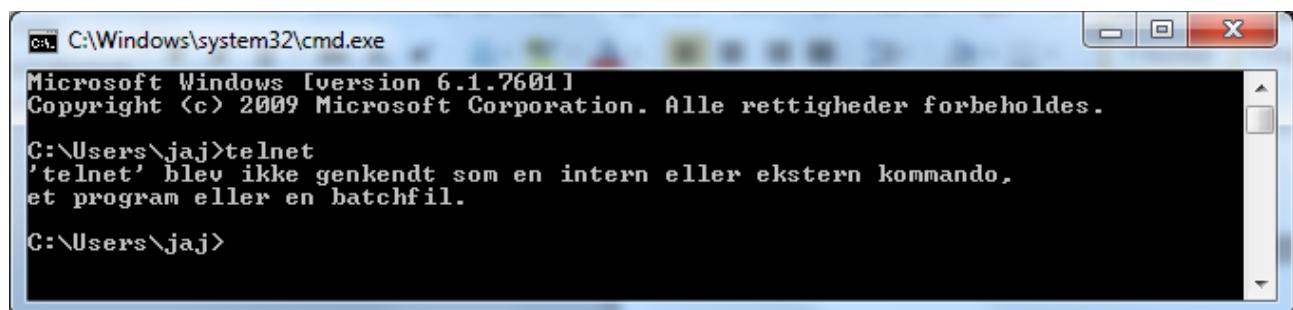
til at omdirigere browseren til login-siden.

Det er en såkaldt klient-omdirigering, hvor browseren får at vide, som svar på en forespørgsel, at den skal bede om en anden side.

For at vise dette, kan man bruge telnet til at sende HTTP-kommandoer som request til web-serveren og se modtagne respons, det hele i tekstform.

Telnet er som udgangspunkt ikke aktiveret i Windows 7.

Åben en CMD-prompt og skriv telnet og enter.



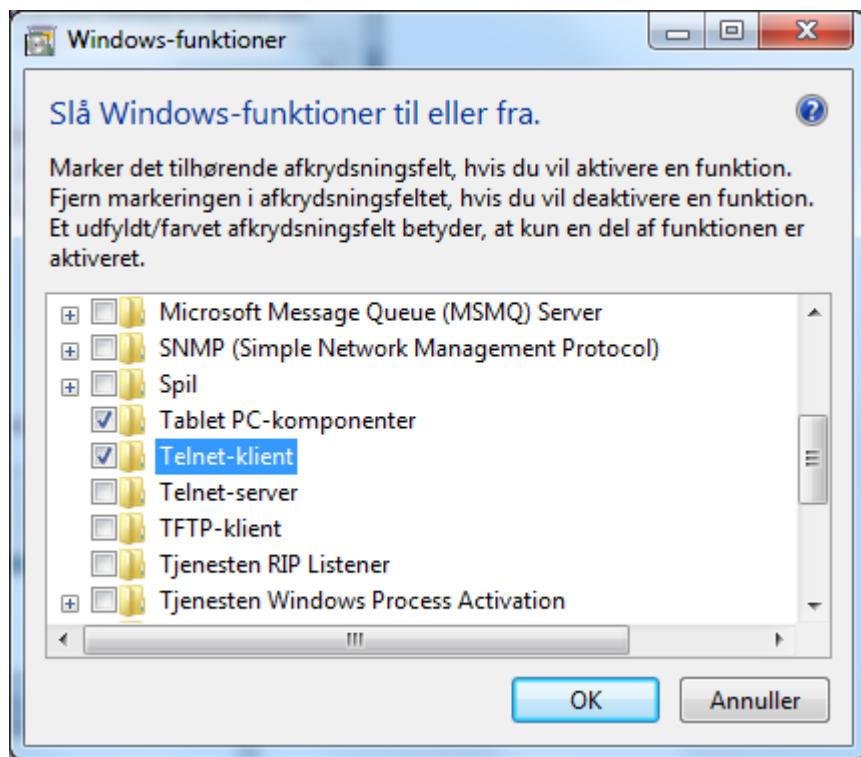
The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. The window displays the following text:

```
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle rettigheder forbeholdes.

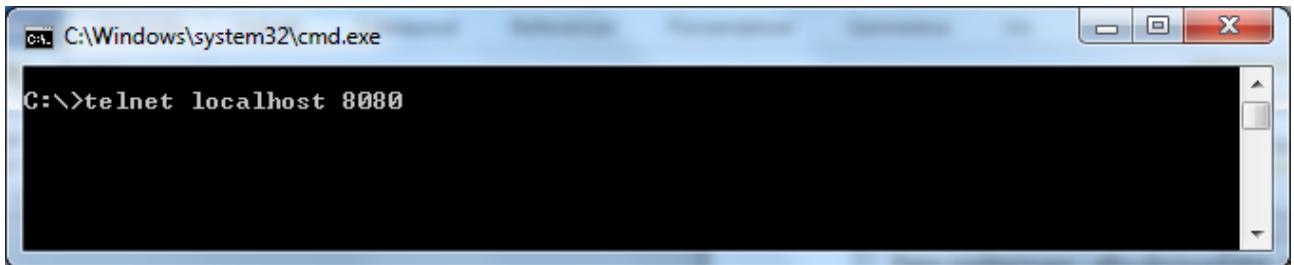
C:\Users\jaj>telnet
'telnet' blev ikke genkendt som en intern eller ekstern kommando,
et program eller en batchfil.

C:\Users\jaj>
```

