



# Back End Development 1 ○

---

+

## Sesi 14

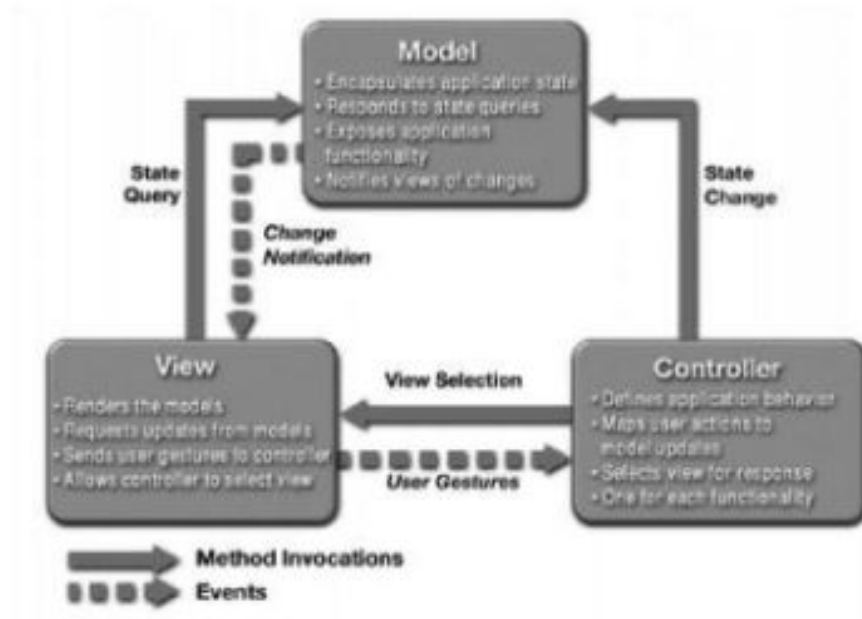


# --- Servlet & Springboot

+

## Servlet & Springboot - Sesi 14

# Arsitektur MVC



[https://www.researchgate.net/figure/Model-View-Controller-MVC-Architecture\\_fig3\\_321293423](https://www.researchgate.net/figure/Model-View-Controller-MVC-Architecture_fig3_321293423)

Arsitektur MVC secara sederhana dirancang dan diadaptasi dalam penggunaan pada web application. Arsitektur yang dihasilkan kemudian disebut Model 2 Architecture.

MVC merupakan sebuah arsitektur model yang membantu aplikasi untuk berkonsentrasi pada tugas dan fungsi masing-masing. Tugas dan fungsi tersebut dibagi menjadi :

- a. Model :mempresentasikan data dan logika bisnis, biasanya berhubungan dengan basis data.
- b. View: menampilkan data atau mengatur tampilan ke pengguna
- c. Controller: menghubungkan antara view dengan model

Aplikasi Model 2 umumnya memiliki :

- Servlet Controller yang menyediakan akses tunggal terhadap keseluruhan aplikasi. Controller ini bertanggung jawab menyediakan manajemen terpusat terhadap alur aplikasi dan juga service lain seperti penanganan security dan user management.
- Controller servlet umumnya menggunakan konfigurasi XML untuk menentukan alur aplikasi dan pemrosesan perintah. Hal itu juga membuat helper components terasosiasi dengan user action dan dibuat/dipanggil untuk menangani actions yang terjadi, memanggil komponen Model sebagaimana diperlukan. Hal ini berfungsi untuk memisahkan antara controller servlet dari model.

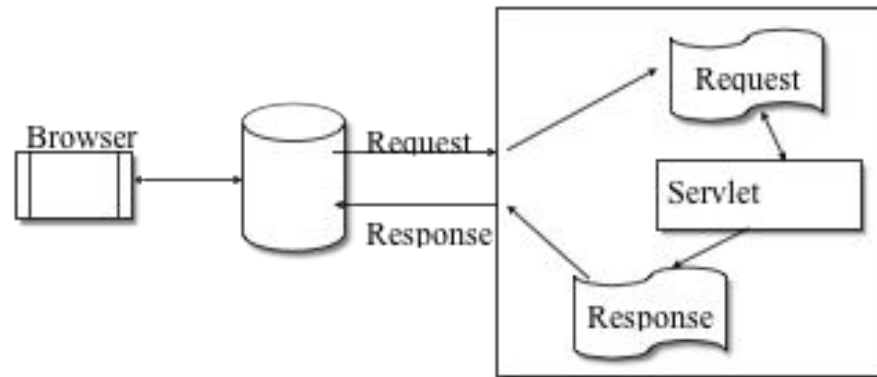
Contoh aplikasi MVC :

Aplikasi ini akan menampilkan saldo rekening bank. Servlet controller akan membaca ID Customer dan meneruskannya pada kode data akses yang mengembalikannya ke bean Bankcustomer. Kemudian, bean tersebut disimpan ke dalam objek `HttpServletRequest` yang dapat diakses halaman JSP tujuan.

