Web-based information systems 1

Getting started
in IntelliJ Idea
with dynamic web projects
using
Thymeleaf

Michiel Noback (NOMI)
Institute for Life Sciences and Technology
Hanze University of Applied Sciences

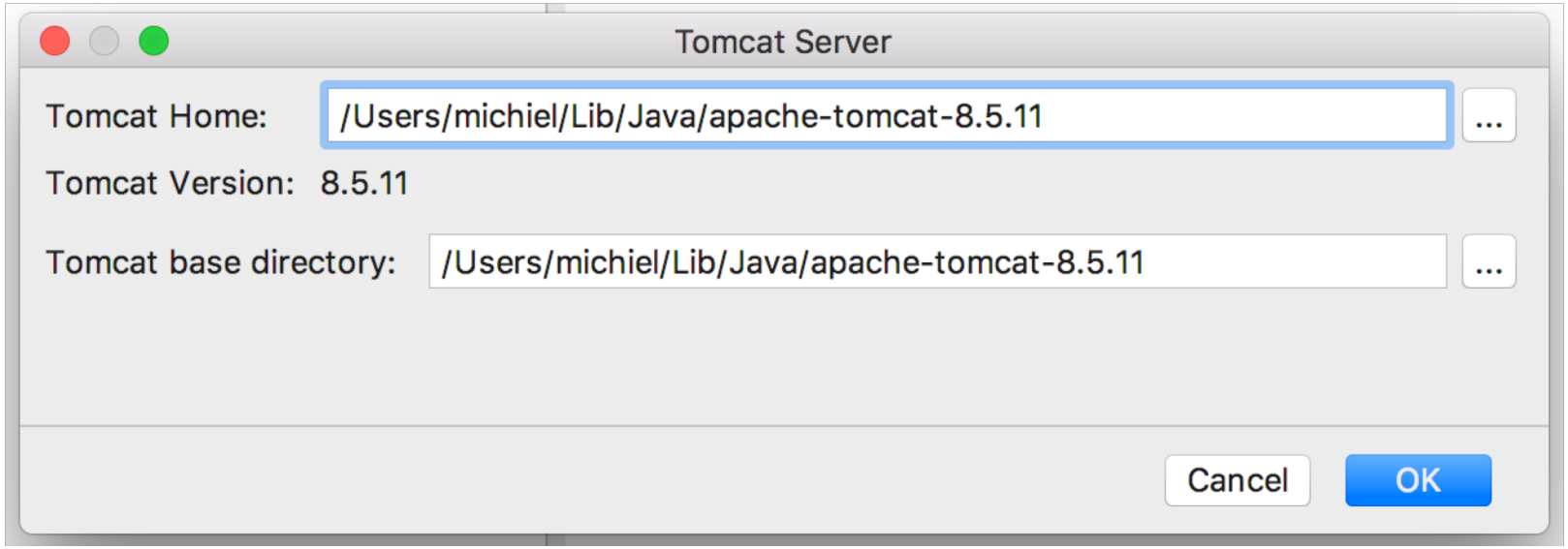
introduction

Getting started with a web app that serves an internationalized welcome message involves quite some steps:

- configuring IntelliJ with the application server Tomcat
- creating a web project
- creating a servlet
- creating a Thymeleaf template
- Creating a resource bundle
- running the web application
- This material is also presented in a blog: http://michielnoback.nl/java-web-project-with-servlets-thymeleaf/