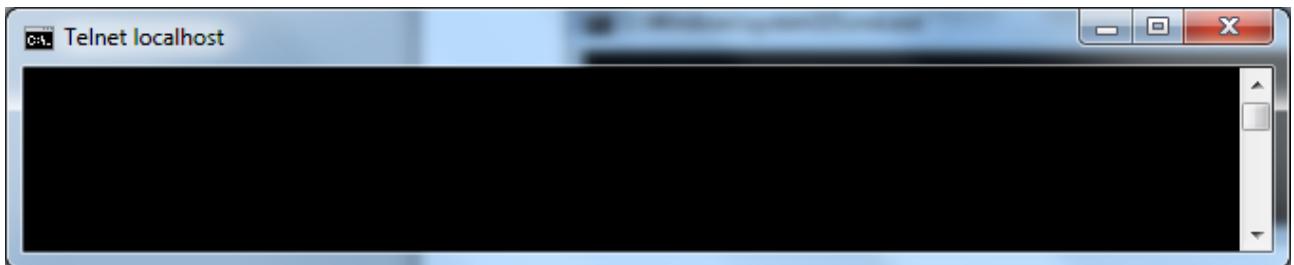
For at slå telnet til på Windows 7, åbnes Kontrolpanel | Programmer | Programmer og funktioner | Slå Windows-funktioner til eller fra.



Nu kan telnet startes og forbindes til serveren på den rette port.



