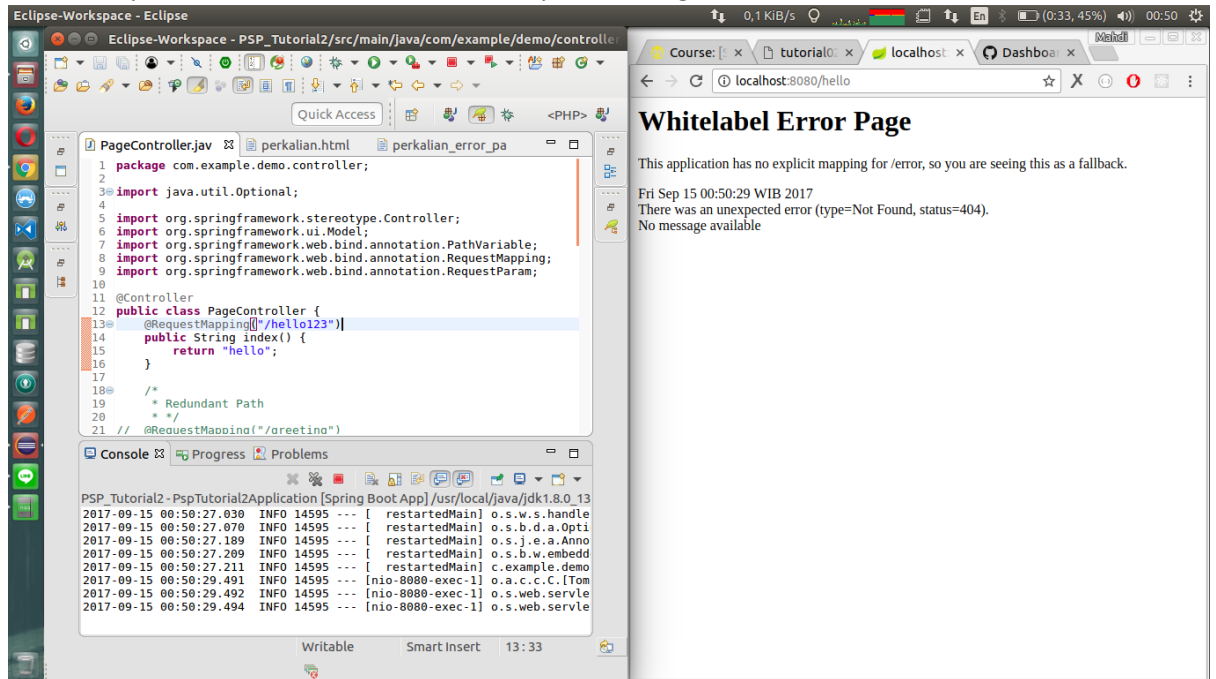


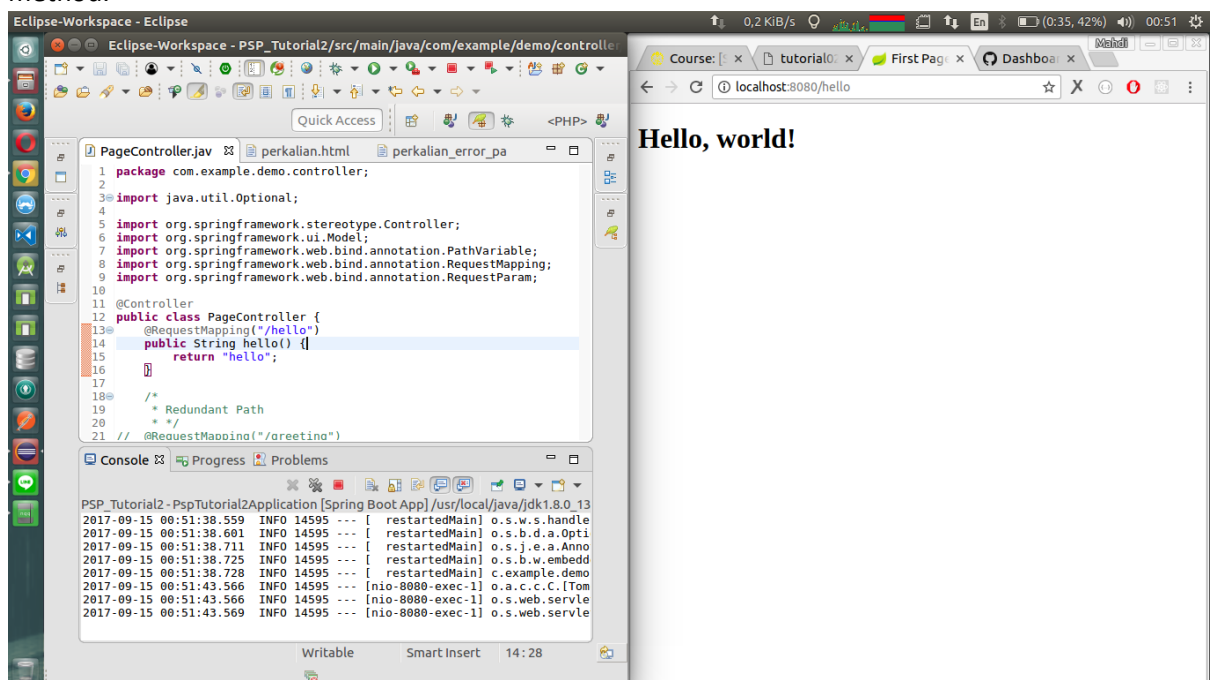
JAWABAN LATIHAN

Latihan Project Hello World

