Developing Java EE Application

Jeongwhan Choi · Follow
5 min read · Sep 20, 2018

Creating a project

1. Click Create New Project on the Welcome screen, or select **File | New | Project**. The New Project wizard opens.

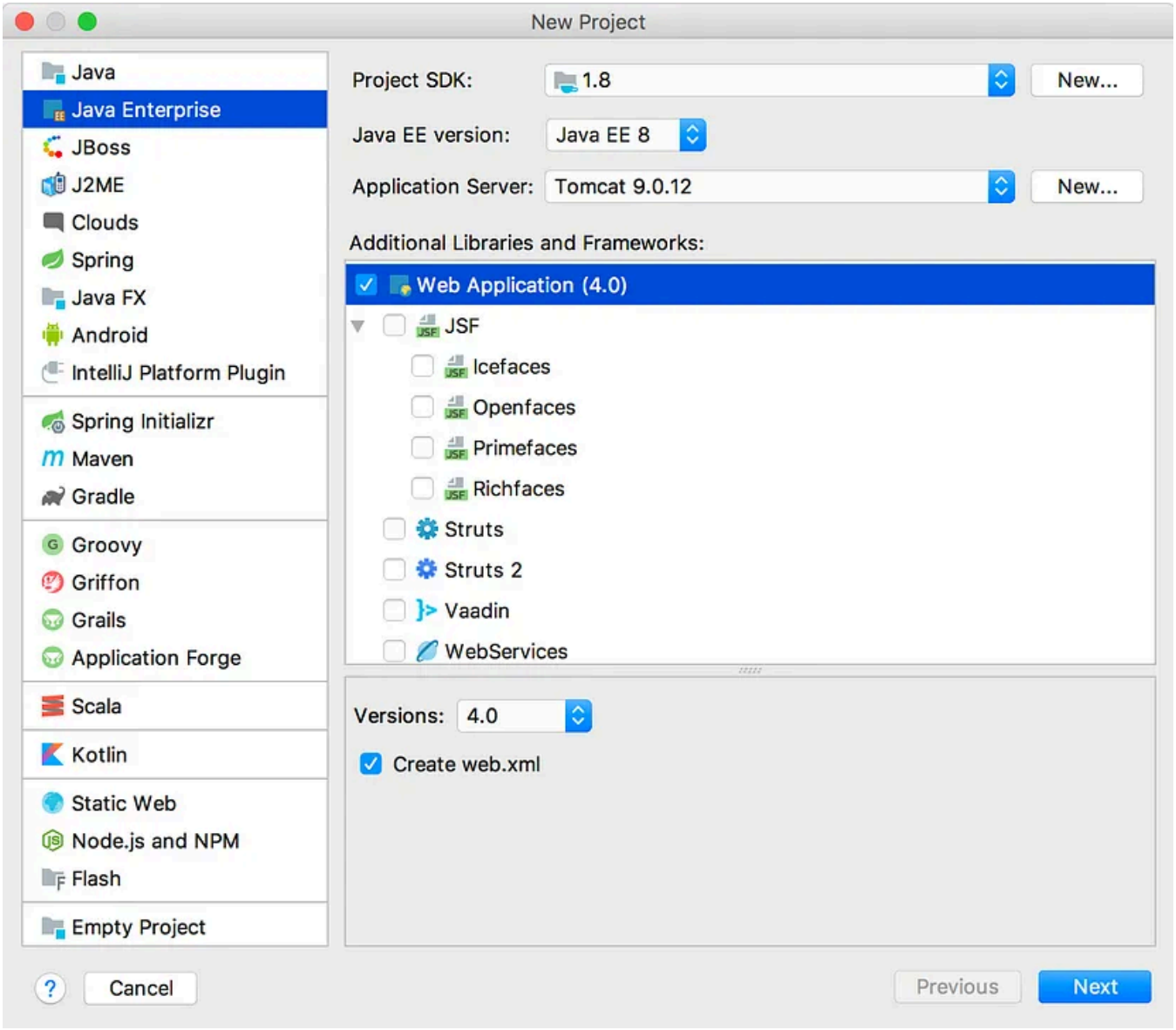
2. In the left-hand pane, select Java Enterprise.

3. Specify the JDK that you are going to use (the Project SDK field): select one from the list, click New and select the JDK installation folder, or click Download JDK.

4. Specify your application server. (We'll use Tomcat Server.) If Tomcat is not defined in IntelliJ IDEA yet, click New to the right of the Application Server field and select Tomcat Server.

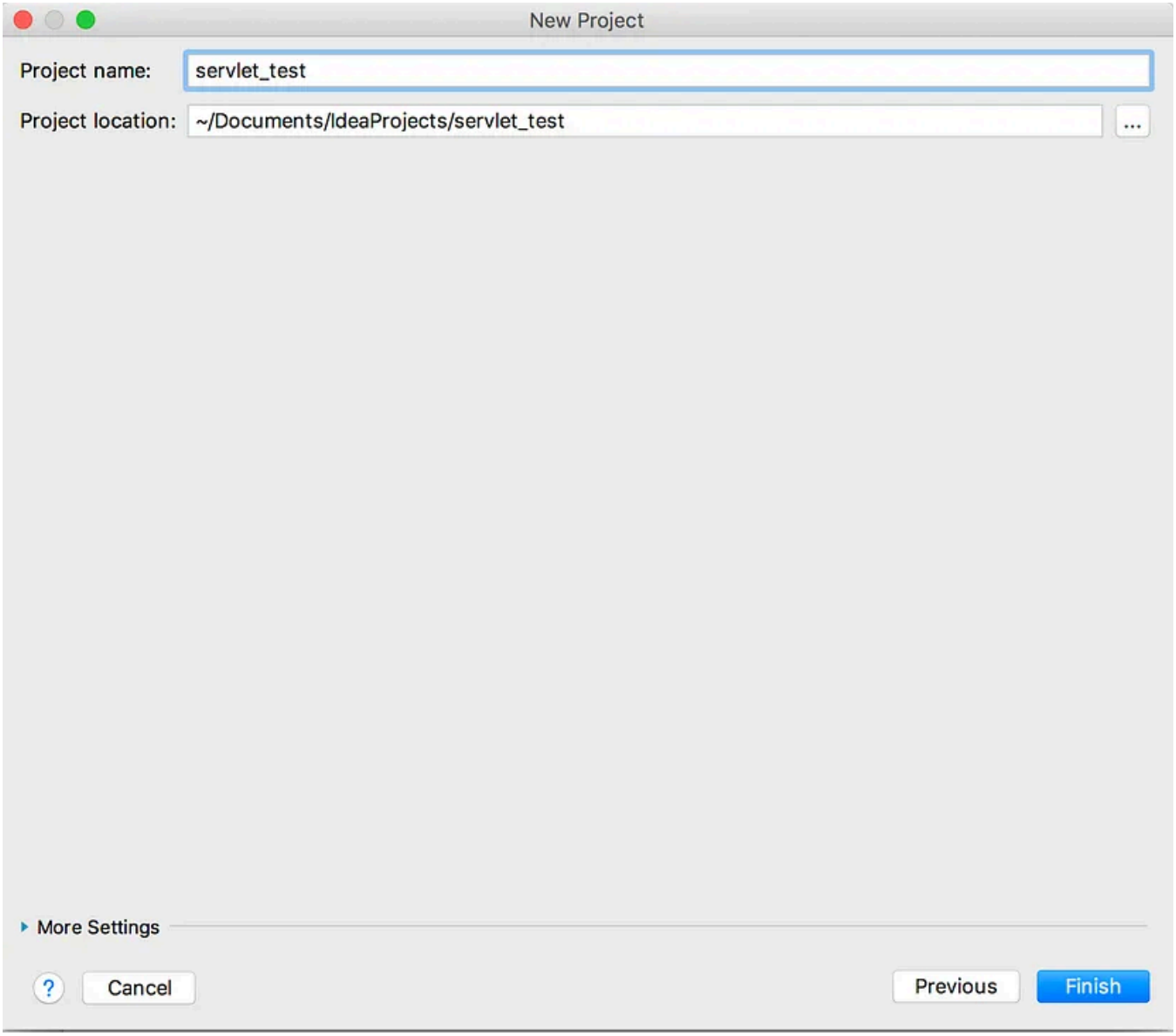
In the Tomcat Server dialog, specify the Tomcat Server installation directory.