Jika saldo rekening negative, maka servlet meneruskannya ke halaman yang menampilkan pesan saldo negative. Jika saldo normal, akan diteruskan ke tampilan untuk saldo normal, dan untuk saldo tinggi, akan ditampilkan halaman ke halaman khusus dengan saldo tinggi. Ketika ID tidak diketahui, maka halaman error akan muncul.

### NICE TO KNOW

Class Java yang memiliki kemampuan sebagai server, sederhananya kita cukup melakukan koding class-class java.



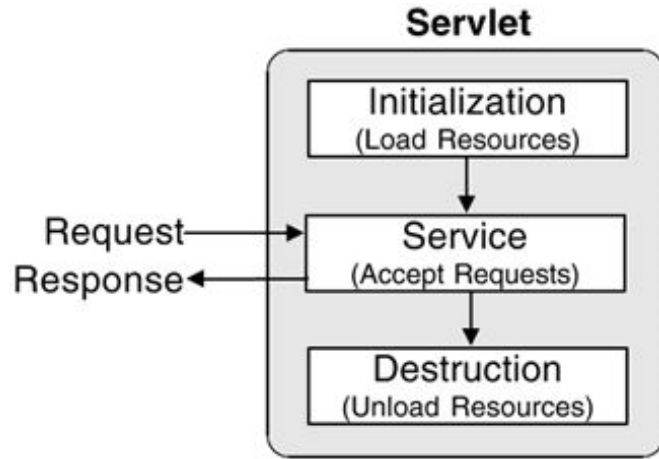
Client (browser) mengirimkan HTTP request pada servlet container, kemudian servlet container akan menghantar request ke servlet yang dimaksud. Servlet akan menjalankan program java dan jika diperlukan servlet bisa mengakses database atau lainnya untuk menghasilkan output berupa halaman HTML.

Halaman HTML ini diserahkan ke servlet container untuk dikirim kembali sebagai response kepada client.

```
public class MyServlet extends HttpServlet
```

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





Siklus hidup servlet diawali dengan method `init()` yang dipanggil oleh web container setelah servlet diinisialisasi.

Setelah servlet di-inisialisasi, web container memanggil method `service()`. Pada tahap ini servlet siap menunggu request untuk kemudian melakukan proses dan mengirimkan response kepada client.

Method `destroy()` dipanggil ketika web container dimatikan atau servlet di undeploy. Untuk mengaktifkan kembali servlet harus di inisialiasi lagi dari awal.

## Servlet & Springboot - Sesi 14

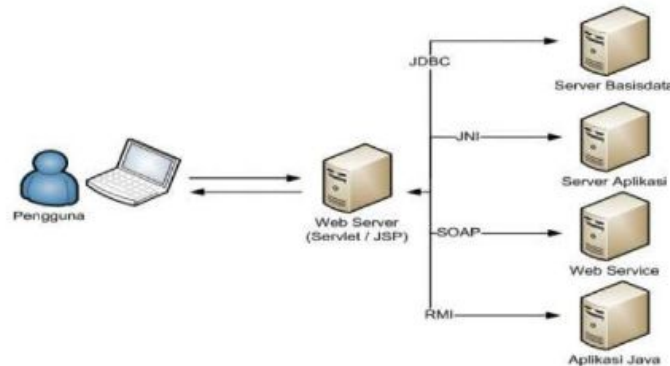
# Introduction of Servlet

Kembali ke point pembelajaran pada sesi 11 ini :

APA ITU SERVLET ?