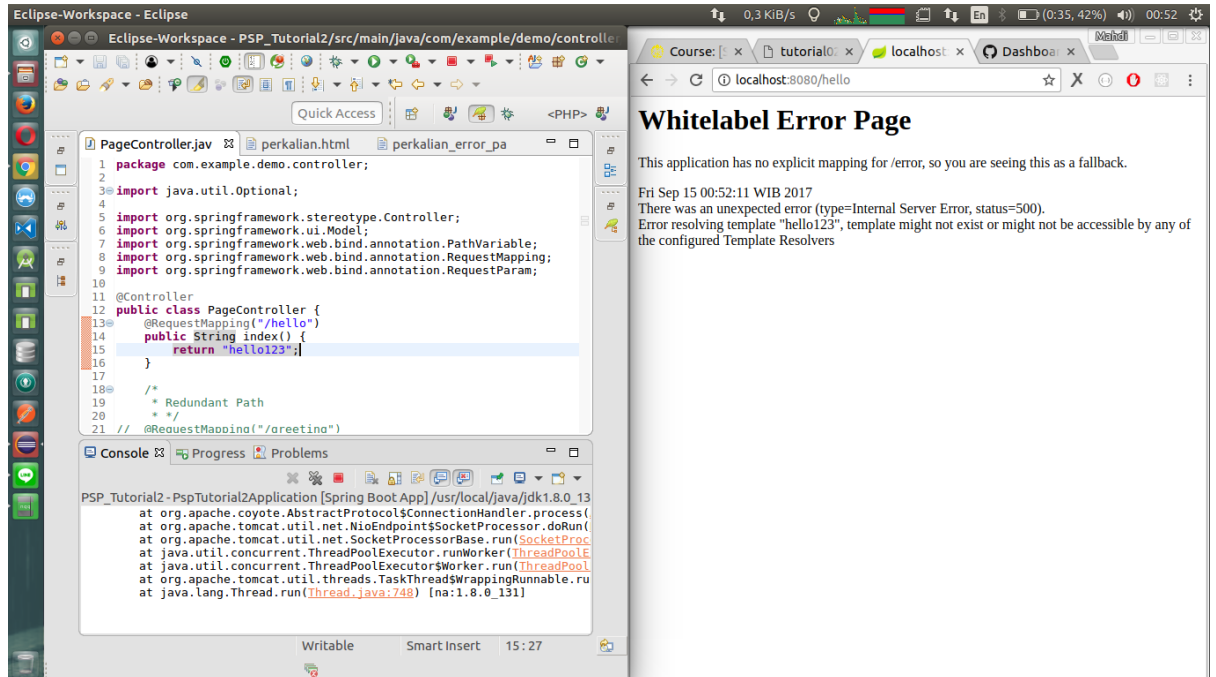
1. Tidak compile error. Error karena tidak terdapat "Routing" untuk /hello.