5. Under Additional Libraries and Frameworks, select the Web Application checkbox.



Click Next.

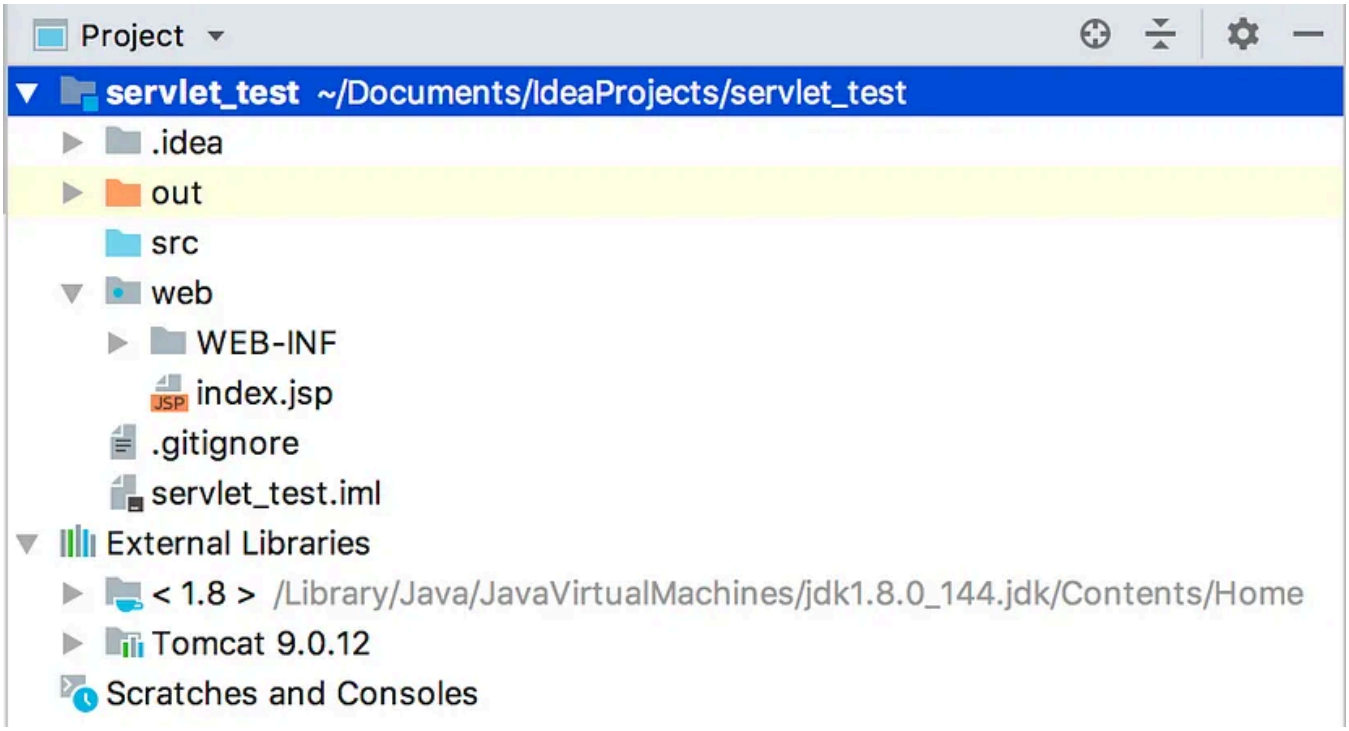
6. Specify the name for your new project



Click Finish and wait while IntelliJ IDEA is creating the project.

Exploring the project structure

When the project is created, you'll see something similar to this in the Project tool window.



- JavaEEHelloWorld is a module folder (which in this case coincides with the project folder). The `.idea` folder and the file `JavaEEHelloWorld.iml` contain configuration data for your project and module respectively. The folder `src` is for your Java source code. The folder `web` is for the web part of your application. At the moment this folder contains the deployment descriptor `WEB-INF/web.xml` and the file `index.jsp` intended as a starting page of your application.
- External Libraries include your JDK and the JAR files for working with Tomcat.

Developing source code

Our application will be a single JSP page application. Its only function will be to output the text *Hello, World!*

