

# Training Java DAC

## Dokumentasi Sesi Ke-4

Selasa, 20 Desember 2016

Pada sesi ke-4 kita mulai menggunakan template engine. Template engine berfungsi untuk memasang variabel java dalam halaman HTML. Bisa juga untuk menghasilkan file text, pdf, xml, json dll. Berbagai template engine di java: Jsp, Velocity, Freemarker, Thymeleaf, Jasper Report dll. Untuk aplikasi spring boot kali ini, kita akan menggunakan thymeleaf karena spring boot sudah support secara penuh library ini.

1. Tambahkan dependency thymeleaf di pom.xml Aplikasi Pelatihan

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

2. Update Maven project dependency dengan cara:
  - Klik kanan root project (*Aplikasi Pelatihan [boot]*)
  - *Maven > Update Project...*
3. Edit HelloController.java menjadi seperti di bawah ini

```
@Controller
public class HelloController {

    @RequestMapping("/hellorest")
    @ResponseBody
    public Map<String, Object> helloRest(
        @RequestParam(value = "nama", required = false)
        String nama) {
        Map<String, Object> hasil = new HashMap<>();
        hasil.put("nama", nama);
        hasil.put("foo", "bar");
        hasil.put("waktu", new Date());
        return hasil;
    }

    @RequestMapping("/hello")
    public void helloHtml(
        @RequestParam(value = "nama", required = false)
        String nama, Model hasil) {
        hasil.addAttribute("nama", nama);
        hasil.addAttribute("waktu", new Date());
    }
}
```

4. Buat folder baru di dalam *src/main/resources* dengan nama *templates*
5. Buat file html dengan nama *hello.html* di dalam folder *templates* yang baru dibuat

- Nama file harus sesuai dengan parameter anotasi `@RequestMapping ("/hello")`
- Jadi file nya dinamakan *hello.html*

6. Isi file *hello.html* sbb:

```
<!DOCTYPE HTML>

<html xmlns:th="http://thymeleaf.org">

    <head>

        <title>Aplikasi spring boot</title>

        <meta charset="UTF-8" />

    </head>

    <body>

        <h1> Templating with Tyhmeleaf </h1>

        <ul>

            <li th:text="'Waktu: ' + ${waktu}"></li>

            <li th:text="'Nama: ' + ${nama}"></li>

        </ul>

    </body>

</html>
```

7. Disable caching pada Thymeleaf, edit *application.properties* lalu tambahkan properti sbb:

```
spring.thymeleaf.cache=false
```

8. Buka <http://localhost:8080/hello?nama=budi>



## Templating with tyhmeleaf

- Waktu: Wed Dec 21 13:52:11 ICT 2016
- Nama: budi

Jika pada sesi sebelumnya kita hanya menggunakan Spring Boot untuk menampilkan response dalam bentuk JSON, kali ini kita akan menambahkan view dalam bentuk html menggunakan Thymeleaf, melakukan CRUD dari frontend, validasi data dan melakukan proteksi dengan menambahkan password.

19. Buat class baru di dalam *com.dac.aplikasi.pelatihan.controller* dengan nama *PesertaHtmlController.java*

```
package com.dac.aplikasi.pelatihan.controller;

import java.text.SimpleDateFormat;
import java.util.Date;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.propertyeditors.CustomDateEditor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.dac.aplikasi.pelatihan.dao.PesertaDao;
import com.dac.aplikasi.pelatihan.entity.Peserta;

@Controller
@RequestMapping("/peserta")
public class PesertaHtmlController {

    @Autowired private PesertaDao pd;

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        dateFormat.setLenient(false);
        binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, true));
    }

    @RequestMapping("/list")
    public void daftarPeserta(Model m) {
        m.addAttribute("daftarPeserta", pd.findAll());
    }

    @RequestMapping("/hapus")
    public String hapus(@RequestParam("id") String id) {
        pd.delete(id);
        return "redirect:list";
    }

    @RequestMapping(value = "/form", method = RequestMethod.GET)
    public String tampilkanForm(@RequestParam(value = "id", required = false) String id, Model m) {

        m.addAttribute("peserta", new Peserta());

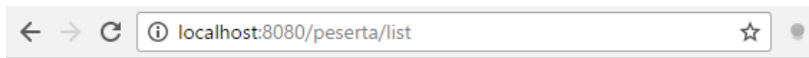
        if (id != null && !id.isEmpty()) {
            Peserta p = pd.findOne(id);
            if (p != null) {
                m.addAttribute("peserta", p);
            }
        }
        return "/peserta/form";
    }

    @RequestMapping(value = "/form", method = RequestMethod.POST)
    public String prosesForm(@Valid Peserta p, BindingResult errors) {
        if (errors.hasErrors()) {
            return "/peserta/form";
        }
        pd.save(p);
        return "redirect:list";
    }
}
```