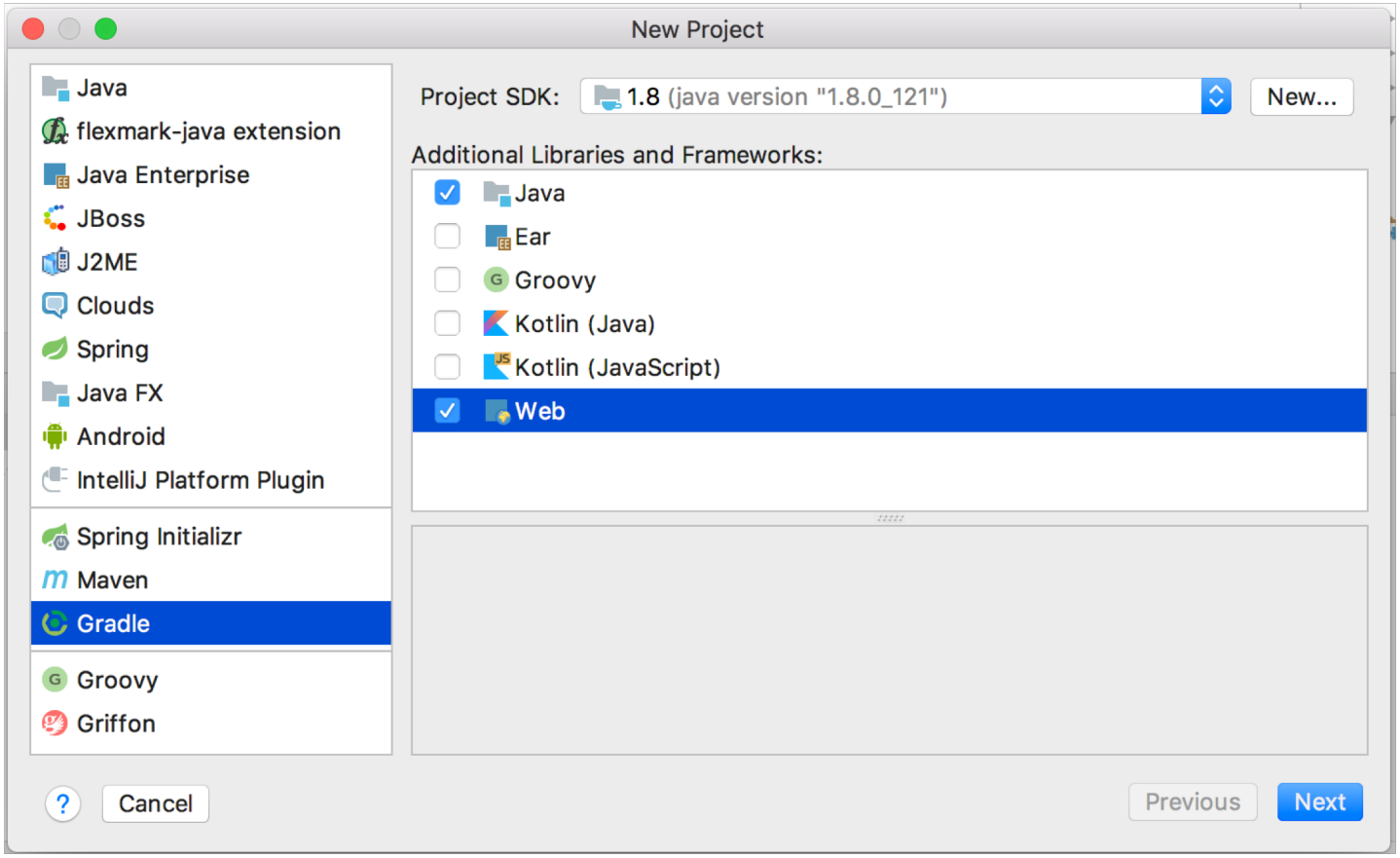
Configuring IntelliJ with Tomcat

- After installation (extraction) of Tomcat, you'll have to let IntelliJ know where it is
- You do this via Preferences -> Build, Execution, Deployment -> Application Servers -> click '+' (Add)



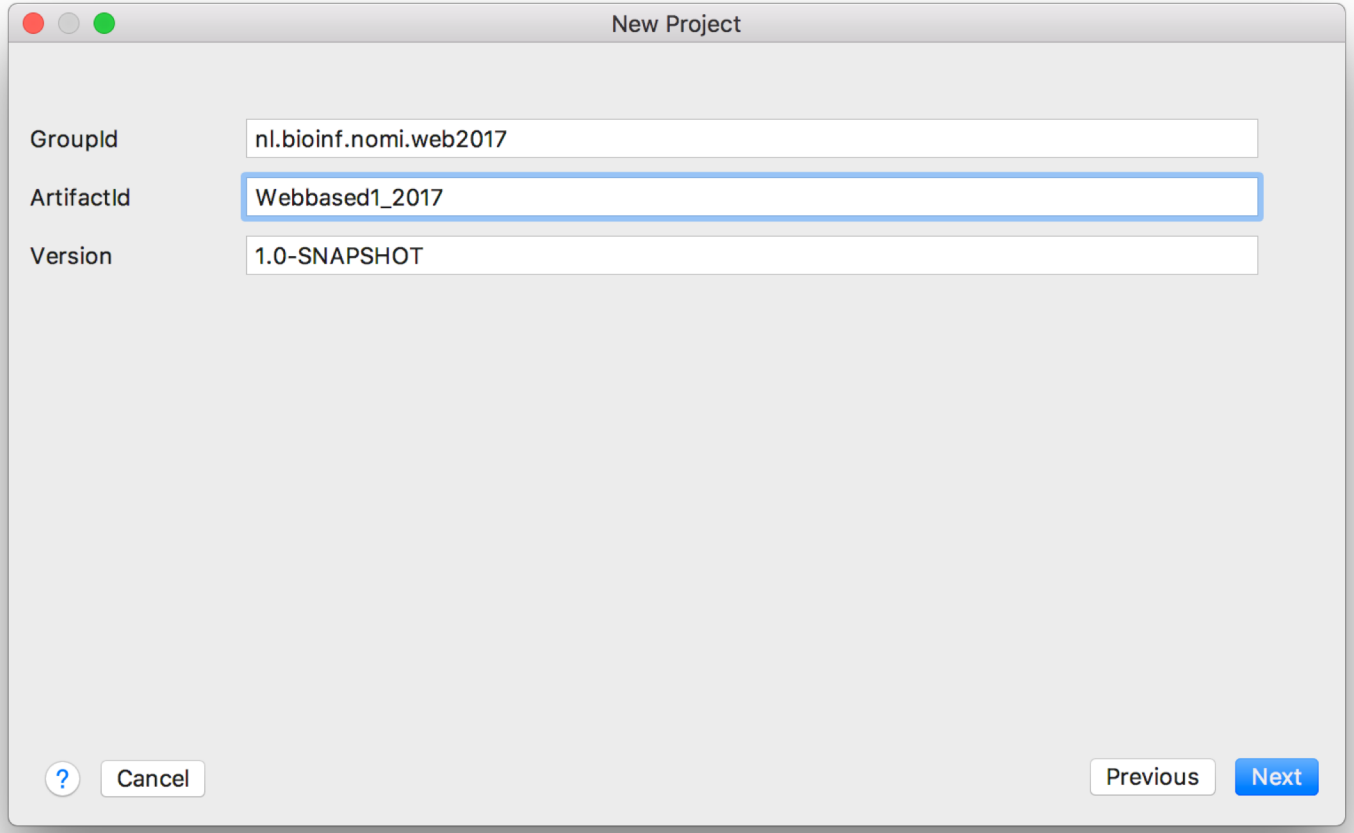
Creating a Gradle-managed web project in IntelliJ (1)