2. Tidak compile error. Masih bisa berjalan karena `index()` ataupun `hello()` hanya sebuah nama method.

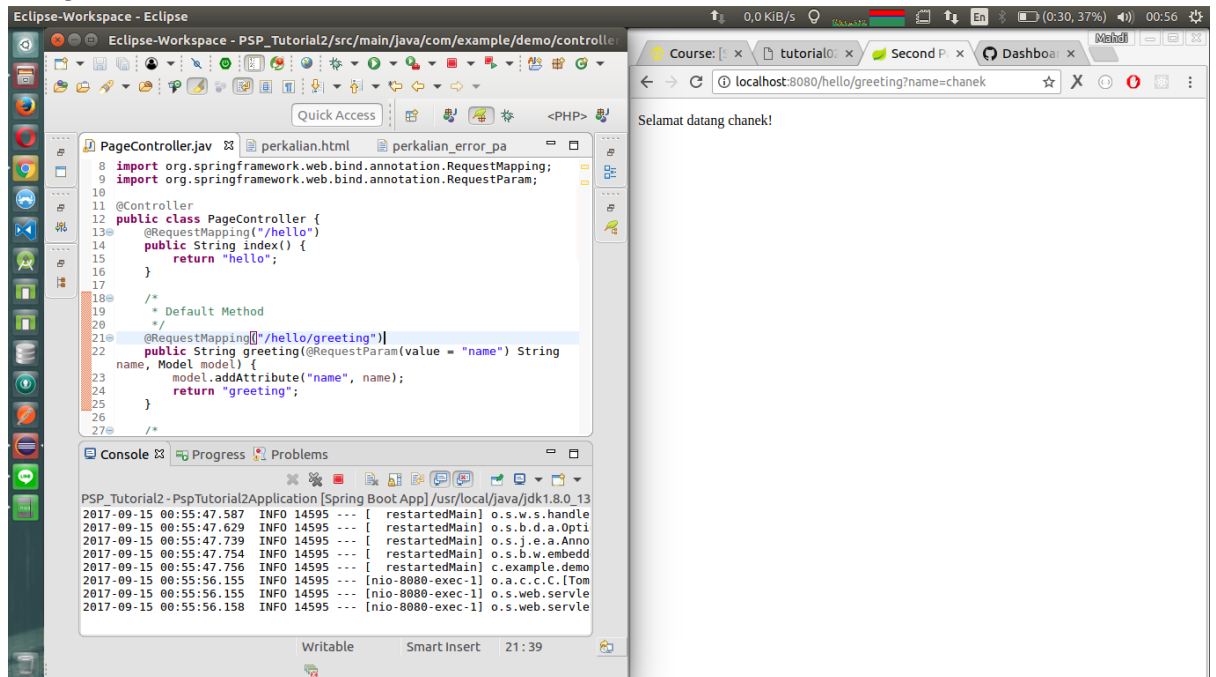


3. Ya, compile error. Menandakan nama file .html pada src/main/resources/templates