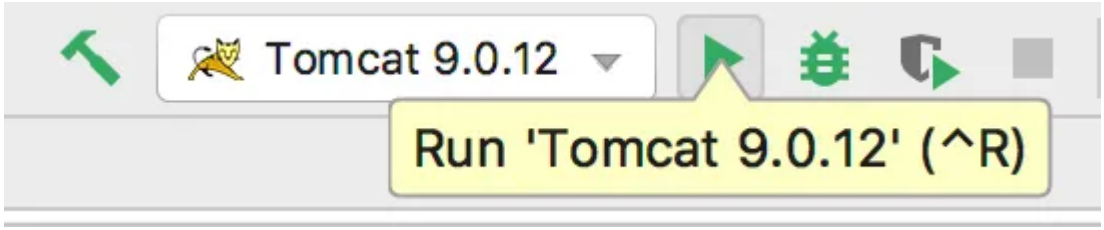
1. Open `index.jsp` for editing: select the file in the Project tool window and press ⌘↓.
2. Between `<body>` and `</body>` type `Hello, World!`

```
lex.jsp
<%--
  Created by IntelliJ IDEA.
  User: jeongwhanchoi
  Date: 15/09/2018
  Time: 7:55 PM
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title>$Title$</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

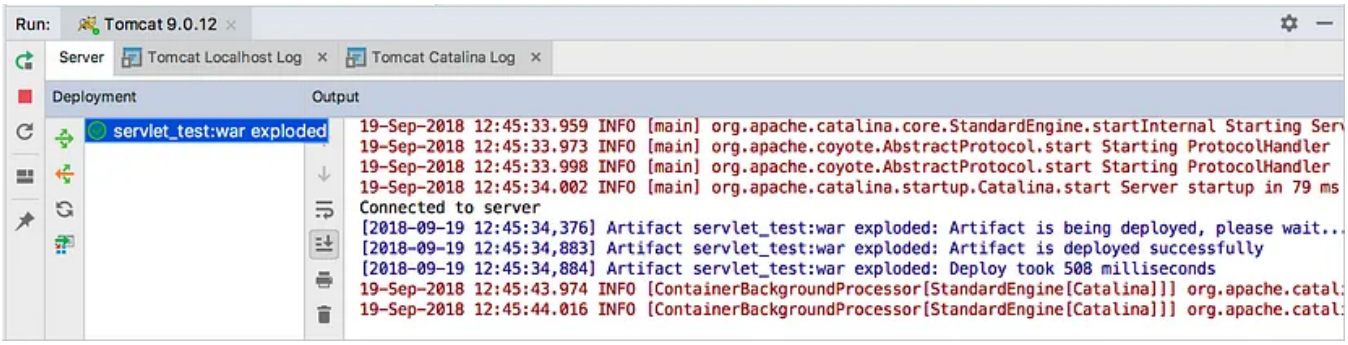
The code at this step is ready.

Running the application

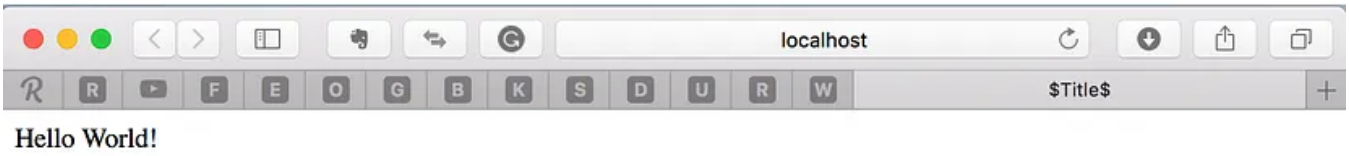
In the upper-right part of the workspace, click ▶.



After that, the Run tool window opens. IntelliJ IDEA starts the server and deploys the artifact onto it.

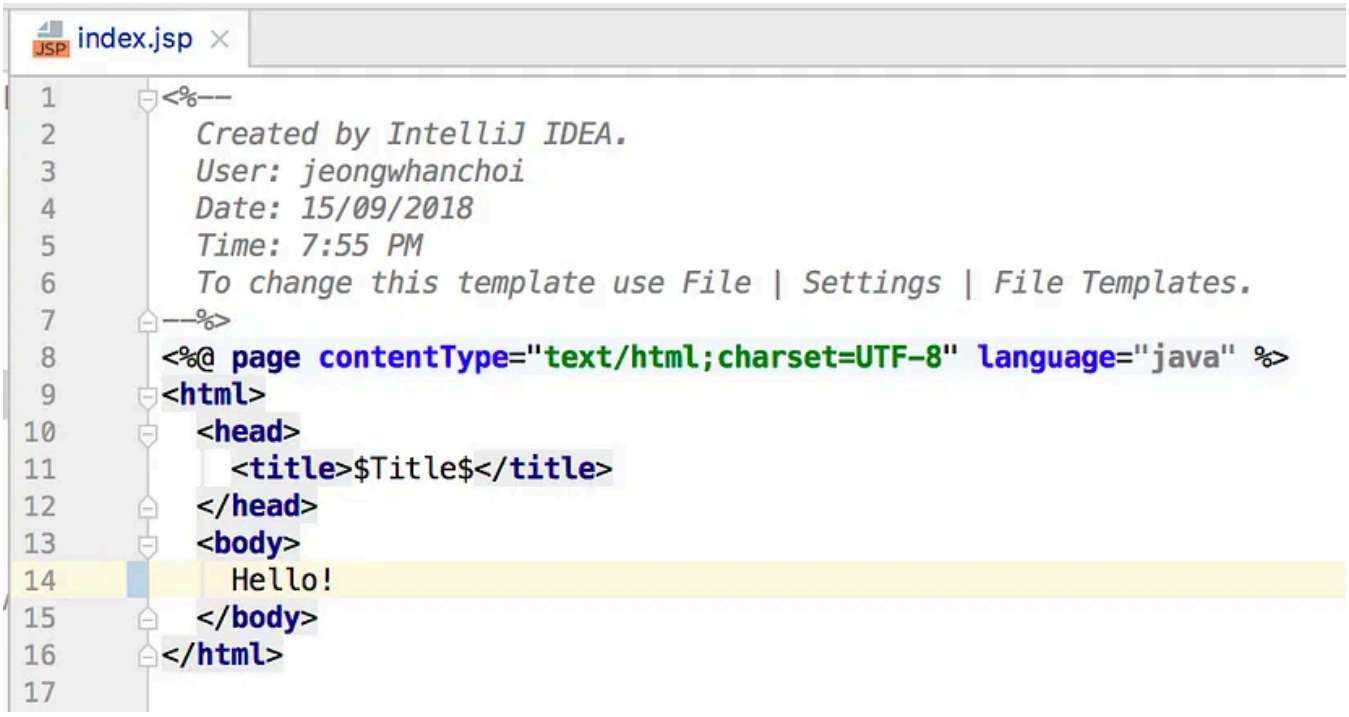


Finally, your default web browser starts and you see the application output `Hello, World!` there.

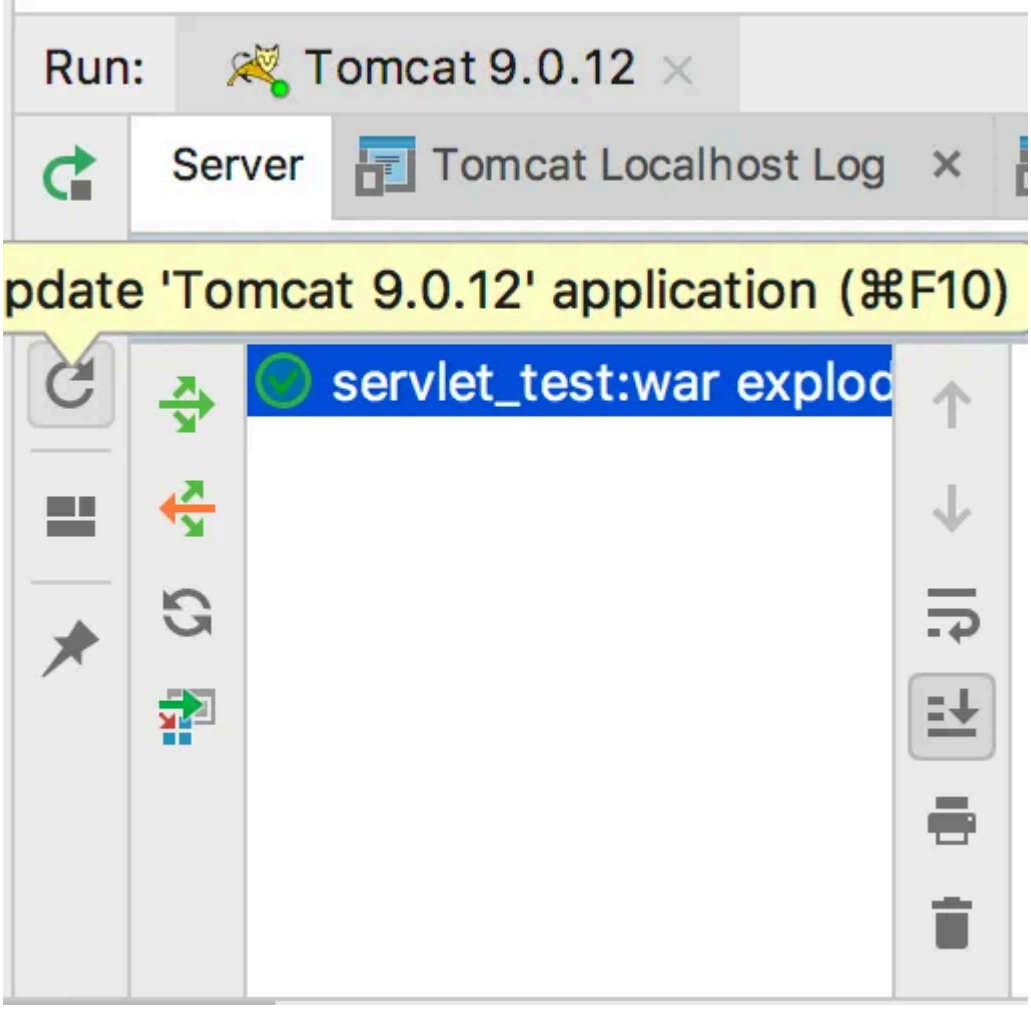


Modifying the code and observing the changes

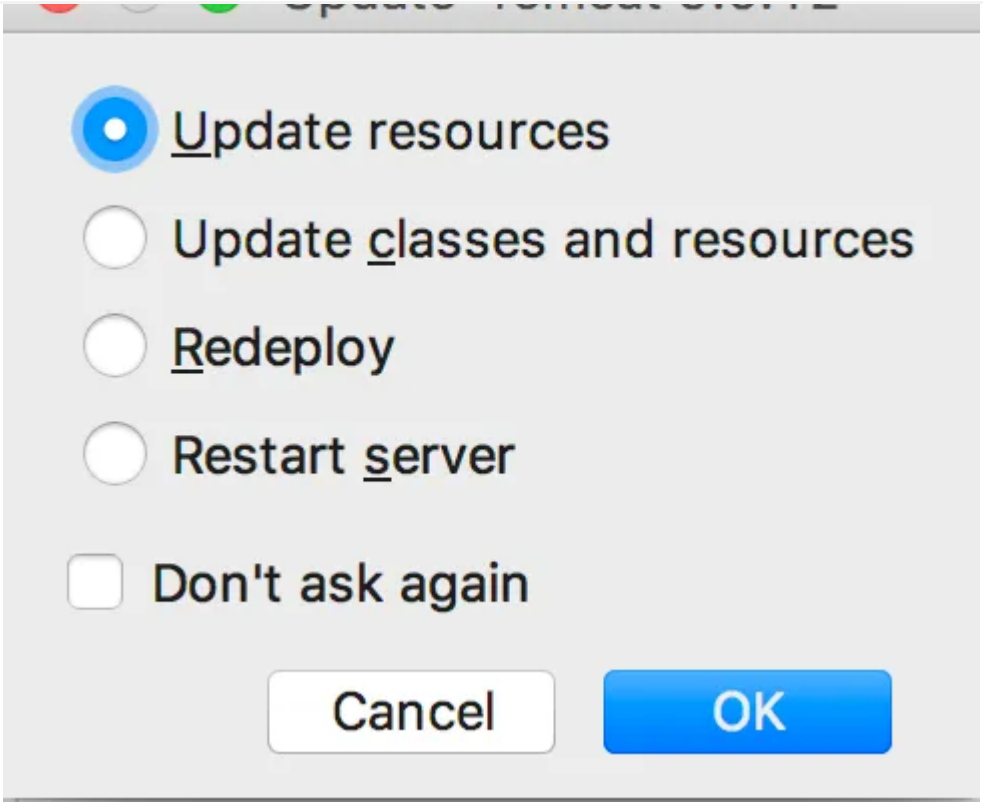
1. In `index.jsp`, change `Hello, World!` to `Hello!`.



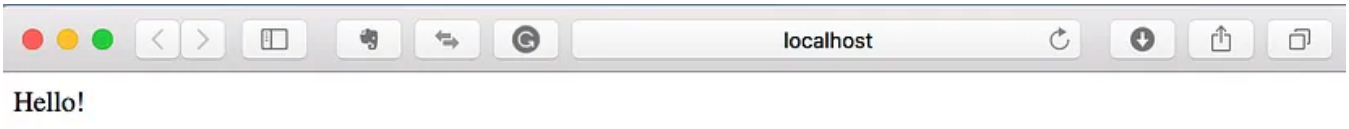
2. In the Run tool window, click Update button



3. In the Update dialog, select Update resources and click OK. (For more