- Choose File -> New -> Project
- Choose Gradle -> + Java + Web -> Next



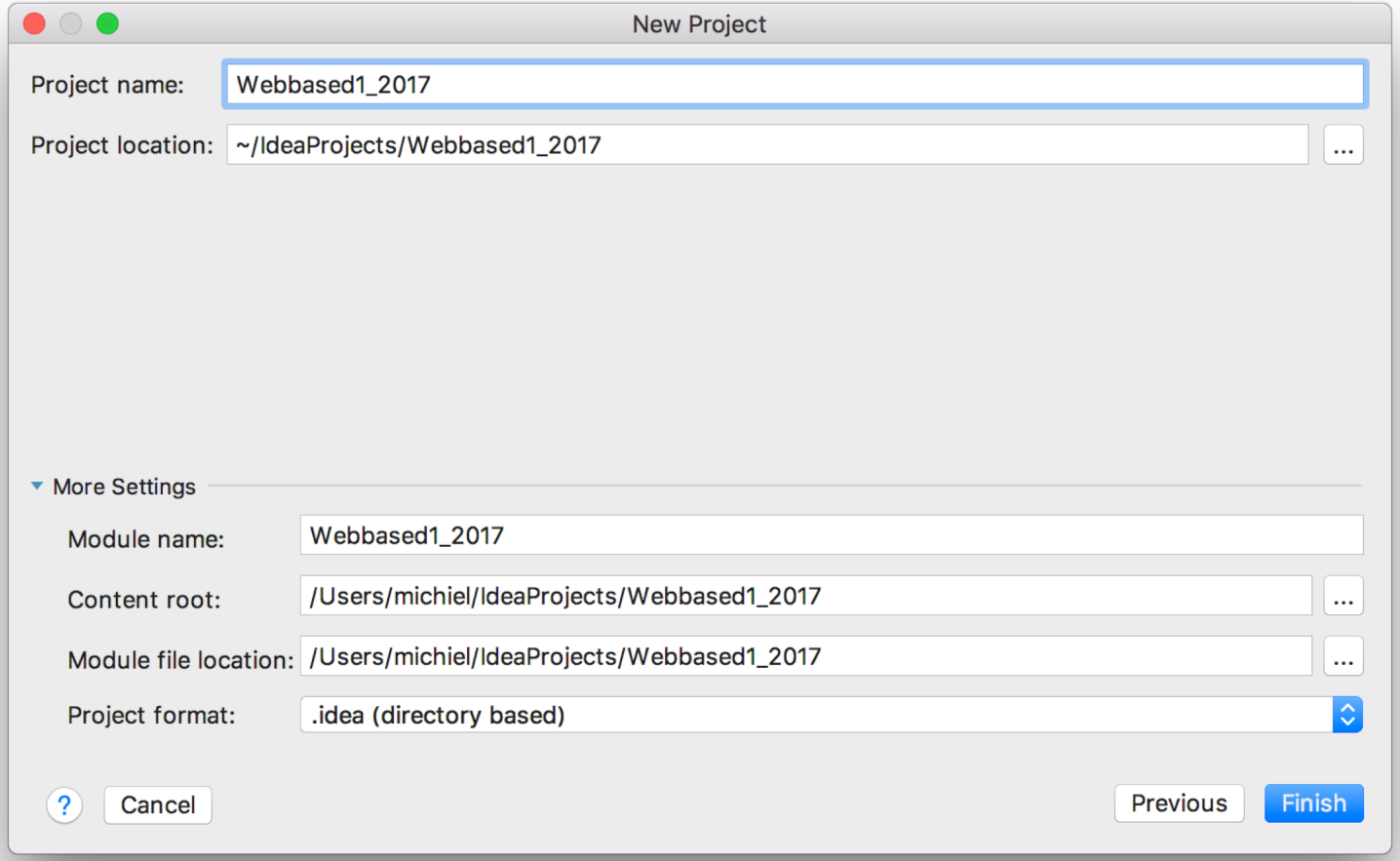
Creating Gradle web project (2)

- Enter GroupId, ArtifactId and Version → Click Next



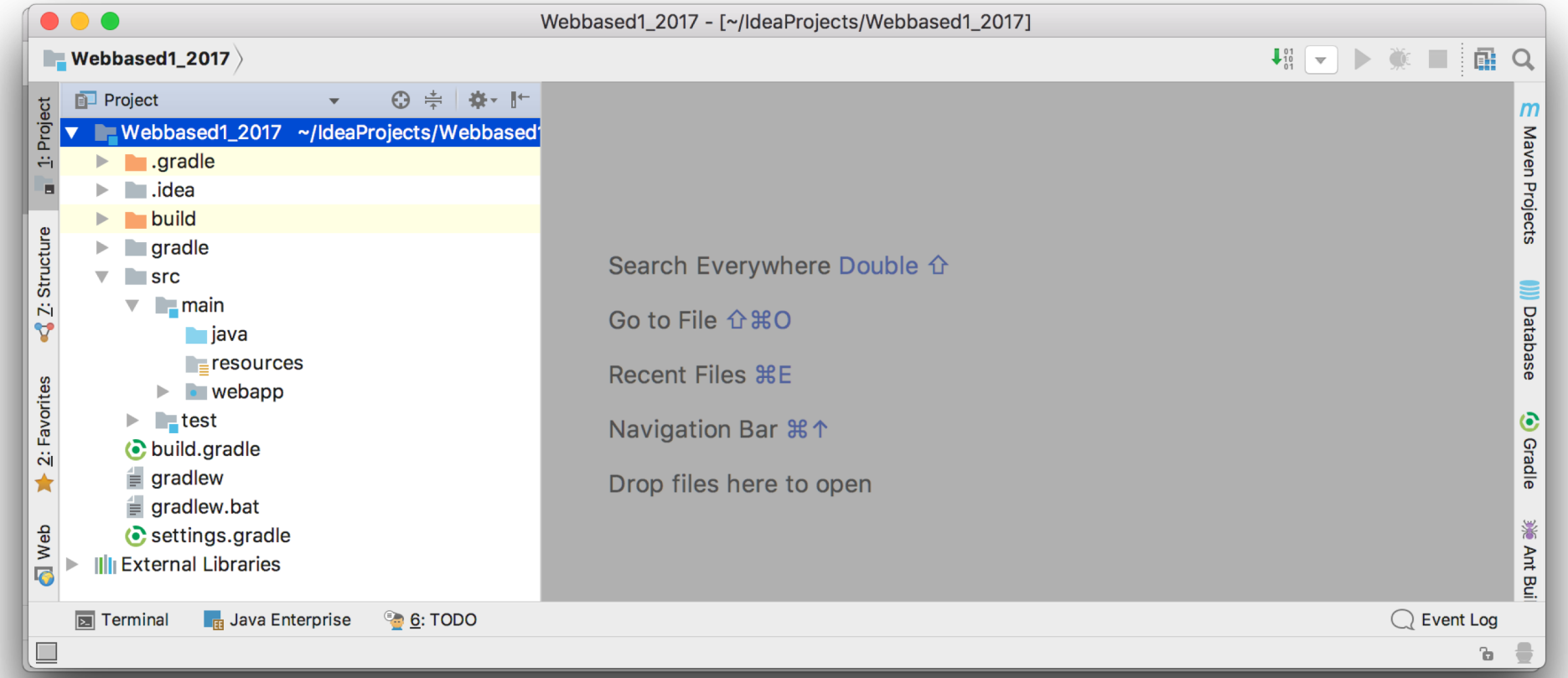
Creating Gradle web project (3)