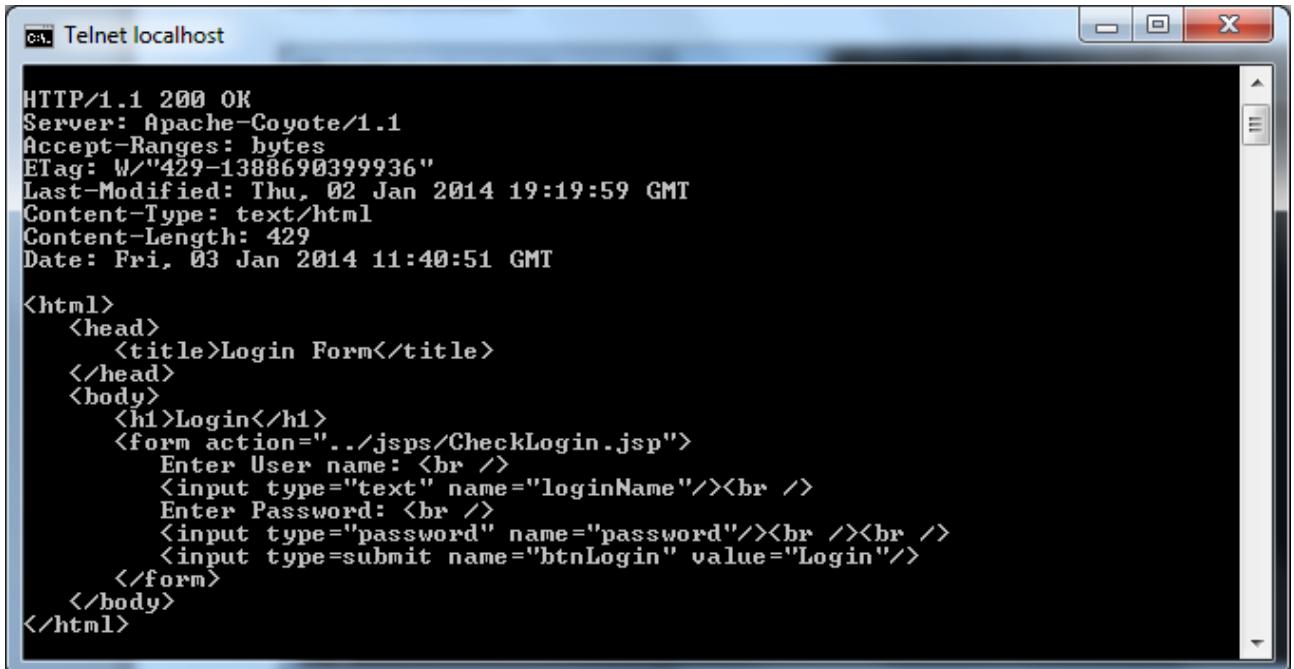
Reaktionen er en tom skærm hvor der kan indtastes HTTP-kommandoer.



Det man skriver ses ikke i prompten (kan nok løses), så kopier følgende kommando ind og tryk enter.

```
GET /LoginProtectedSite/htmls/Login.html HTTP/1.1  
Host: localhost:8080
```

Dette giver følgende response fra serveren, hvis der er logged ind.

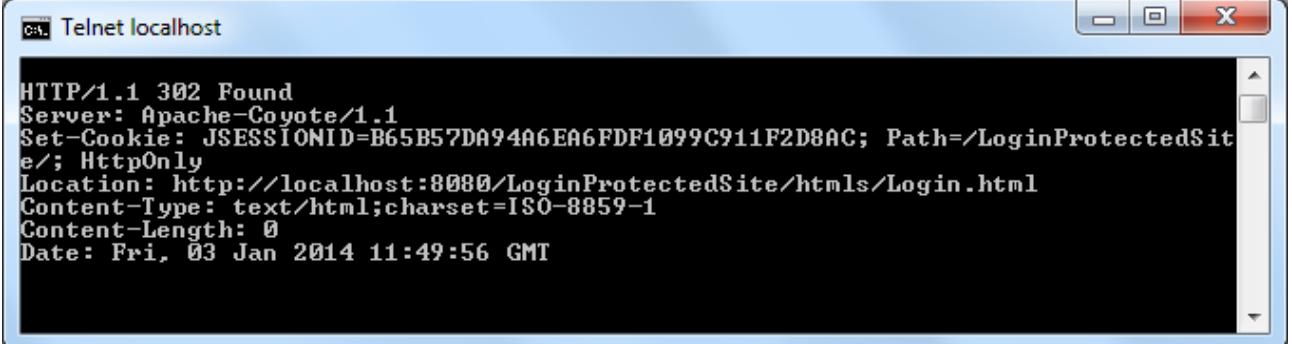


Hvis det forsøges at hente den beskyttede side uden at være logget ind med følgende kommando