4. Switch to the web browser and reload the page to see the changes.



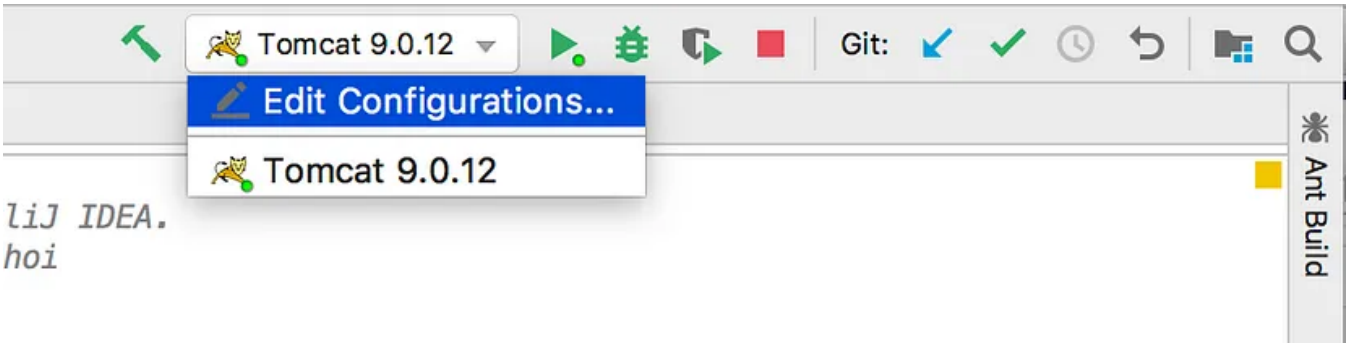
See also, Updating Applications on Application Servers.

Exploring a run configuration

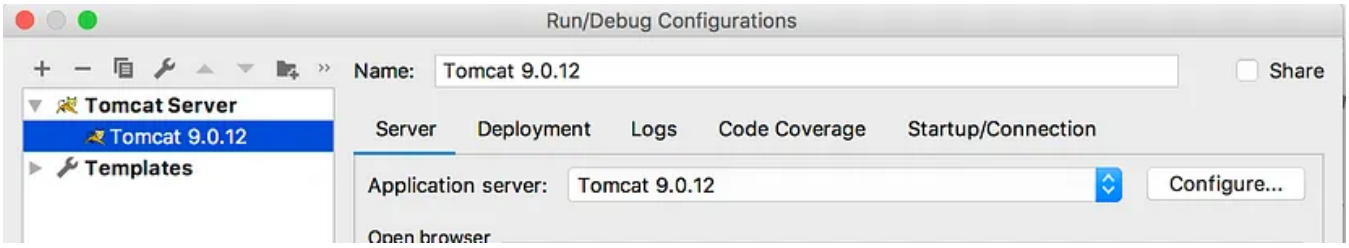
When creating the project, we specified Tomcat as an application server. As a result, IntelliJ IDEA created a run configuration for Tomcat.

When we performed the Run command (▶), we started that run configuration. Now let's take a look at the run configuration and see how its settings map onto the events that we've just observed.

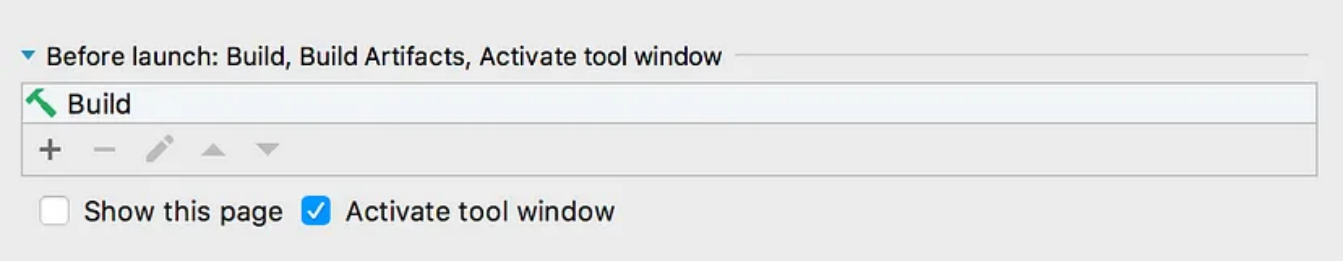
1. Click the run configuration selector and select **Edit Configurations**.



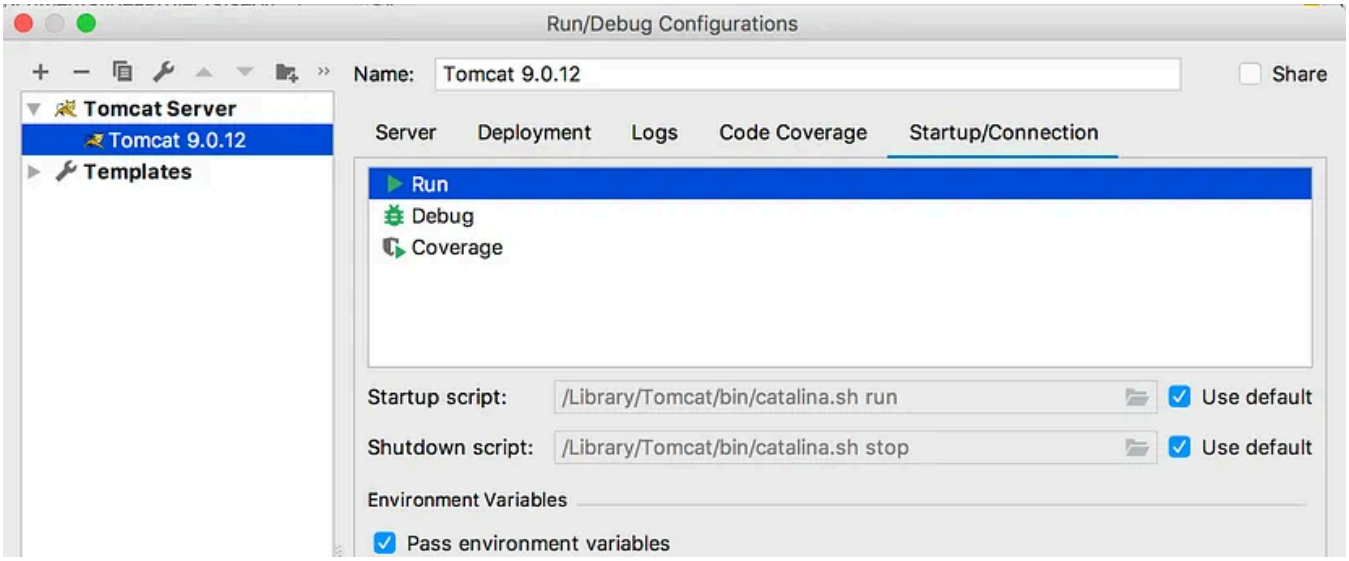
The Run/Debug Configurations dialog opens and the settings for the Tomcat run configuration are shown.



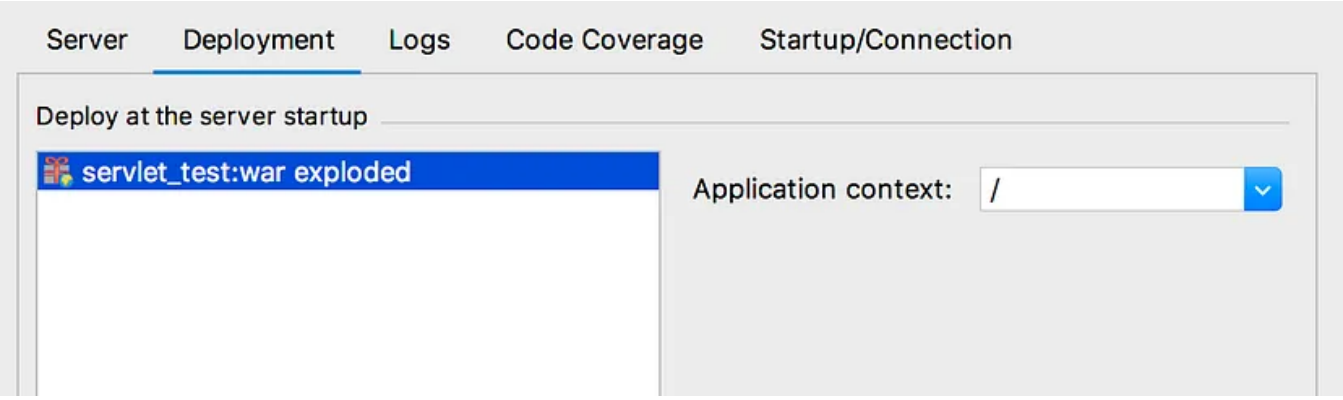
The Before launch task list (in the lower part of the dialog) specifies that the application code should be compiled and the corresponding artifact should be built prior to executing the run configuration.



2. Select the **Startup/Connection** tab to see how the server is started in the run, debug and code coverage modes.