- Check “Use auto-import” → Click Next → Click Finish



Creating Gradle web project (4)

- You will have something like this



Preparing to use Thymeleaf

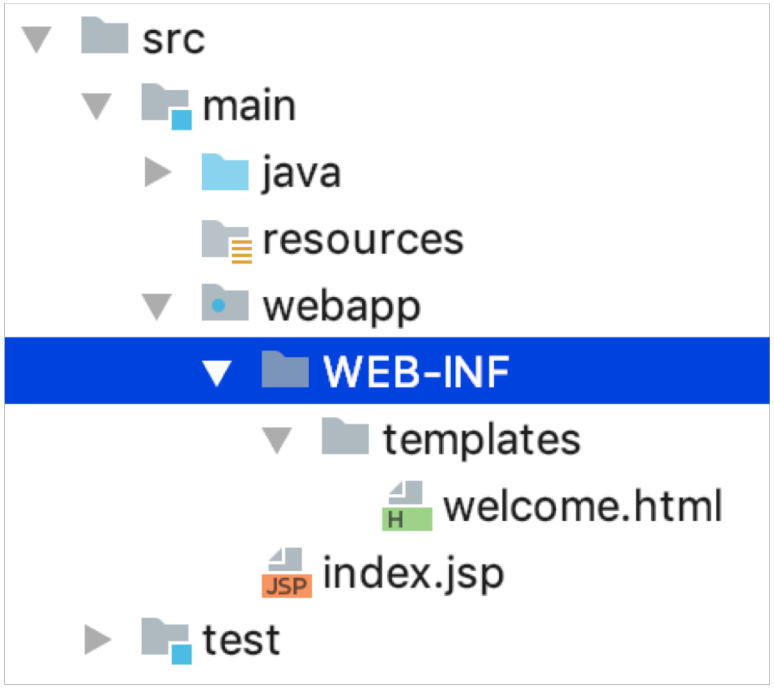
- Since we’re going to use Thymeleaf instead of JSP as templates, and and JUnit 5 as testing framework, some additional configuration is required
- Hang in there!

Configure `build.gradle`

```
plugins {
    id 'java'
    id 'war'
    id 'idea'
}

group 'nl.bioinf.wis_thymeleaf'
version '1.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    compile 'javax.servlet:javax.servlet-api:3.1.0'
    compile 'org.thymeleaf:thymeleaf:3.0.0.ALPHA'
    testCompile 'org.junit.jupiter:junit-jupiter-engine:5.1.0'
}
```

Create a WEB-INF folder

- Under `webapp`, create a folder called `WEB-INF` and within that one a folder called `templates`.
- Within `templates`, create the html file `welcome.html` shown on the next slide



welcome.html

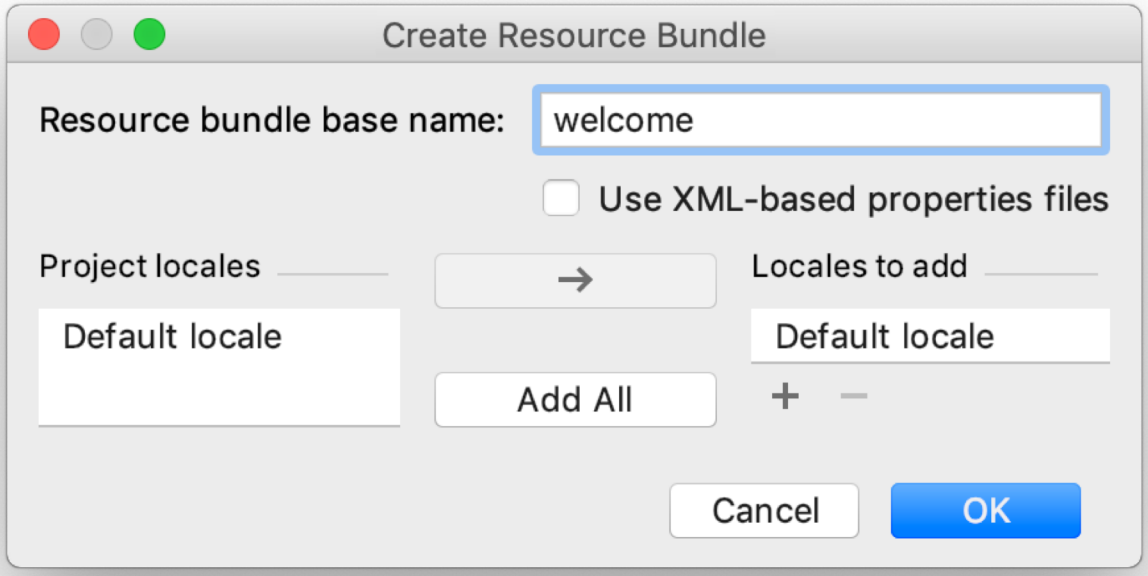
```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-
thymeleaf-4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Thymeleaf Starter</title>
    <meta http-equiv="Content-Type" content="text/html;
          charset=UTF-8" />
</head>
<body>
    <!-- welcome message read from resource bundle welcome.html-->
    <p th:text="#{welcome.msg}">Welcome Offline</p>

    <!-- date forwarded from the servlet-->
    <p>Date: <span th:text="${currentDate}">Sat Oct 24
2015</span></p>
</body>
</html>
```

#{…} and ${…}

- **`#{...}`** fetches the value from the resource bundle (files named `xxxxx.properties`).
  - The expression *`#{welcome.msg}`* gets the value of key *`welcome.msg`* from the file `welcome_xx.properties`
- **`${...}`** this expression fetches the value set in the `WebContext` instance.
  - The expression*`${currentDate}`* gets the value for the attribute *currentDate* set in WebContext.
- Much more on this later!