20. Buat folder baru dengan nama *peserta* di dalam *src/main/resources/templates*
21. Buat file html dengan nama *list.html* di dalam *src/main/resources/templates/peserta*
22. Anotasi `@RequestMapping("/peserta")` memberikan instruksi kepada spring boot untuk mencari view di dalam folder peserta ketika url */peserta/* di akses.
23. Anotasi `@RequestMapping("/list")` mewakili file *list.html* di dalam *src/main/resources/templates/peserta*
24. Berikut ini isi file *list.html*

```
<!DOCTYPE HTML>
<html xmlns:th="http://thymeleaf.org">
  <head>
    <title>Daftar Peserta</title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <h1>Daftar Peserta </h1>
    <table border="1">
      <thead>
        <tr>
          <td>Nama</td>
          <td>Email</td>
          <td>Tanggal lahir</td>
          <td>&nbsp;</td>
        </tr>
      </thead>
      <tbody>
        <tr th:each="p : ${daftarPeserta}">
          <td th:text="${p.nama}">Endy</td>
          <td th:text="${p.email}">Endy@gmail.com</td>
          <td th:text="${p.tanggalLahir}">2015-01-01</td>
          <td>
            <a href="edit.html" th:href="@{/peserta/form(id=${p.id})}">
              edit
            </a> |
            <a href="view.html" th:href="@{/peserta/view(id=${p.id})}">
              view
            </a> |
            <a href="view.html" th:href="@{/peserta/hapus(id=${p.id})}">
              hapus
            </a>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

25. Jika kita kunjungi <http://localhost:8080/peserta/list> maka akan muncul tampilan sbb:



## Daftar Peserta

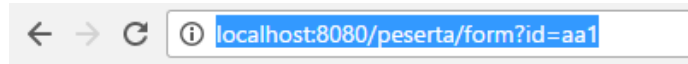
Nama	Email	Tanggal lahir	
Wati Jaya	massagz@gmail.com	2000-01-01	<a href="#">edit</a> <a href="#">view</a> <a href="#">hapus</a>
Peserta Test 1x	peserta.test.001z@gmail.com	2011-01-01	<a href="#">edit</a> <a href="#">view</a> <a href="#">hapus</a>
Peserta Test 002	peserta.test.002@gmail.com	2011-01-02	<a href="#">edit</a> <a href="#">view</a> <a href="#">hapus</a>
Peserta Test 003	peserta.test.003@gmail.com	2011-01-03	<a href="#">edit</a> <a href="#">view</a> <a href="#">hapus</a>

26. Untuk melakukan pengeditan, buat file *form.html* di dalam di dalam folder *src/main/resources/templates/peserta*

27. Berikut ini isi dari *form.html*