Servlet adalah **program Java yang berjalan di web atau aplikasi server.**

**Servlet berfungsi sebagai middle layer (penghubung) antar web browser atau client HTTP dengan database atau aplikasi pada server HTTP.**





Penjelasan :

1. Membaca data yang dikirim oleh client. Data yang diterima biasanya berasal dari form HTML, Applet, dan HTML client ;
2. Membaca request yang dikirim oleh browser
3. Menghasilkan output yang akan ditampilkan ke pengguna
4. Mengirim dokumen kepada client

Dalam membangun servlet, perlu diperhatikan empat point sebagai berikut :

1. Kode program adalah kode Javabiasa. Ada API baru, tapi tidak ada sintaks yang baru;
2. Memiliki pernyataan import yang tidak familiar
3. Turunan kelas dari kelas `HTTPServlet`. Servlet menyediakan berbagai macam fitur untuk menangani HTTP
4. OverridemethoddoGet.Servletmemilikimethodyangberbedauntuk mengananiberbagai jenis perintah HTTP



## Servlet & Springboot - Sesi 14

# Introduction of JavaBean

Ada tiga hal yang perlu diketahui tentang JavaBean, yaitu :

1. Kelas Java Bean seharusnya memiliki konstruktor tanpa argument (default constructor)
2. Kelas Java Bean seharusnya memiliki atribut yang didefinisikan sebagai private
3. Nilai variable di objek bean harus diakses melalui method getXxx dan SetXxx

Berikut ini merupakan keuntungan dari Java Bean :

1. Dengan menggunakan bean, halaman JSP dapat memanipulasi objek hanya dengan menggunakan sintaks XML
2. Pada konstruksi JSP dan Java Bean, berbagi (sharing) obyek di antara berbagai halaman atau request akan lebih mudah daripada menggunakan kode Java Eksplisit
3. Konstruksi JSP dan Java Bean menyederhanakan proses request pada saat membaca parameter, mengubah parameter dari string, dan menempatkan hasil perubahan tersebut di dalam objek.



## Menggunakan Bean

Untuk membangun dan memanipulasi komponen Java Bean dalam halaman JSP, gunakan tiga tag berikut ini :

### **jsp:useBean**

Tag ini membangun Bean baru. Format tag selengkapnya adalah sebagai berikut :

```
<jsp:useBean id ="beanName" class="package.Class"/>
```

### **jsp:getProperty**

Tag ini membaca nilai property bean. Membaca property pada prinsipnya memanggil method bernama getXxx. Format tag selengkapnya adalah sebagai berikut :

```
<jsp:getProperty name="beanName" property="propertyName"/>
```

### **jsp:setProperty**

Tag ini memodifikasi sebuah property bean (misalnya pemanggilan method bernama setXxx). Format tag selengkapnya adalah sebagai berikut :

```
<jsp:setProperty name = ``beanName" property = "propertyName" value="propertyValue"/>
```



Terakhir pembahasan Teknologi akhir dari J2EE adalah JSP

JSP (Java Server Page) adalah suatu teknologi web berbasis bahasa pemrograman java dan berjalan di Platform Java, serta merupakan bagian teknologi J2EE (Java 2 Enterprise edition). JSP memiliki sifat-sifat sebagai berikut :

1. Portable karena dibuat dengan teknologi java
2. Manajemen memory
3. Memiliki akses ke API Java yang lengkap seperti JDBC dan Java Mail
4. Dapat menggunakan komponen yang portable dan reusable (JavaBean)
5. Memiliki kinerja tinggi terhadap banyak request atau proses sekaligus dalam waktu yang sama
6. Mudah dalam deployment dan maintenance

Selain itu, JSP memiliki kelebihan yang membuatnya patut dipertimbangkan sebagai bahasa pemrograman web untuk pembuatan aplikasi web yang tangguh. Kelebihan tersebut adalah :

1. Memisahkan presentasi static dan isi yang dinamik
2. Menekankan komponen reusable
3. Memudahkan pembuatan aplikasi dengan tag
4. Berbasis pemrograman bahasa java
5. Bagian dari platform Java
6. Terintegrasi dalam J2EE

Lokasi default pemasangan halaman JSP di Tomcat dengan menggunakan `http://host/SomeFile.jsp`

Nah dalam prakteknya di dunia MVC, Langsung saja buat sebuah project Java Web baru dengan menggunakan netbeans kalian.