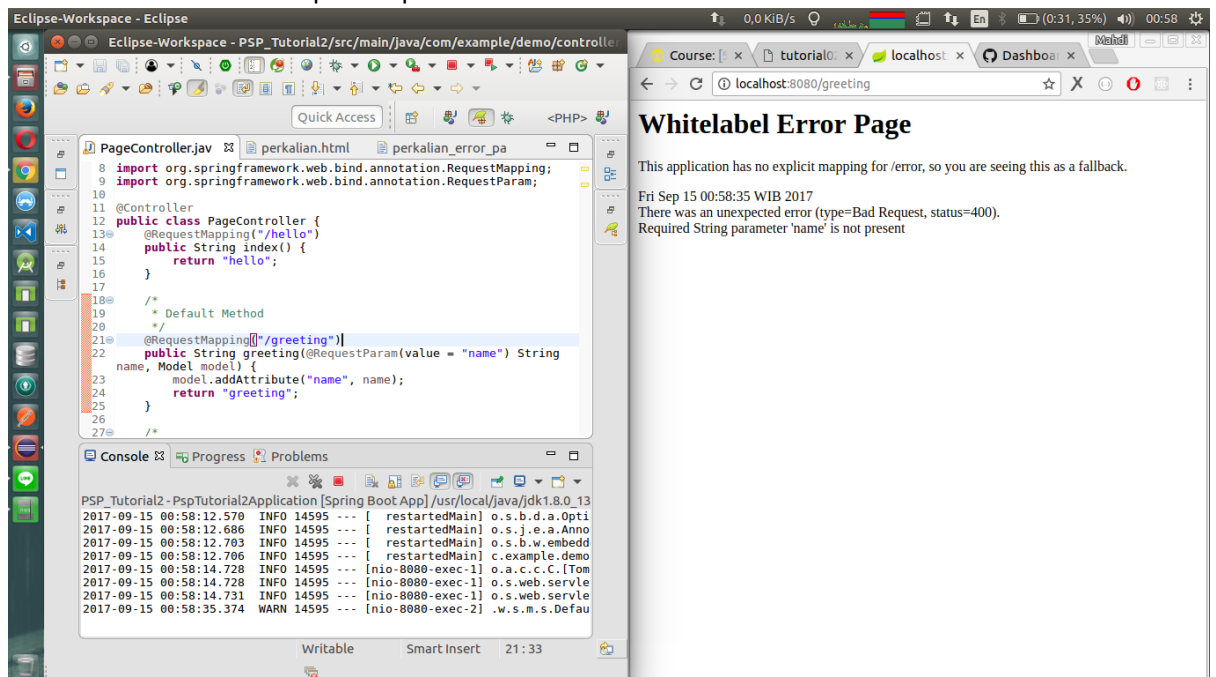


Latihan Request Parameter

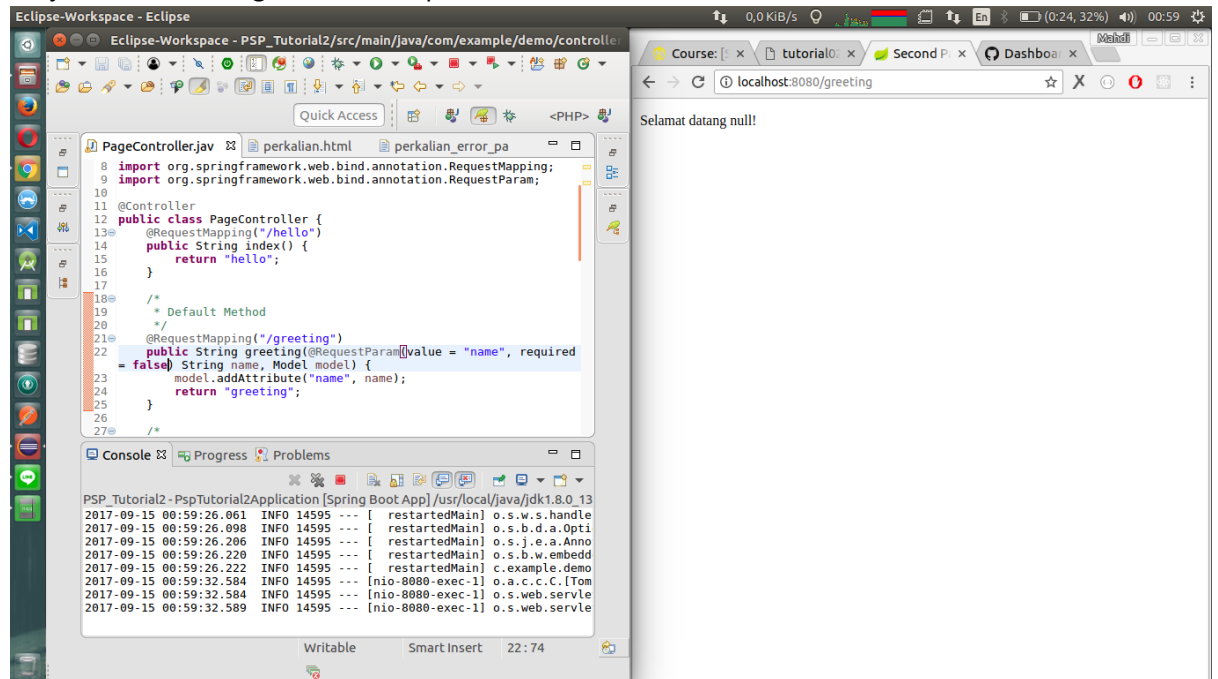
1. Tidak berubah, sama seperti sebelumnya. Karena "Routing" diubah begitu juga dengan cara mengakses di browser.