```
<!DOCTYPE HTML>
<html xmlns:th="http://thymeleaf.org">
  <head>
    <title>Edit Peserta</title>
    <meta charset="UTF-8" />
  </head>
  <body>
    <h1> Edit Peserta </h1>
    <form action="#" th:action="@{/peserta/form}" th:object="${peserta}" method="post">
      <input type="hidden" th:field="*{id}" />
      <table border="1">
        <tbody>
          <tr>
            <td>Nama</td>
            <td><input type="text" th:field="*{nama}" /></td>
            <td th:if="${#fields.hasErrors('nama')}" th:errors="*{nama}">pesan error</td>
          </tr>
          <tr>
            <td>Email</td>
            <td><input type="text" th:field="*{email}" /></td>
            <td th:if="${#fields.hasErrors('email')}" th:errors="*{email}">pesan error</td>
          </tr>
          <tr>
            <td>Tanggal Lahir</td>
            <td><input type="text" th:field="*{tanggalLahir}" /></td>
            <td th:if="${#fields.hasErrors('tanggalLahir')}" th:errors="*{tanggalLahir}">pesan error</td>
          </tr>
          <tr>
            <td>&nbsp;</td>
            <td><input type="submit" value="simpan" /></td>
          </tr>
        </tbody>
      </table>
    </form>
  </body>
</html>
```

28. Jika kita klik link *edit* di salah satu item pada alamat <http://localhost:8080/peserta/list> maka akan menampilkan alamat baru dengan tampilan sbb:



## Edit Peserta

Nama	Peserta Test 1x
Email	peserta.test.001z@gmail.cor
Tanggal Lahir	2011-01-01
	<input type="button" value="simpan"/>

29. Jika kita kosongkan semua field dan menekan tombol submit maka akan muncul pesan error, karena data yang barusan kita submit divalidasi terlebih dahulu sebelum dimasukkan ke dalam database.



## Edit Peserta

Nama		may not be empty size must be between 3 and 15
Email		may not be empty
Tanggal Lahir		Failed to convert property value of type java.lang.String to required type java.util.Date for property tanggalLahir; nested exception is org.springframework.core.convert.ConversionFailedException: Failed to convert from type [java.lang.String] to type [javax.persistence.Column @javax.persistence.Temporal @javax.validation.constraints.Past @javax.validation.constraints.NotNull java.util.Date] for value '': nested exception is java.lang.IllegalArgumentException
	<input type="button" value="simpan"/>	

30. Selain mengedit, di halaman <http://localhost:8080/peserta/list> kita juga bisa menghapus data, karena sudah ada implementasinya di *PesertaHtmlController.java* sedangkan untuk menu *view* belum bisa dilakukan karena belum ada implementasinya.

31. Menambahkan proteksi menggunakan password, langkah-langkahnya:

- Tambahkan dependency spring-boot-starter-security di pom.xml

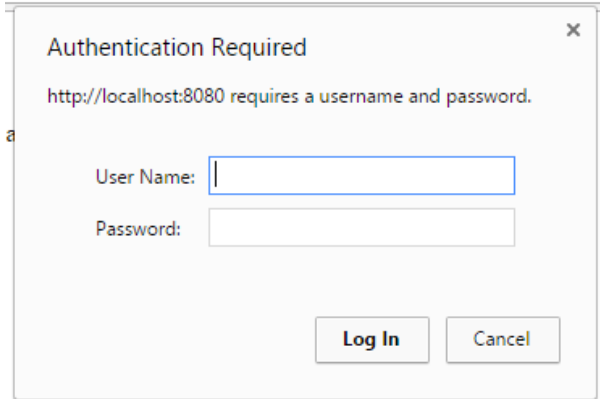
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

32. Update Maven project dependency dengan cara:

- Klik kanan root project (*Aplikasi Pelatihan [boot]*)
- *Maven > Update Project...*

33. Tambahkan properti `security.user.password=abc123` di `application.properties`

34. Jika kita mengunjungi <http://localhost:8080/peserta/list> maka akan muncul login form sbb:



Authentication Required

http://localhost:8080 requires a username and password.

User Name:

Password:

Log In Cancel

35. Untuk login user name default nya adalah `user` dan passwordnya sesuai dengan settingan `security.user.password` di `application.properties`, yaitu `abc123`