Buatlah sebuah class Member yang akan bertindak sebagai model pada package com.sesi11.model

```
public class Member
{
    private String username;
    private String password;

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

Lalu pada halaman index.jsp yang bertindak sebagai view ubah menjadi seperti ini

```
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3    "http://www.w3.org/TR/html4/loose.dtd">
4
5  <html>
6    <head>
7      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8      <title>JSP Page</title>
9    </head>
10   <body>
11     <form action="doLogin.do" method="post">
12       <table border="1">
13         <tr>
14           <td>Username</td>
15           <td><input type="text" name="username" id="username">
16         </tr>
17         <tr>
18           <td>Password</td>
19           <td><input type="password" name="password" id="password"></td>
20         </tr>
21         <tr>
22           <td colspan="2">
23             <input type="submit" value="Login">
24           </td>
25         </tr>
26       </table>
27       <%
28         String err = (String) request.getAttribute("err");
29         if(err!=null)
30         {
31           %>
32           <script type="text/javascript">
33             alert('<%=err%>');
34           </script>
35           <%
36         }
37       %>
38     </form>
39   </body>
40 </html>
```



Pada form di atas, jika disubmit akan menuju ke doLogin.do.

Nah doLogin.do itulah yang bertindak sebagai servlet sekaligus sebagai controller.

Maka buatlah sebuah class LoginController di dalam package com.sesi11.controller

```
1 package com.sesi11.controller;
2
3 import com.sesi11.model.Member;
4 import java.io.IOException;
5 import javax.servlet.RequestDispatcher;
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import javax.servlet.http.HttpSession;
11 import org.apache.catalina.Session;
12
13 public class LoginController extends HttpServlet
14 {
15     protected void doPost(HttpServletRequest request, HttpServletResponse response)
16         throws ServletException, IOException
17     {
18         String username = request.getParameter("username");
19         String password = request.getParameter("password");
20         String err="";
21         int q=0;
22         if (username.equals("") || username==null)
23         {
24             err="Username harus diisi";
25         }
26         else if(password.equals("") || password==null)
27         {
28             err="Password harus diisi";
29         }
30         else
31         {
32             HttpSession session = request.getSession();
33             RequestDispatcher rd = request.getRequestDispatcher("member.jsp");
34             Member member = new Member();
35             member.setUsername(username);
36             member.setPassword(password);
37             session.setAttribute("member", member);
38             rd.forward(request, response);
39         }
40     }
41 }
```





```
41     if(q==0)
42     {
43         request.setAttribute("err", err);
44         RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
45         rd.forward(request, response);
46     }
47 }
48 }
```



Sekarang kita akan mendaftarkan class LoginController tersebut sebagai servlet di file web.xml.

Buka file web.xml kalian dan ubah menjadi seperti ini :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
4  ="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
5
6
7      <servlet>
8          <servlet-name>loginServlet</servlet-name>
9          <servlet-class>com.sesi11.controller.LoginController</servlet-class>
10     </servlet>
11     <servlet-mapping>
12         <servlet-name>loginServlet</servlet-name>
13         <url-pattern>/doLogin.do</url-pattern>
14     </servlet-mapping>
15
16     <session-config>
17         <session-timeout>
18             30
19         </session-timeout>
20     </session-config>
21     <welcome-file-list>
22         <welcome-file>index.jsp</welcome-file>
23     </welcome-file-list>
24 </web-app>
```



Sekarang kita tinggal membuat file member.jsp yang bertindak sebagai view

```
1  <html>
2      <head>
3          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4          <title>JSP Page</title>
5      </head>
6
7      <jsp:useBean id="member" scope="session" class="com.sesi11.model.Member" />
8      <body>
9          Welcome, <%=member.getUsername()%>
10     </body>
11 </html>
12
```

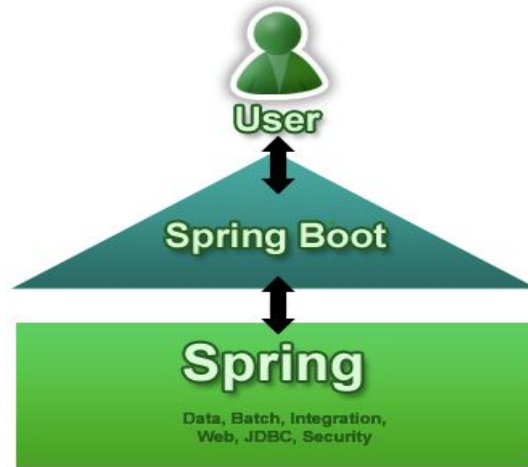
selesai dan jalankan..



**Spring boot** +

## Servlet & Springboot - Sesi 14

# Introduction



Spring merupakan framework Java yang mempermudah para programmer dalam membuat sebuah aplikasi Java dengan menerapkan salah satunya adalah design-pattern : dependency-injection.