GET /LoginProtectedSite/jsp/ProtectedSite.jsp HTTP/1.1

Host: localhost:8080

Fås denne response fra serveren, da den pågældende side omdirigerer til login-siden



```
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=B65B57DA94A6EA6FDF1099C911F2D8AC; Path=/LoginProtectedSite; HttpOnly
Location: http://localhost:8080/LoginProtectedSite/htmls/Login.html
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 0
Date: Fri, 03 Jan 2014 11:49:56 GMT
```

Browseren får at vide at den ønskede side er flyttet til den angivne Location, og browseren sender så en ny forespørgsel til denne, og får login-siden igen.

Klient-omdirigering kan også udføres med følgende javascript kode

```
<script>document.location="../htmls/Login.html"</script>
```

Server-omdirigering

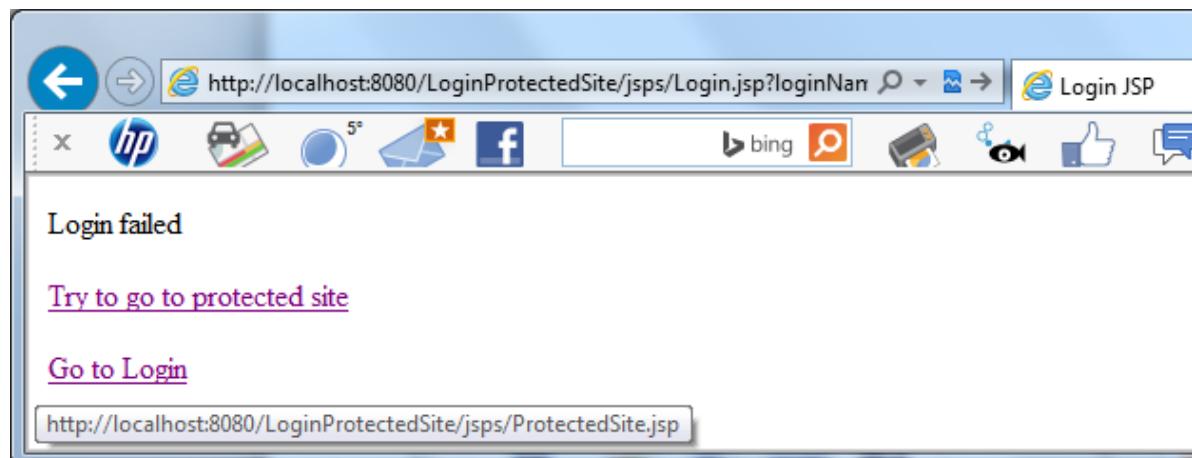
Server-omdirigering kan udføres med følgende tag

```
<jsp:forward page=". /htmls/Login.html"></jsp:forward>
```