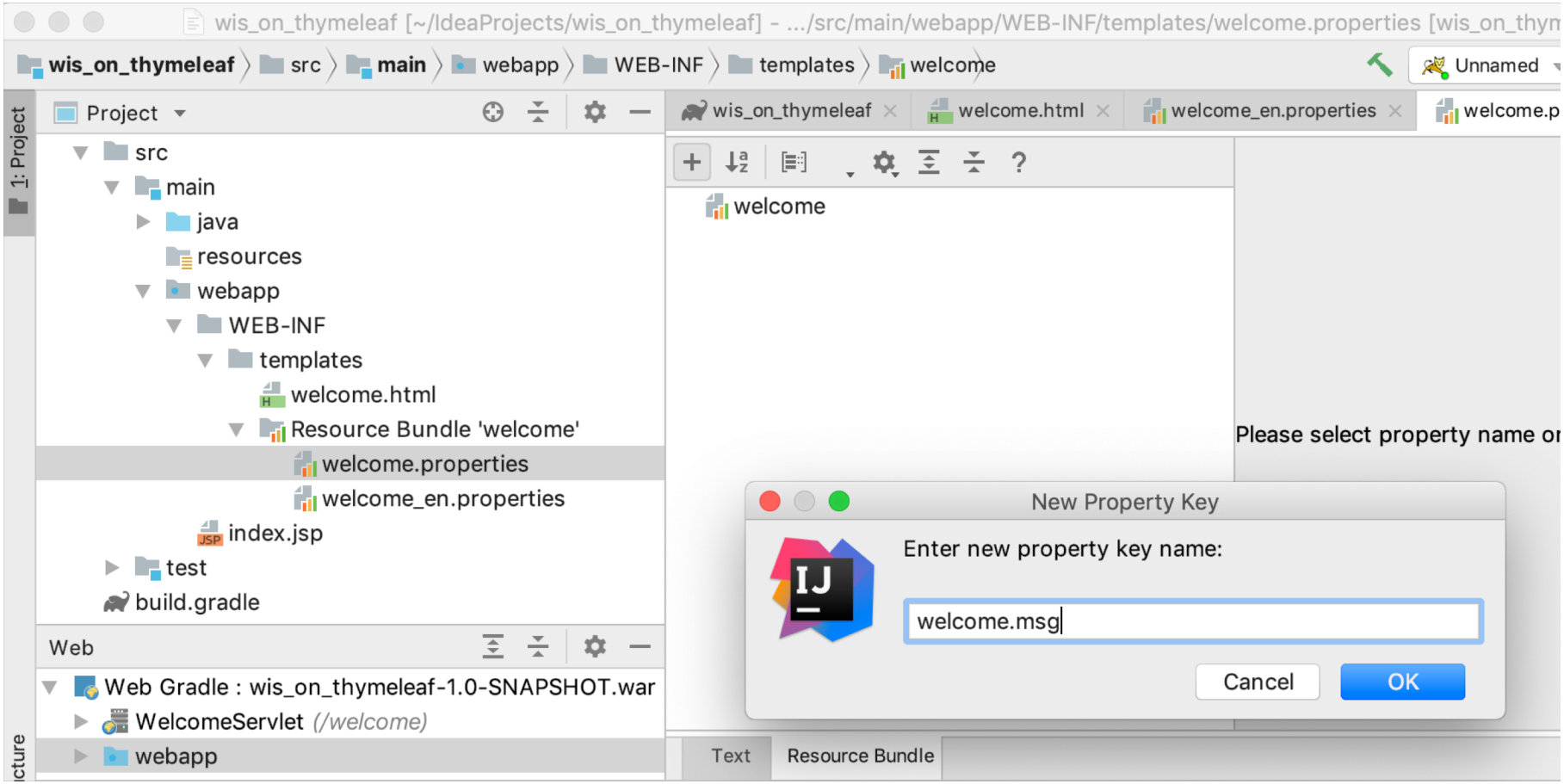
Create a resource bundle

- Right-click the templates folder, select New -> Resource Bundle
- Type `welcome` as base name
- Click '+' under 'Locales to add' and add 'en'
- Click OK