*Dependency Injection* atau yang biasa disingkat DI terkenal didunia pemrograman setelah banyak bermunculan framework yang menerapkan konsep ini. Salah satunya dan yang paling terkenal adalah Spring.

DI dimaksudkan agar suatu kelas tidak terikat erat dengan kelas yang lain sehingga hubungan antar bagian kode menjadi longgar (loosely coupled).

Diartikan secara bahasa gampang DI berarti kita memberikan/menginjeksi suatu kelas ke kelas yang lain yang merupakan dependensinya (membutuhkannya).

Mengapa Menggunakan Framework Spring ?

1. Dependency Injection
2. Aspect Oriented Programming
3. Spring MVC dan Restful Web Service
4. Support koneksi database, dsb.

**Spring Boot** merupakan salah satu jenis framework dari Spring. Namun di Spring Boot , kita lebih dipermudah dalam pembuatan program karena :

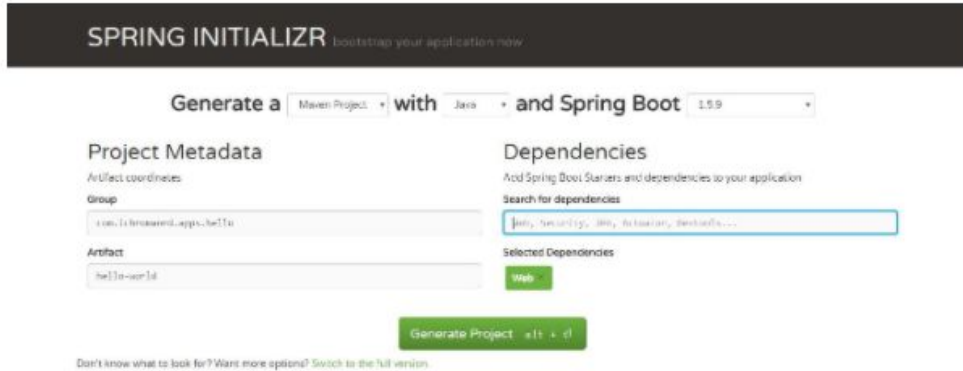
1. Sudah disediakan Tomcat dan beberapa server lain, sehingga kita hanya perlu run.
2. Menggunakan Maven sebagai build manager yang dapat kita atur di Project Object Model (POM)
3. Anotasi-anotasi yang mempermudah kita dalam menentukan komponen kelas-kelas, dsb.



## Servlet & Springboot - Sesi 14

# Get's Started

Buka situs [Spring Initializr](https://start.spring.io). Disini kita akan meng-generate project Spring Boot pertama kita.



The screenshot shows the Spring Initializr web application. At the top, there's a dark header with the text "SPRING INITIALIZR" and a subtitle "bootstrap your application now". Below the header, there's a form to generate a project. The form has a title "Generate a" followed by a dropdown menu set to "Maven Project", then "with" followed by a dropdown menu set to "Java", and finally "and Spring Boot" followed by a dropdown menu set to "1.5.9". Below this, there are two main sections: "Project Metadata" and "Dependencies". The "Project Metadata" section has a label "Artifact coordinates" and a "Group" input field containing "com.hacktiv8.apps.hello". Below that is an "Artifact" input field containing "hello-world". The "Dependencies" section has a label "Add Spring Boot Starters and dependencies to your application" and a "Search for dependencies" input field containing "jpa, security, jsp, thymeleaf, bootstrap...". Below the search field is a "Selected Dependencies" section with a "Step" button. At the bottom of the form is a large green button labeled "Generate Project" with a download icon. Below the button is a link that says "Don't know what to look for? Want more options? Switch to the full version."