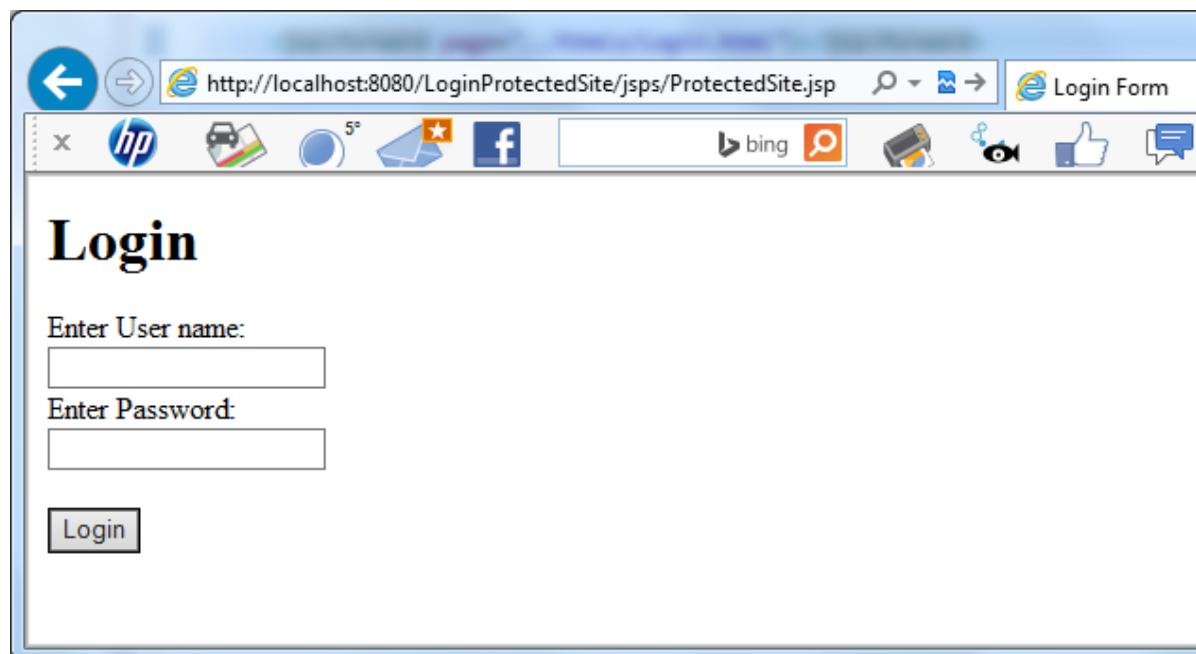
Ved server-omdirigering er browseren uvidende om omdirigeringen.

Herunder ses en login der er fejlet.

Ved tryk på linket Try to go to protected site, udføres ProtectedSite.jsp (Muse-cursoren står på linket og viser hint forneden)



ProtectedSite.jsp omdirigerer til Login.html



Men der står stadig ProtectedSite.jsp adressefeltets URL.

I omdirigerings-tagget er det muligt at angive parametre til den side, der omdiriges til.

Denne skulle logge ind. (Men jeg fik det ikke til at virke)

```
<jsp:forward page="Login2.jsp">
  <jsp:param value="Jan" name="LoginName"/>
  <jsp:param value="P@ssw0rd" name="password"/>
</jsp:forward>
```

Server-omdirigering kan også ske med følgende java-kode.

```
request.getRequestDispatcher("../htmls/Login.html").forward(request, response);
```

Men jeg fik denne exception

```
java.lang.IllegalStateException: getOutputStream() has already been called for this response
```

NB! Jeg oplevede en masse problemer med Server-omdirigering. Som om at jsp-filerne ikke blev oversat, sådan at ændringer i dem ikke havde virkning. Det løstes ved at rename filerne, hvilket jo ikke er tilfredsstillende. Problemet er måske Eclipse-Tomcat forbindelsen. Man kan dobbeltklikke på serveren for det aktuelle projekt og se hvor de oversatte jsp-servlet-klasser lægges, men jeg kunne ikke finde alle. Der kan vælges med radiobuttons at lægge filerne normalt på Tomcat, men de var greyet out.

Dette virker

Der overføres en ArrayList fra servlet til en jsp via request-objektet.



NewName.jsp:

```
<form method="post" action="NewNameServlet">
    <input type="text" name="txtNewName" />
    <input type="submit" name="btnNewName" />
</form>
```

NewNameServlet.java:

```
@WebServlet("/NewNameServlet")
public class NewNameServlet extends HttpServlet
{
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        String name = request.getParameter("txtNewName");

        ArrayList<String> names = new ArrayList<String>();
        names.add(name);
        names.add(name);
        names.add(name);

        request.setAttribute("names", names);
        request.getRequestDispatcher("ShowAll.jsp").forward(request, response);
    }
}
```

ShowAll.jsp:

```
<h1>Hej fra ShowAll.jsp</h1>
<%
    ArrayList<String> names = (ArrayList<String>)request.getAttribute("names");
    for(String name : names)
    {
        <h2>Det nye navn var <%=name %></h2>
    }
%>
```



The screenshot shows the Eclipse IDE interface with the 'Server View' open. The title bar reads 'Tomcat v7.0 Server at localhost (5)'. The toolbar has various icons for file operations like Open, Save, and Cut. Below the toolbar, several files are listed: Login.html, Login3.jsp, CheckLoggedIn3.jsp, ProtectedSite3.jsp, and Tomcat v7.0 Server at localhost (5). The main area is titled 'Overview'.