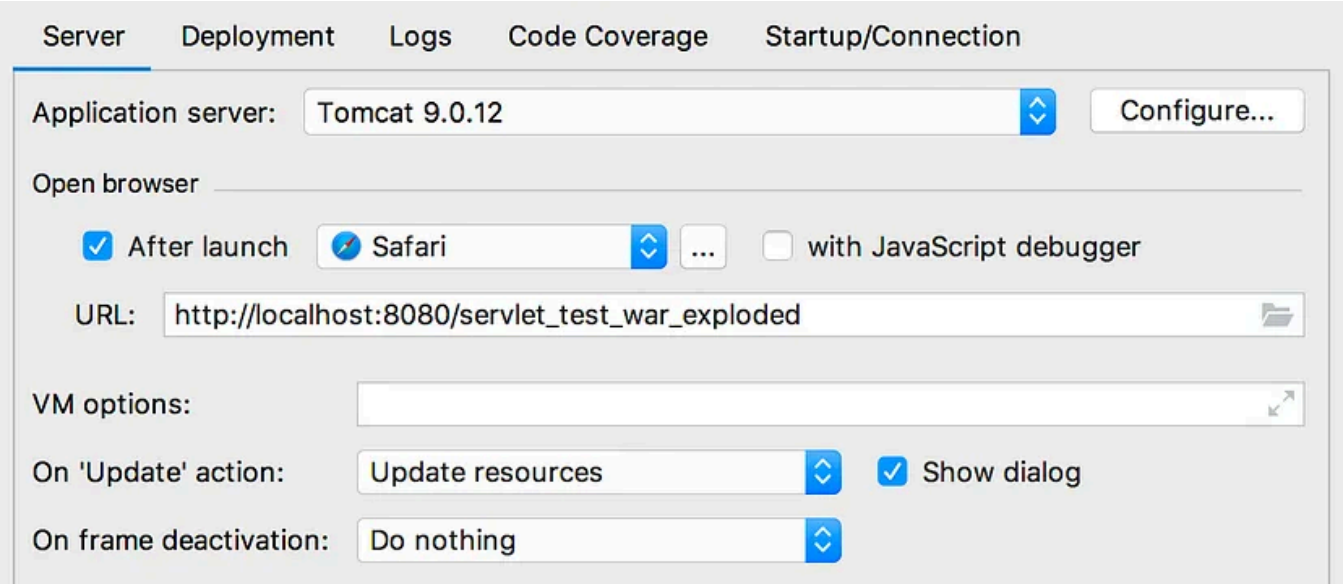
3. Select the **Deployment** tab to see which artifacts are deployed after the server is started.



4. Go back to the **Server** tab.

The settings under **Open browser** specify that after launch (i.e. after the server is started and the artifacts are deployed onto it) the default web browser should start and go to the specified URL (`http://localhost:8080/servlet_test_war_exploded`).

The settings to the right of On 'Update' action specify that on clicking the reload button in the Run tool window the Update dialog should be shown and the Update resources option should be used by default. (The last used update option becomes the default one).



5. Click OK.

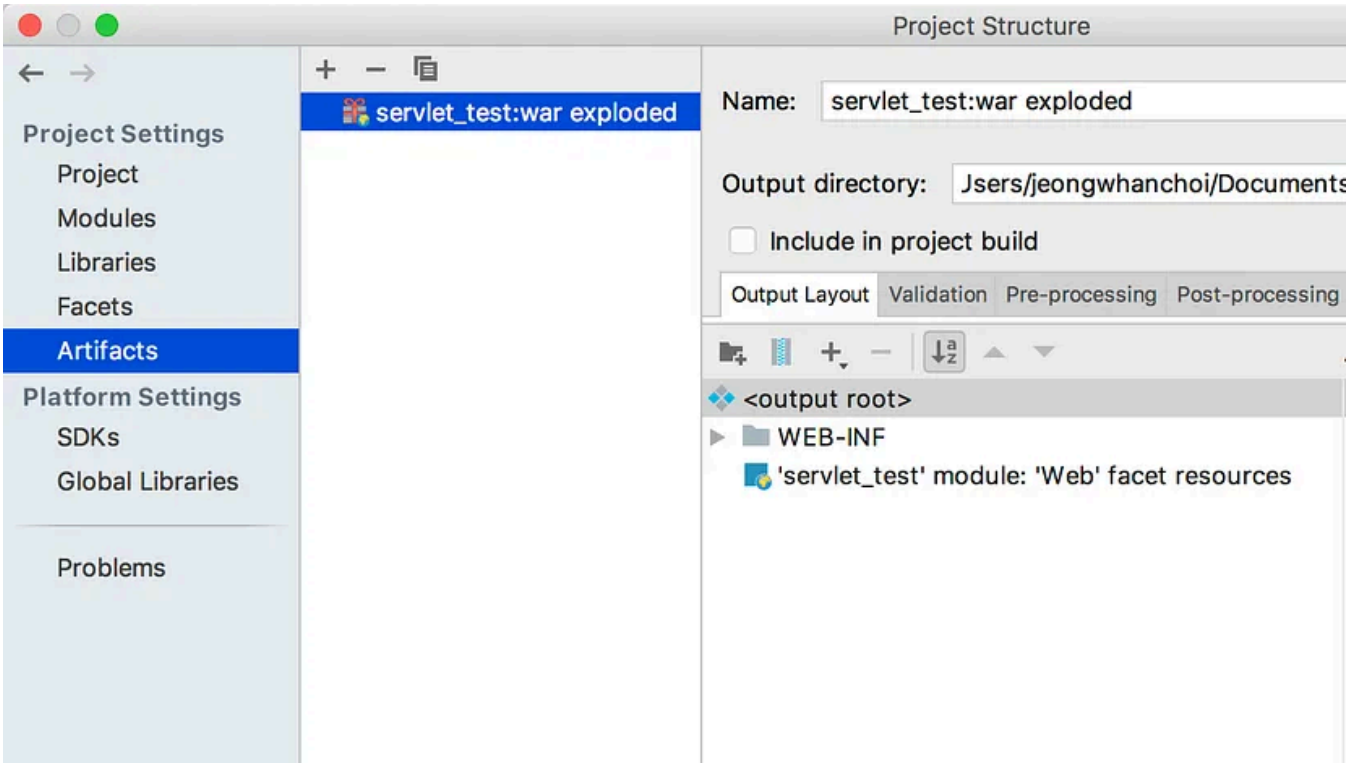
Exploring an artifact configuration

When creating the project, we indicated that we were going to develop a web application. As a result, IntelliJ IDEA, among other things, created a configuration for building a web application artifact. Let's have a look at this configuration.

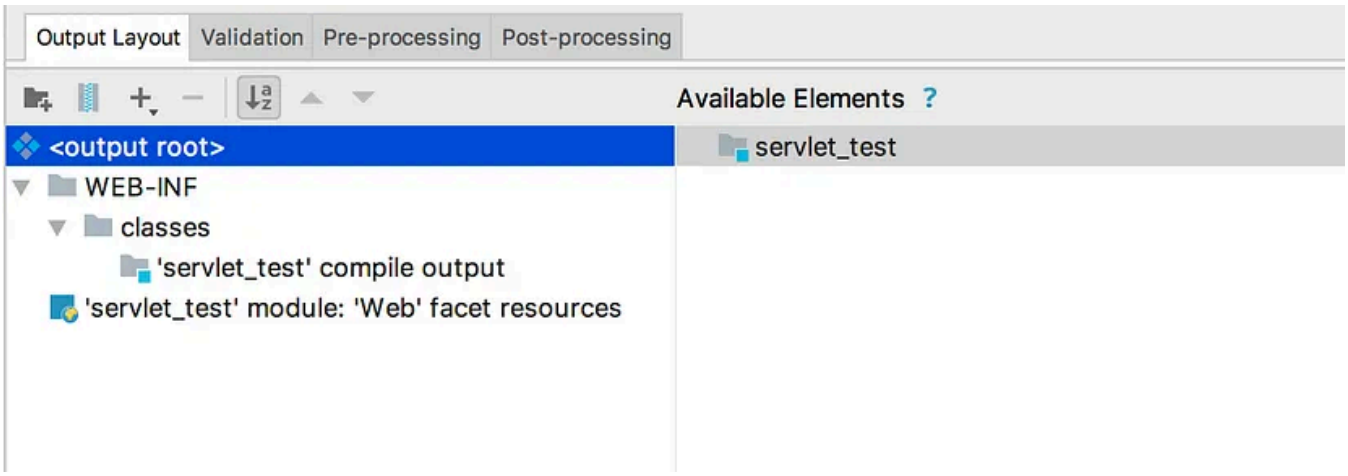
1. Open the Project Structure dialog: **File | Project Structure** or ⌘; .

2. Under **Project Settings**, select **Artifacts**.