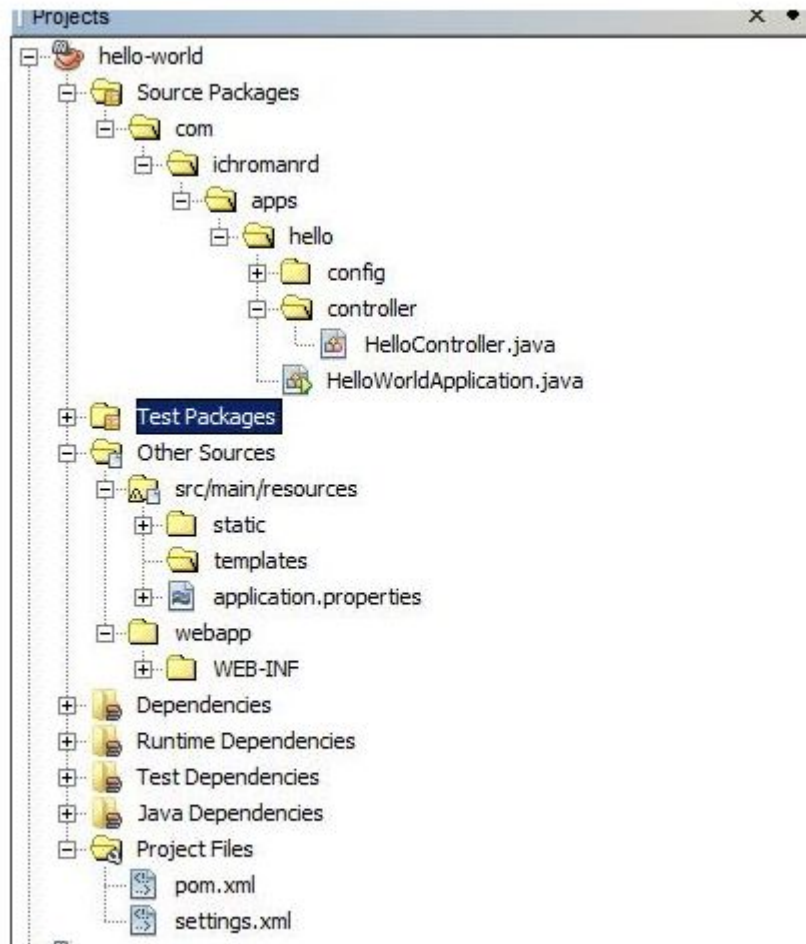
Keterangan :

- a. Isi field Group dengan nama organization atau project group kamu. Nantinya ini akan dipakai sebagai struktur package. Soal penamaan group ini baiknya mengikuti [Naming Conventions ini](#).**
- b. Field Artifact diisi dengan nama project kita.**
- c. Di bagian Dependencies sendiri, kita bisa bebas memilih dependency atau module apa yang dibutuhkan untuk aplikasi kita. Karena kita cuma pengen bikin aplikasi hello world, dependency Web sudah cukup lah.**
- d. Lalu klik Generate Project, dan itu akan otomatis men-download project kita dalam bentuk .zip. Silakan buka dengan IDE kamu. Saya pakai IDE kesayangan saya, NetBeans.**

Project yang baru di-generate itu dalam bentuk Maven Project (kita akan bahas tentang Maven di lain waktu).

Setelah di-import masuk ke dalam NetBeans, kamu akan melihat struktur project seperti ini:





ket.

Source Packages adalah lokasi source code kamu. Nantinya di situ kita akan buat controller, dan lain-lain. Selain itu juga sudah disediakan satu class utama: HelloWorldApplication.java.

Test Packages adalah lokasi source untuk melakukan unit test.

Di Other Source adalah tempat file-file lain selain file Java, biasanya file HTML, CSS, Javascript atau gambar-gambar. Ada satu file penting: application.properties. File ini akan selalu dibaca oleh Spring Boot sebagai file konfigurasi default. Folder webapp/WEB-INF/ adalah tempat kita nantinya meletakkan file-file html.

Lalu ada Dependencies, yaitu lokasi semua dependency yang kita definisikan di dalam file pom.xml.

Runtime Dependencies adalah dependency yang aktif pada saat runtime.

Test Dependencies adalah dependency dalam scope testing.

Java Dependencies berisi dependency default untuk Java, yaitu JDK.

Project Files berisi file pom.xml – konfigurasi dasar untuk project Maven kita. Dan file settings.xml adalah properties Maven untuk project kita.

Selanjutnya, buka HelloWorldApplication.java. Tuliskan code di bawah ini:

```
package com.ichromanrd.apps.hello.helloworld;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HelloWorldApplication {
    public static void main(String[] args) {
        SpringApplication.run(HelloWorldApplication.class, args);
        System.out.println("Hello world!");
    }
}
```



## Servlet & Springboot - Sesi 14

### Debug

[illegible]

Sekarang, coba buat sebuah controller dengan nama HelloController.java.



```
package com.ichromanrd.apps.hello.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HelloController {
    @RequestMapping("/hello")
    public String hello() {
        return "hello";
    }
}
```