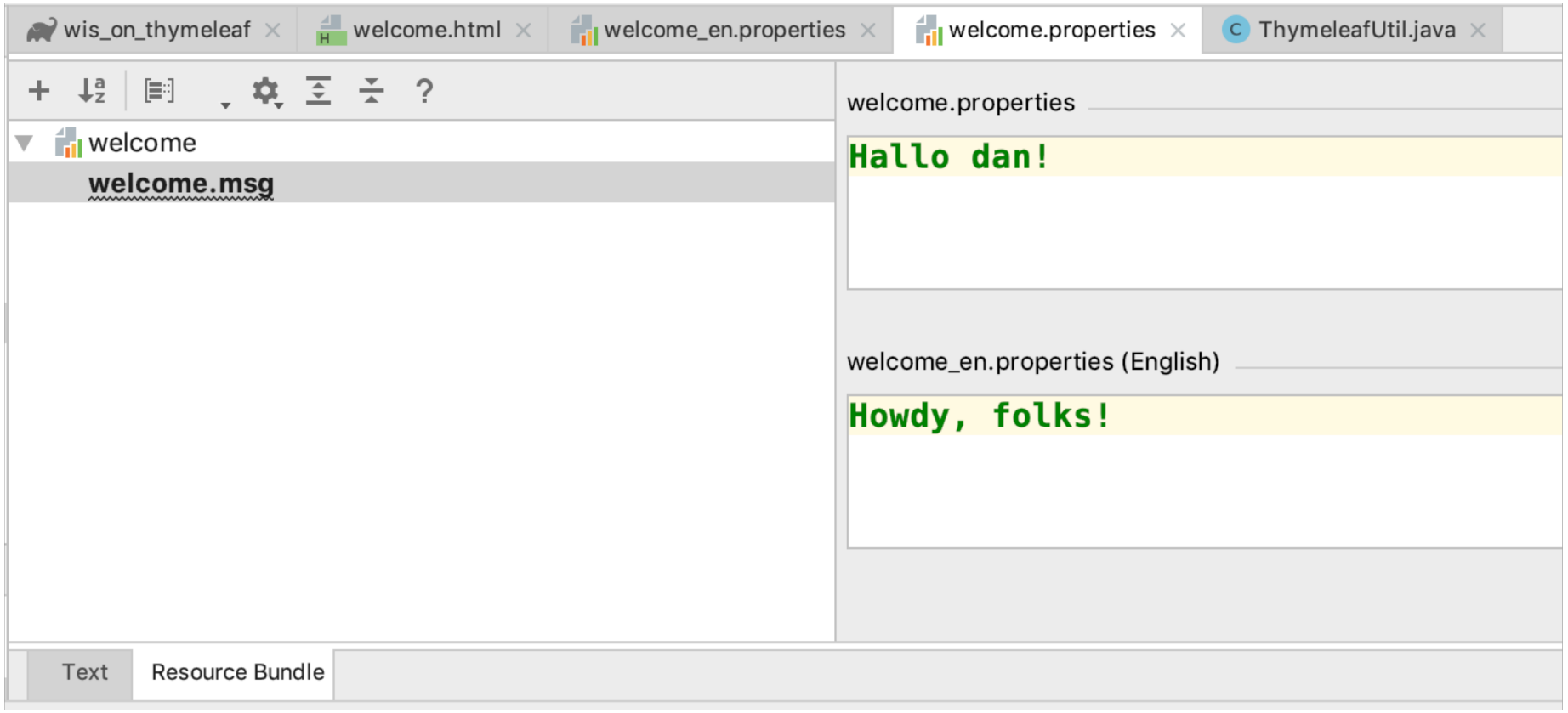
Add to resource bundle

- In resource bundle called ‘welcome’, click ‘+’ to add a new Property key and name it welcome.msg



Add to resource bundle (2)

- Give some Dutch and English values for `welcome.msg`



Create Java packages

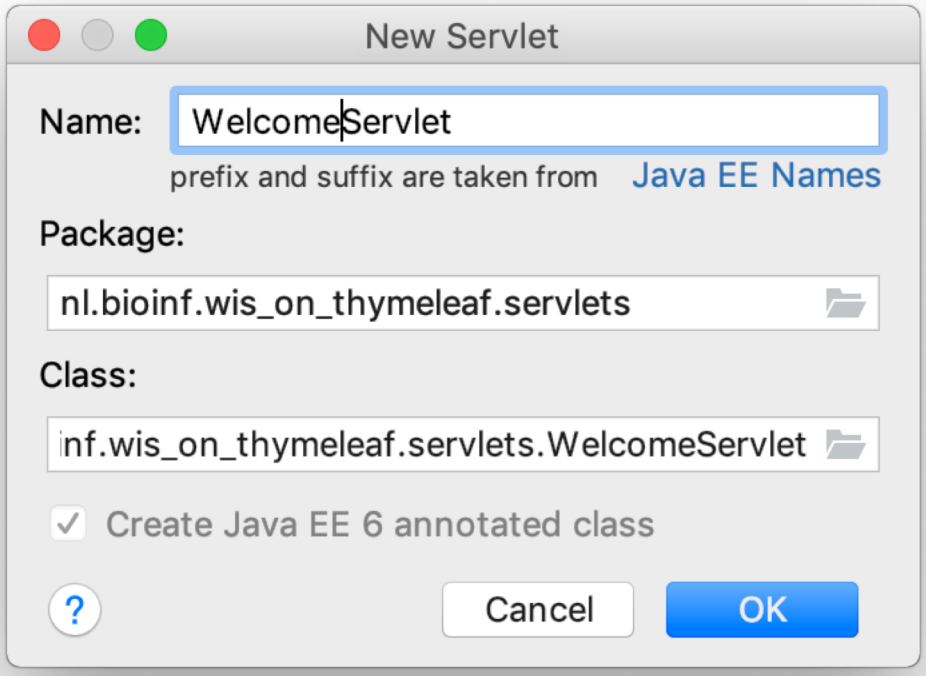
- Under main/java, create a base package, e.g. `nl.bioinf.<your_name>.wis` with two packages:
  - config
  - servlets
- Under config, create a new Java class file `ThymeleafUtil.java`
- Under servlets, create a new Java class file `WelcomeServlet.java`

Class ThymeleafUtil.java

```
package nl.bioinf.wis_on_thymeleaf.config;
//imports omitted
public class ThymeleafUtil {
    private static TemplateEngine templateEngine;

    public static void createTemplateEngine(ServletContext servletContext) {
        ServletContextTemplateResolver templateResolver =
                new ServletContextTemplateResolver(servletContext);
        templateResolver.setTemplateMode("XHTML");
        templateResolver.setPrefix("/WEB-INF/templates/");
        templateResolver.setSuffix(".html");
        templateResolver.setCacheTTLMs(3600000L);
        templateResolver.setCacheable(true);
        templateEngine = new TemplateEngine();
        templateEngine.setTemplateResolver(templateResolver);
    }

    public static TemplateEngine getTemplateEngine() {
        return templateEngine;
    }
    //method omitted
}
```