The available artifact configurations are shown in the pane to the right under + and –. (There's only one configuration at the moment.)



The artifact settings are shown in the right-hand part of the dialog.



Type. The artifact type is Web Application: Exploded. This is a decompressed web application archive (WAR), a directory structure that is ready for deployment onto a web server.

Output directory. The artifact, when built, is placed into `<project_folder>/out/artifacts/JavaEEHelloWorld_war_exploded`.

Output Layout. The artifact structure is shown in the left-hand pane of the Output Layout tab.

The `<output root>` corresponds to the output directory. Other elements have the following meanings:

- **'JavaEEHelloWorld' compile output** represents compiled Java classes whose sources are located in the `src` directory. These are placed into `WEB-INF/classes` in the output directory.
- 'Web' facet resources represent the contents of the `web` directory.

Reference

https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-ee-application.html#explore_run_config

Java Tomcat Jsp

Follow

Written by Jeongwhan Choi

105 Followers

jeongwhanchoi.me

More from Jeongwhan Choi

9.0.12

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
  - 32-bit Windows zip (pgp, sha512)
  - 64-bit Windows zip (pgp, sha512)
  - 32-bit/64-bit Windows Service Installer (pgp, sha512)
- Full documentation:
  - tar.gz (pgp, sha512)
- Deployer:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
- Extras:
  - JMX Remote jar (pgp, sha512)
  - Web services jar (pgp, sha512)
- Embedded:
  - tar.gz (pgp, sha512)
  - zip (pgp, sha512)

Jeongwhan Choi

How to Install Apache Tomcat on Mac OS X

Install Tomcat

3 min read · Sep 15, 2018

138 2

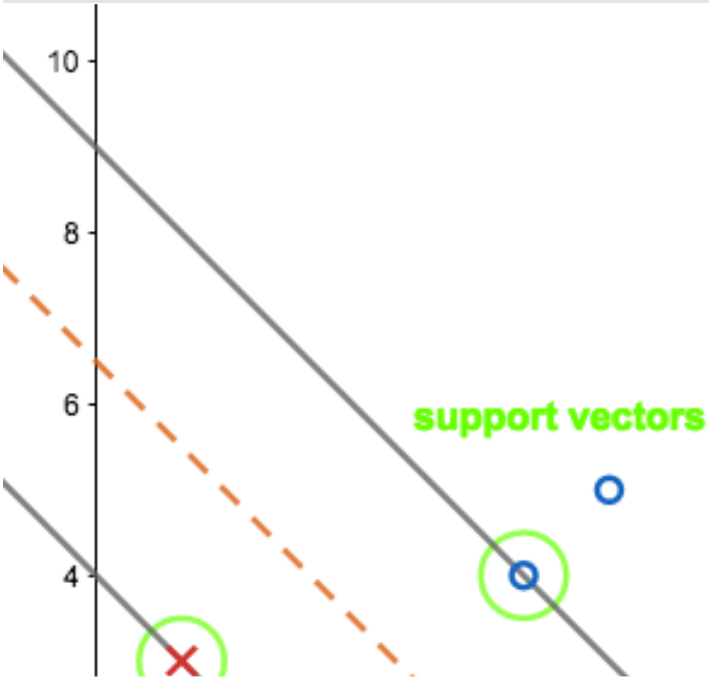
Jeongwhan Choi

Useful Tips for Writing a Paper on Overleaf

From helpful highlighting tips when collaborating, to working with tables, and how to avoid mistakes.

4 min read · Sep 12, 2021

118



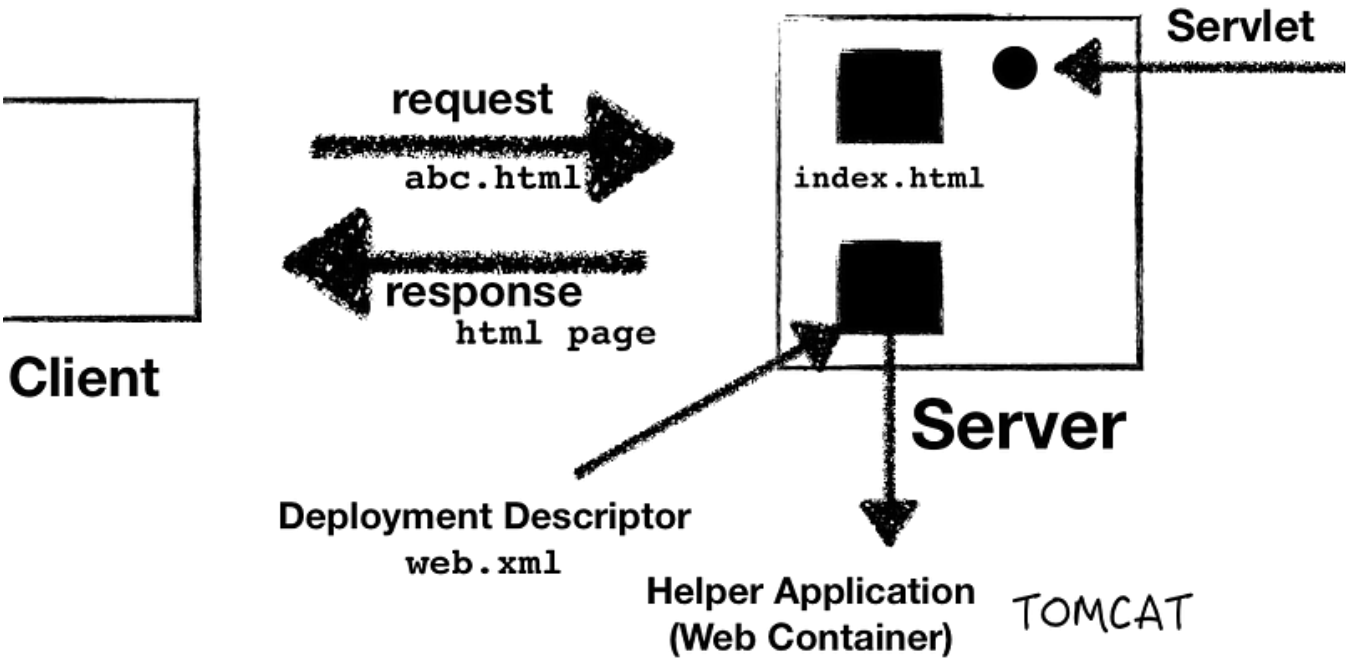
Jeongwhan Choi

Intro to Support Vector Machine

Decision Boundary

6 min read · Sep 14, 2018

20



Jeongwhan Choi

Introduction to Servlets

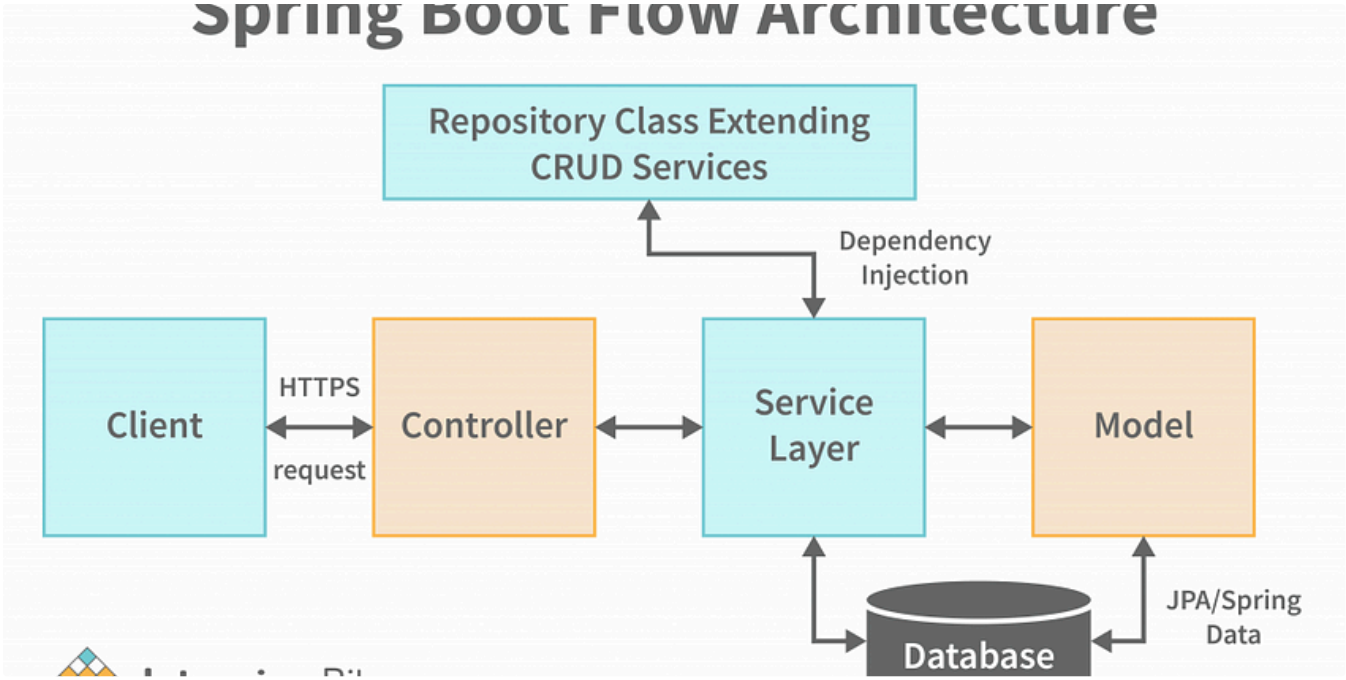
How Servlet Works

5 min read · Sep 15, 2018

112

See all from Jeongwhan Choi

Recommended from Medium



Gagan Jain

How to start with spring boot web application with maven