Sekarang kita harus membuat konfigurasi ViewResolver, biar controller bisa membaca letak file html kita. Buat file dengan nama MvcConfiguration.java di package config.

```
package com.ichromanrd.apps.hello.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.DefaultServletHandlerConfigurer;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

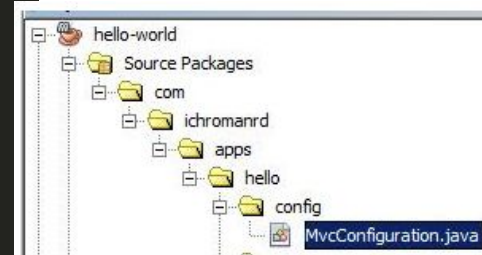
@Configuration
@EnableWebMvc

public class MvcConfiguration extends WebMvcConfigurerAdapter {

    @Bean

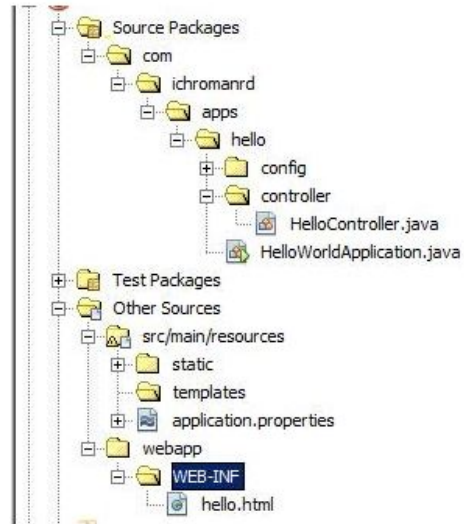
    public InternalResourceViewResolver getViewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/");
        resolver.setSuffix(".html");
        return resolver;
    }

    @Override
    public void configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {
        configurer.enable();
    }
}
```



Lalu coba kita buat file hello.html di folder src/main/webapp/WEB-INF

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello!</title>
  </head>
  <body>
    <h2>Hello World!</h2>
  </body>
</html>
```



## Servlet & Springboot - Sesi 14

# Introduction

Lalu sekarang coba re-run file HelloWorldApplication.java. Dan buka url <http://localhost:8080/hello>.



**Hello World!**

Yeah, Hello World!



# Servlet & Springboot - Sesi 14

## Manual Installation

### Requirements:

#### 1. JDK

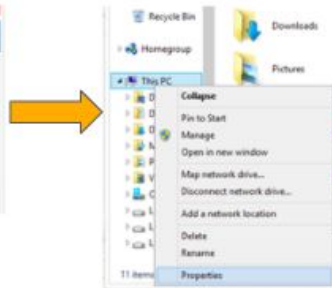
#### 2. XAMPP

#### 3. SpringBoot Tools

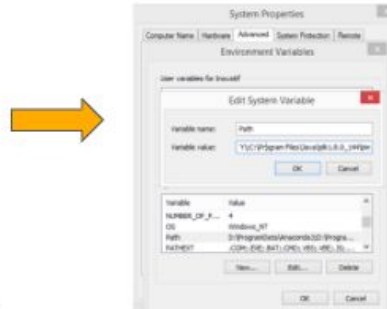
#### 4. Maven

#### JDK

Instalasi : <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



setting path nya dengan cara klik kanan pada bagian "This PC" dan pilih properties.



Klik "Advance system settings" dan pilih "Enviroment Variables". Cari variable path kemudian klik edit dan tambahkan