Create a servlet

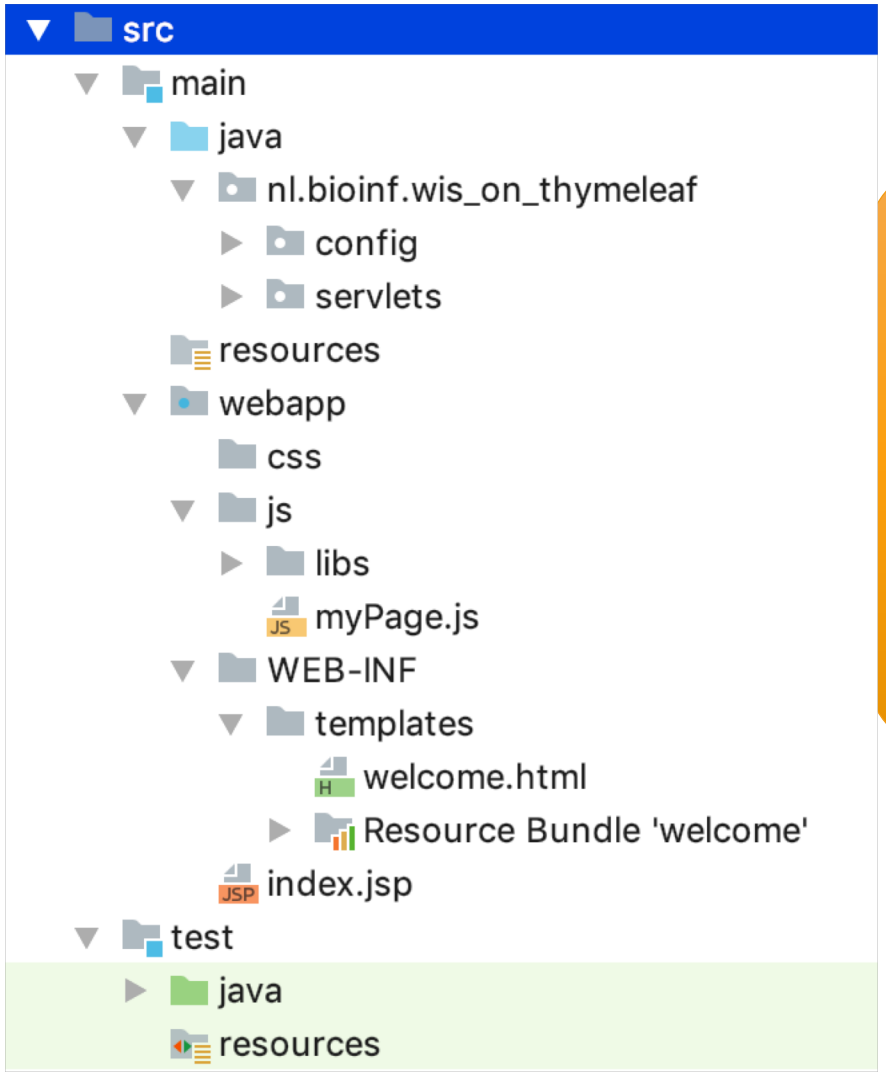
- Right-click the servlets package → New → Servlet → Give it a name and click OK



Class `WelcomeServlet.java`

```java
package nl.bioinf.wis_on_thymeleaf.servlets;

import ...

@WebServlet(name = "WelcomeServlet", urlPatterns = "/welcome", loadOnStartup = 1)
public class WelcomeServlet extends HttpServlet {
    private TemplateEngine templateEngine;

    @Override
    public void init() throws ServletException {
        final ServletContext servletContext = this.getServletContext();
        this.templateEngine = ThymeleafUtil.createTemplateEngine(servletContext);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
        WebContext ctx = new WebContext(
                request,
                response,
                request.getServletContext(),
                request.getLocale());
        ctx.setVariable("currentDate", new Date());
        templateEngine.process("welcome", ctx, response.getWriter());
    }
}
```

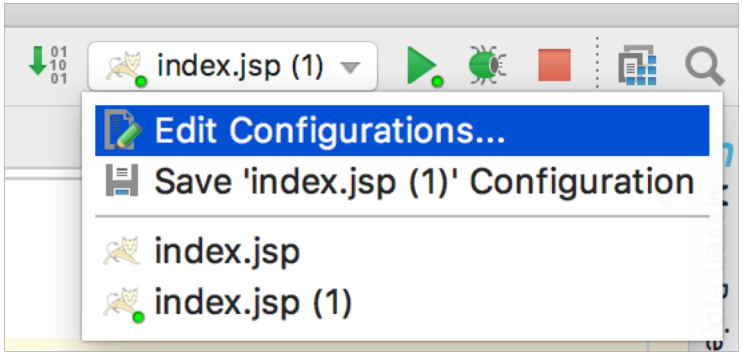
The **src** structure



- The directory src holds ALL your code
- main is for production, test for JUnit test code
- main/java holds java code
- main/resources will hold files not deployed with the app
- main/webapp will hold all web content
- organize css and js files in their own folders

Deploy (1)

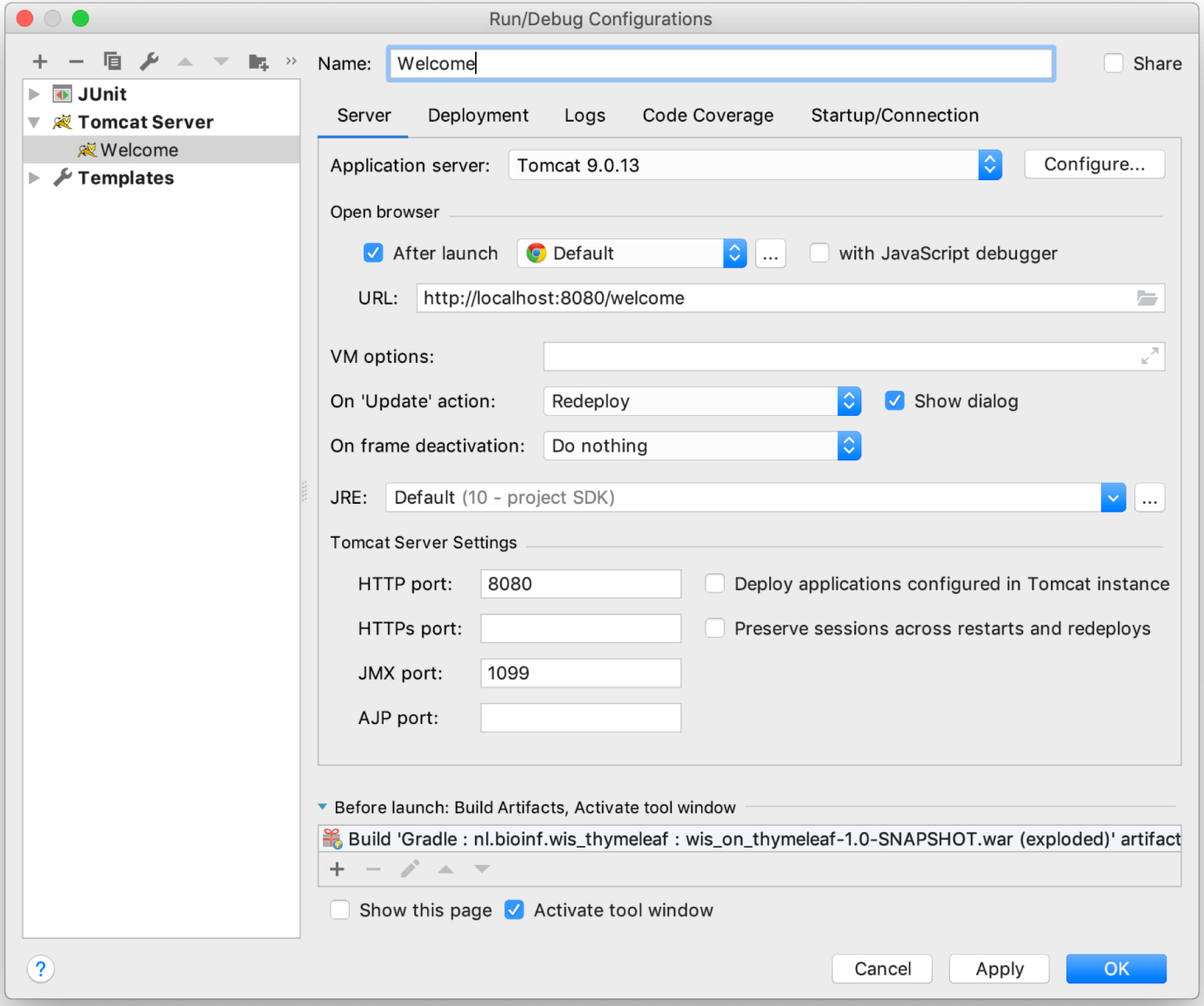
- Create a Run Configuration (Edit Configurations…)