Starting a Spring Boot web application with Maven is a common and straightforward process. You can create a simple web application by...

2 min read · Nov 6, 2023

12

Ajay Rathod

Cisco Java Developer Interview Transcript 2024(Java,Spring-Boot,Hibernate)

“Hello folks, I am jotting down the full tech interview round for a Java developer position at Cisco. All these Q&A are actual questions...

12 min read · Apr 14, 2024

196 1

Lists

General Coding Knowledge

20 stories · 1139 saves

data science and AI

40 stories · 136 saves

Gozde Saygili Yalcin

Using Docker Compose with Spring Boot and PostgreSQL

In today's software world, Docker Compose makes it easy to handle Spring Boot apps in Docker containers, especially complex ones. You can...

5 min read · Dec 5, 2023

107

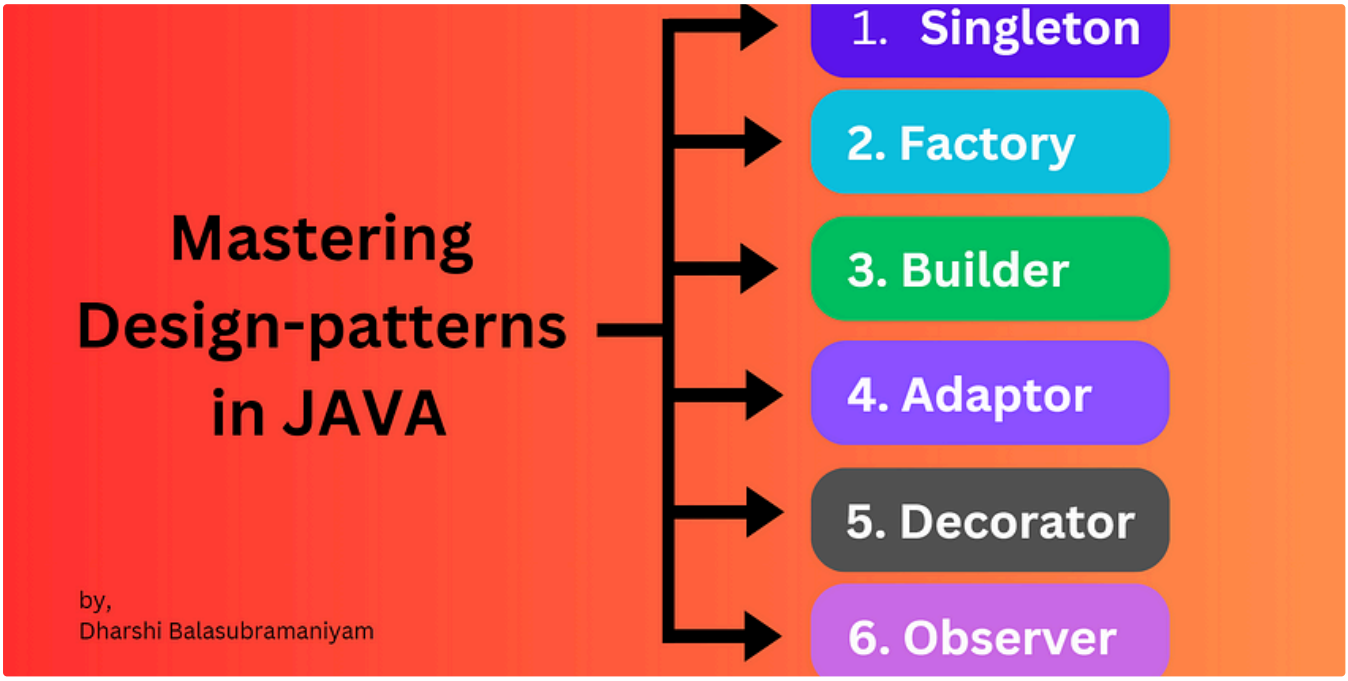
Page against the machine

Building a core banking microservice with Spring boot and MongoDB

This article describes the creation and performance testing of a core banking-style microservice with MongoDB. It includes both a basic...

9 min read · Apr 11, 2024

9



Dharshi Balasubramaniyam in Javarevisited

Mastering Design Patterns in Java

Figure 1: Design patterns

16 min read · Feb 26, 2024

242 6