```
C:\Program Files\Java\jdk1.8.0_144\bin;
```



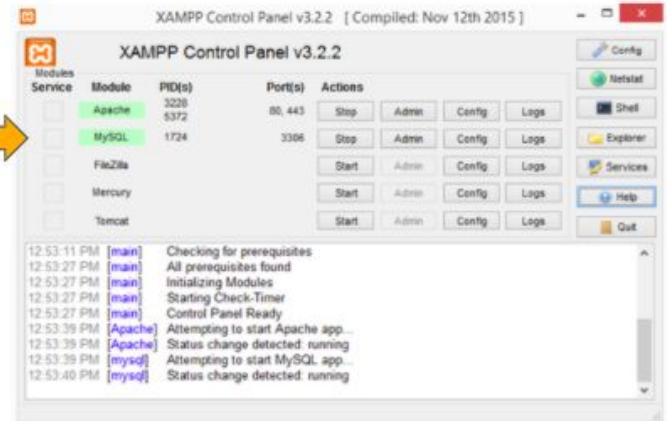
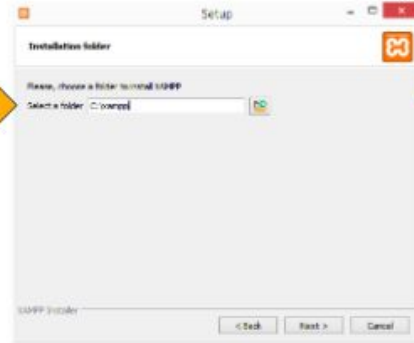
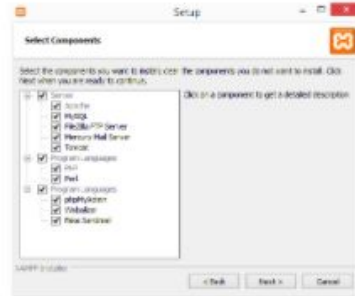
cek dengan cara buka cmd kemudian ketikan "java" atau "javac"



HACKTIV8

# Install XAMPP

<https://www.apachefriends.org/download.html>.



Pilih folder penyimpanan kemudian next hingga selesai.



### 3. Install Spring Boot Tools ( STS )

<https://spring.io/tools/sts/all>

**cttn.**

**Pada dokumentasi ini gunakan STS 3.9.2.RELEASE 64bit**

**Extract file .rar tersebut, kemudian buka direktori `spring-tool-suite-3.9.2.RELEASE-e4.7.2-win32-x86_64 > sts-bundle > sts-3.9.2.RELEASE` dan klik STS.exe**



**HACKTIV8**

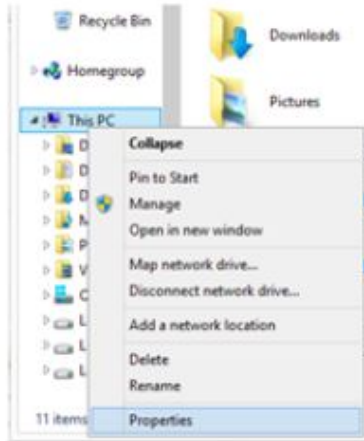
#### 4. Install Maven

<https://maven.apache.org/download.cgi>

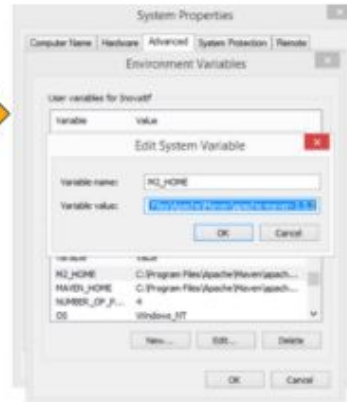
Pada dokumentasi ini gunakan apache-maven-3.5.2.

Set home path maven nya dengan cara extract file .rar tersebut, kemudian pindahkan ke direktori

**C:\Program Files\Apache\Maven\apache-maven-3.5.2**

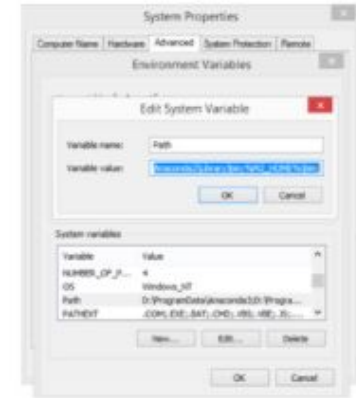


Kemudian setting path nya dengan cara klik kanan pada bagian "This PC" dan pilih properties.



Klik "Advance system settings" dan pilih "Enviroment Variables".  
Tambahkan system variables baru dengan cara klik "new"..  
Pada bagian "variable name" isikan dengan M2\_HOME, dan "variable value" dengan (lokasi home path maven tadi).

**C:\Program Files\Apache\Maven\apache-maven-3.5.2**



Tambahkan variable M2\_HOME tadi ke variable path dengan cara cari variable "path" pada bagian system variable. Kemudian klik edit.  
Tambahkan ";%M2\_HOME%\bin;" tanpa tanda kutip pada bagian variable value kemudian klik ok



**HACKTIV8**

```
Command Prompt

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Inovatif>mvn -v
Apache Maven 3.5.2 (138ed61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T14:58:13+07:00)
Maven home: C:\Program Files\Apache\Maven\apache-maven-3.5.2\bin\..
Java version: 1.8.0_144, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_144\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 8.1", version: "6.3", arch: "amd64", family: "windows"

C:\Users\Inovatif>
```