Deploy (2)

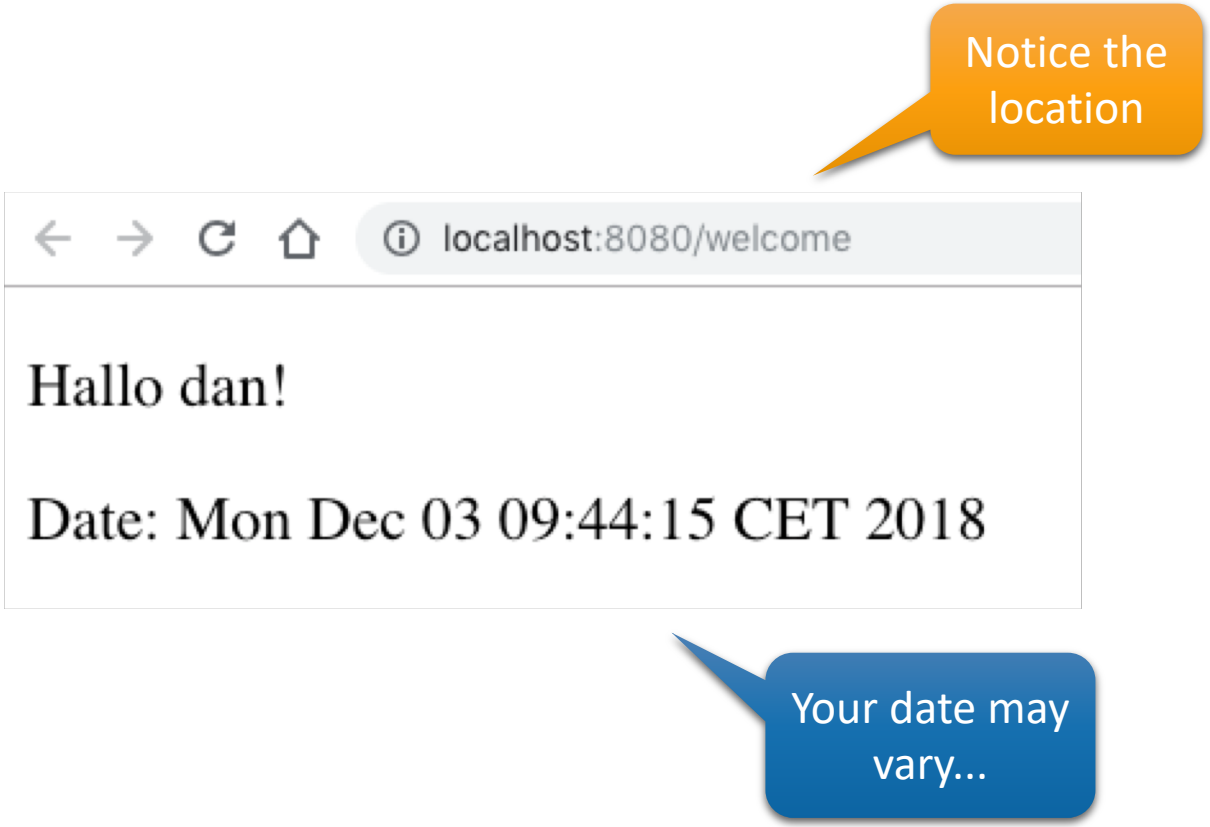
- New Configuration → Tomcat Server → Local
- Give it a name and default behaviour

Deploy (3)


Run/Debug Configurations
JUnit
Tomcat Server
Welcome
Templates
Name: Welcome
Share
Server
Deployment
Logs
Code Coverage
Startup/Connection
Application server: Tomcat 9.0.13
Configure...
Open browser
After launch
Default
with JavaScript debugger
URL: http://localhost:8080/welcome
VM options:
On 'Update' action: Redeploy
Show dialog
On frame deactivation: Do nothing
JRE: Default (10 - project SDK)
Tomcat Server Settings
HTTP port: 8080
Deploy applications configured in Tomcat instance
HTTPs port:
Preserve sessions across restarts and redeploys
JMX port: 1099
AJP port:
Before launch: Build Artifacts, Activate tool window
Build 'Gradle : nl.bioinf.wis_thymeleaf : wis_on_thymeleaf-1.0-SNAPSHOT.war (exploded)' artifact
Show this page
Activate tool window
Cancel
Apply
OK

Deploy (4)

Run it and view



Deploy (5)

Go to chrome://settings/languages and change your language to English & refresh the view