General Information
Specify the host name and other common settings.

Server name: Tomcat v7.0 Server at localhost (5)
Host name: localhost
Runtime Environment: Apache Tomcat v7.0
Configuration path: /Servers/Tomcat v7.0 Server at localhost (5)-config [Browse...](#)
[Open launch configuration](#)

Server Locations
Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

(Use workspace metadata (does not modify Tomcat installation))
(Use Tomcat installation (takes control of Tomcat installation))
(Use custom location (does not modify Tomcat installation))

Server path: .metadata\plugins\org.eclipse.wst.server.core\tmp4 [Browse...](#)
[Set deploy path to the default value \(currently set\)](#)

Deploy path: wtpwebapps [Browse...](#)

Server Options
Enter settings for the server.

Serve modules without publishing
 Publish module contexts to separate XML files
 Modules auto reload by default
 Enable security
 Enable Tomcat debug logging (not supported by this Tomcat version)

Overview [Modules](#)

Server path ligger under projektet i Eclipse workspace.

Java og databaser

For at få adgang til databaser på MS Sql Server fra Java, skal der hentes en JDBC-driver.

Søg på Google og vælg det viste nederste hit.

Google search results for "JDBC Driver for Microsoft SQL Server". The top result is an advertisement from Microsoft for the SQL Server JDBC driver. Below it is a link to the Microsoft JDBC Driver for SQL Server - MSDN page.

Vælg det feste link.

Microsoft JDBC Driver for SQL Server - MSDN

msdn.microsoft.com/en-us/sqlserver/aa937724.aspx • Oversæt denne side
In its continued commitment to interoperability, Microsoft provides a Java Database Connectivity (JDBC) driver for use with SQL Server.
Microsoft JDBC Driver 4.0 for ... - Connecting to SQL Server with ... - Versions

SQL Server Developer Center > Home > Versions > SQL Server 2008 R2 and SQL Server 2008 > Connectivity > Microsoft JDBC Driver for SQL Server

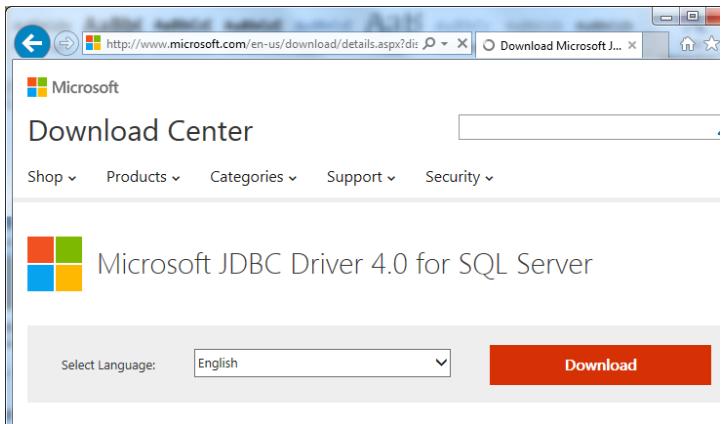
Microsoft JDBC Driver for SQL Server

In our continued commitment to interoperability, Microsoft provides a Java Database Connectivity (JDBC) driver for use with SQL Server, SQL Azure, and Parallel Data Warehouse (PDW). The Microsoft JDBC Driver for SQL Server is available at no additional charge, and provides Java Database Connectivity from any Java application, application server, or Java-enabled applet. This driver is a Type 4 JDBC driver that provides database connectivity through the standard JDBC application program interfaces (APIs).