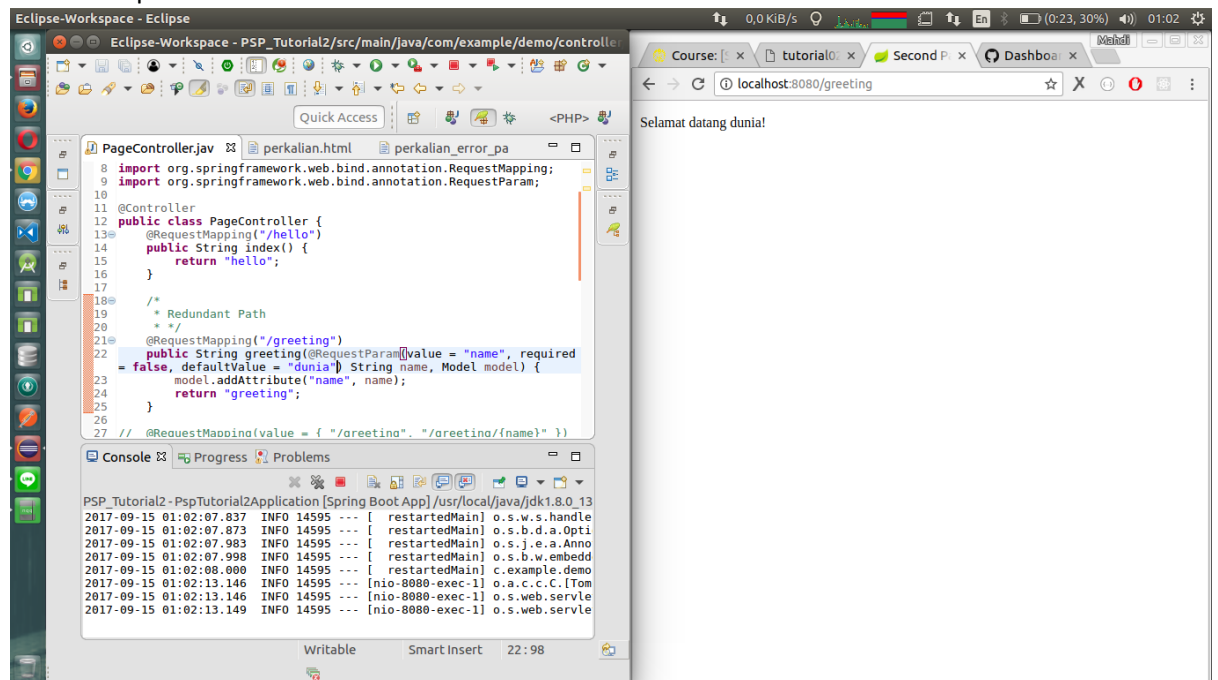
2. Error karena tidak mendapatkan parameter "name".



Berjalan normal dengan default isi parameter "name" = null.



3. Default isi parameter "name" = dunia.



4. Tag Thymeleaf tersebut me-replace "Sapaan untuk user".

Using *th:text* and externalizing text

Externalizing text is extracting fragments of template code out of template files so that they can be kept in separate files (typically `.properties` files) and that they can be easily replaced with equivalent texts written in other languages (a process called internationalization or simply *i18n*). Externalized fragments of text are usually called *"messages"*.

Messages always have a key that identifies them, and Thymeleaf allows you to specify that a text should correspond to a specific message with the `#{...}` syntax:

```
<p th:text="{home.welcome}">Welcome to our grocery store!</p>
```

What we can see here are in fact two different features of the Thymeleaf Standard Dialect:

- The `th:text` attribute, which evaluates its value expression and sets the result as the body of the host tag, effectively replacing the `"Welcome to our grocery store!"` text we see in the code.
- The `{home.welcome}` expression, specified in the *Standard Expression Syntax*, instructing that the text to be used by the `th:text` attribute should be the message with the `home.welcome` key corresponding to whichever locale we are processing the template with.