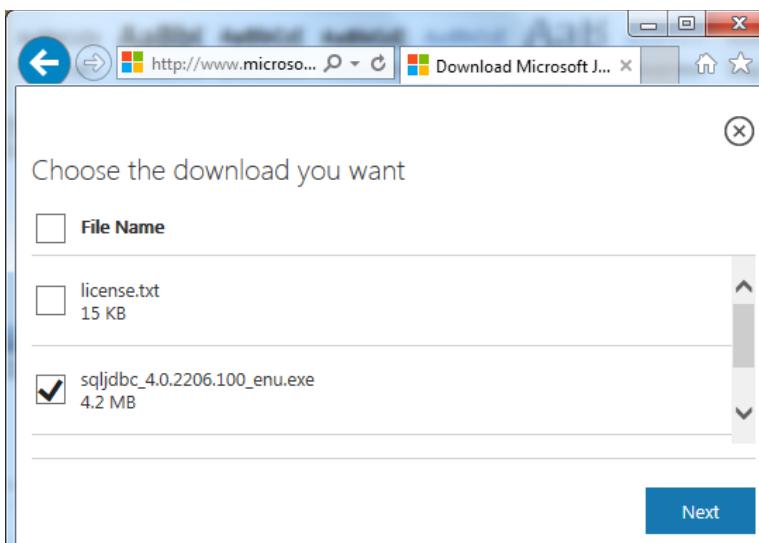
The Microsoft JDBC Driver for SQL Server has been tested against major application servers such as IBM WebSphere, and SAP NetWeaver.

Download the Microsoft JDBC Driver 4.0 for SQL Server

The 4.0 version of the JDBC Driver provides support for features introduced in SQL Server 2012, including AlwaysOn, Correlated Tracing via XEvents, and UTF-16 support. This release also adds Type 4 Kerberos support on Windows and non-Windows platforms.



Og hent sqljdbc_4.0.2206.100_enu.exe.



Gem filen og udfør den derefter.

Der bliver udpakket en filstruktur, hvor filen sqljdbc4.jar kan findes.



Træk denne fil ind i Eclipse-projektet under mappen lib.

Så kan der arbejdes med databaser fra Java.

(Det er også muligt at referere til filen ved at gå ind i Properties | Java Build Path | Libraries | Add External Jar, men det virker ikke når der skal udføres på webserver Apache Tomcat)

Databaseeksempel

Der oprettes en tabel i databasen af navnet Elev

```
create table Elev
(
    Id int,
    FName varchar(25),
    LName varchar(25),
    Job varchar(20),
    Salary decimal(8,2)
)

insert into Elev values(1, 'Allan', 'Larsen', 'Programmør', 1234.50)
insert into Elev values(2, 'Jakob', 'Rames', 'Rengøringsmanager', 32344.50)
insert into Elev values(3, 'Jonas', 'Adersen', 'Tagrenderenser', 56230.50)
insert into Elev values(4, 'Nicholas', 'Volante', 'Kagehjemmebringer', 10234.50)
```

Der oprettes en tilsvarende klasse Elev.

Elev.java:

```
public class Elev
{
    int id;
    String fName;
    String lName;
    String job;
    double salary;

    public Elev(int id, String fName, String lName, String job, double salary)
    {
        this.id = id;
        this.fName = fName;
        this.lName = lName;
        this.job = job;
        this.salary = salary;
    }

    public int getId(){return this.id;}
    public String getFName(){return this.fName;}
    public String getLName() {return lName;}
    public String getJob() {return job;}
    public double getSalary() {return salary;}
}
```

Der oprettes en database-hjælpeklasse med en constructor, der opretter forbindelse til databasen, samt en metode, der kan returnere et elev-objekt svarende til en række i tabellen Elev.