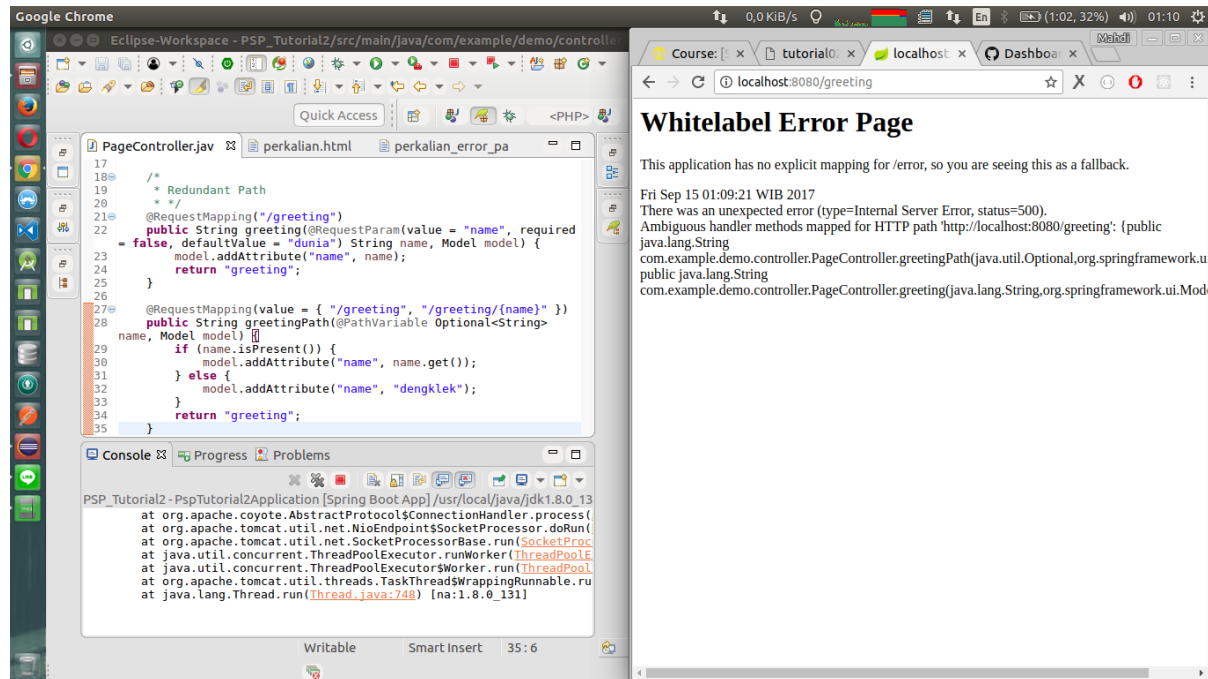
Now, where is this externalized text?

The location of externalized text in Thymeleaf is fully configurable, and it will depend on the specific `org.thymeleaf.messageresolver.IMessageResolver` implementation being used. Normally, an implementation based on `.properties` files will be used, but we could create our own implementations if we wanted, for example, to obtain messages from a database.

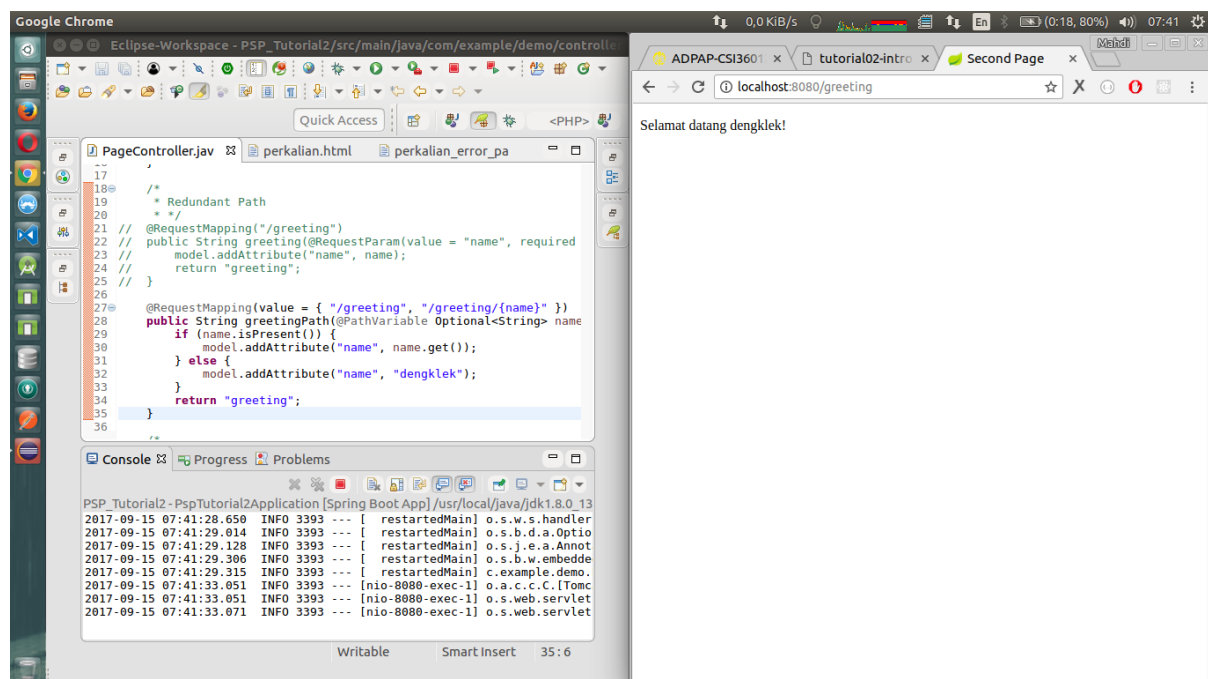
However, we have not specified a message resolver for our template engine during initialization, and that means that our application is using the *Standard Message Resolver* implemented by

Latihan Path Variable

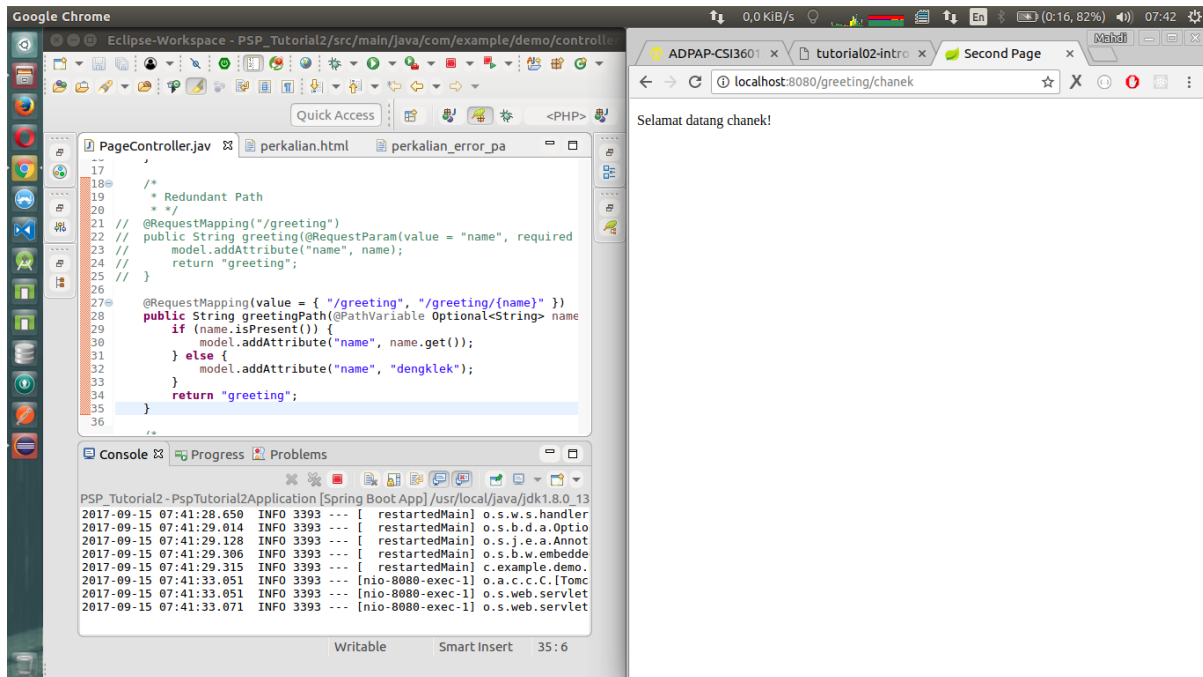
Jika method greeting() masih ada, maka akan terjadi error karena redundansi "Routing".



Jika method greeting() di comment, /greeting akan masuk klausa else karena tidak ada parameter "name",



Sedangkan /greeting/chanek akan masuk klausa if karena terdapat parameter "name" = chanek.



APA YANG DIPELAJARI

- Annotasi `@RequestParam` akan langsung throw exception jika parameter yang ada tidak sesuai dengan apa yang didefinisikan. Misal `param=123asd` sedangkan definisi isi parameter adalah `int`, maka akan langsung throw exception bahkan sebelum memasuki isi method.
- Cara terbaik untuk memvalidasi input dari `@RequestParam` adalah dengan menerimanya sebagai `String`, lalu mengolahnya di dalam method, baik dengan regex atau validasi input lainnya. *CMIW