DBTools.java:

```
import java.sql.*;

public class DBTools
```

```

{
    String conStr =
        "jdbc:sqlserver://localhost:1433; databaseName=JavaDB; User=sa; Password=1234;";

    Connection con;
    Statement stmt;
    ResultSet rs;

    public DBTools()
    {
        try
        {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

            con = DriverManager.getConnection(conStr);

            stmt = con.createStatement();

            System.out.println("Alt er godt!!!!");
        } catch (SQLException e) {
            System.out.println("SQL Exception!!! " + e.getMessage());
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            System.out.println("Class not found Exception!!! " + e.getMessage());
            e.printStackTrace();
        }
    }

    public Elev getElevById(int id)
    {
        Elev elev = null;
        try
        {
            rs = stmt.executeQuery("select * from Elev where id = " + id);
            if(rs.next())
            {
                elev = new Elev(
                    rs.getInt("Id"),
                    rs.getString("FName"),
                    rs.getString("LName"),
                    rs.getString("Job"),
                    rs.getDouble("Salary")
                );
            }
        } catch (SQLException e) {
            System.out.println("Exception!!! " + e.getMessage());
            e.printStackTrace();
        }
        return elev;
    }
}

```

Dette benyttes i jsp-filen til at udskrive navnet på en elev.

ElevShow.jsp:

....

```

<body>
<%
    DBTools dbTools = new DBTools();

```

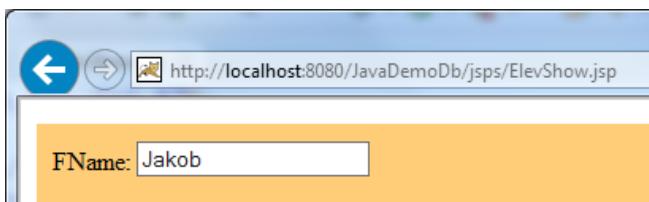
```

Elev elev = dbTools.getElevById(2);
%>
<form>
    FName: <input type="text" name="txtFName" value="<%elev.getFName()%>" />
</form>

</body>
</html>

```

Resultat:



Et ResultSet er online, således at connection til databasen skal opretholdes indtil det er brugt færdigt. Den henter altså data fra databasen når man læser fra den.

Det er ikke særligt praktisk at returnere et ResultSet fra en metode, fordi connection skal lukkes ved færdig brug. Her kan man i stedet bruge CachedRowSet, som indeholder hele resultatet. For at gøre dette skal der hentes og benyttes com.sun.rowset.jar.

```

import com.sun.rowset.CachedRowSetImpl;
import java.sql.*;

public class SQLInterface {

    private CachedRowSetImpl crs;
    private String url;

    public SQLInterface(String username, String password) {
        url = "jdbc:mysql://localhost/catsdb?user=" +
              username + "&password=" + password;
    }

    public boolean execQuery(String query) {
        Statement stmt = null;
        Connection conn = null;
        ResultSet resultSet = null;

        try {
            Class.forName(
                "com.mysql.jdbc.Driver").newInstance();

            conn = DriverManager.getConnection(url);

            stmt = conn.createStatement();
            resultSet = stmt.executeQuery(query);

            // create CachedRowSet and populate
        }
    }
}

```

```

        crs = new CachedRowSetImpl();
        crs.populate(resultSet);

        // note that the connection is being closed
        conn.close();

        return true;

    } catch (SQLException se) {
        ...
        return false;

    } catch (Exception e) {
        ...
        return false;
    }
}

public CachedRowSetImpl getRowSet() {
    return crs;
}
}

```

Noter

Nogle gange kommer der en fejlmeddeelse når man starter Tomcat inde fra Eclipse, om at der er ikke TLD'er i jar-filer. Fejlen er en timeout, som kan udvides ved at højreklikke på serveren i vinduet servers og vælge Open. Så kan Start-time limit sættes. Den er 45 sekunder fra starten.

