

Spring Boot

Bootstrap your application development, batteries included!

Spring Boot

- Spring Boot is the easiest way to launch a new application and maintain an existing one.
- “Starter” dependencies include all required dependencies for standard applications
- Curated compatibility of all dependency versions
- Auto configures deployment environment
- Easily override defaults
- Easily snap in to EOS and/or Propstore

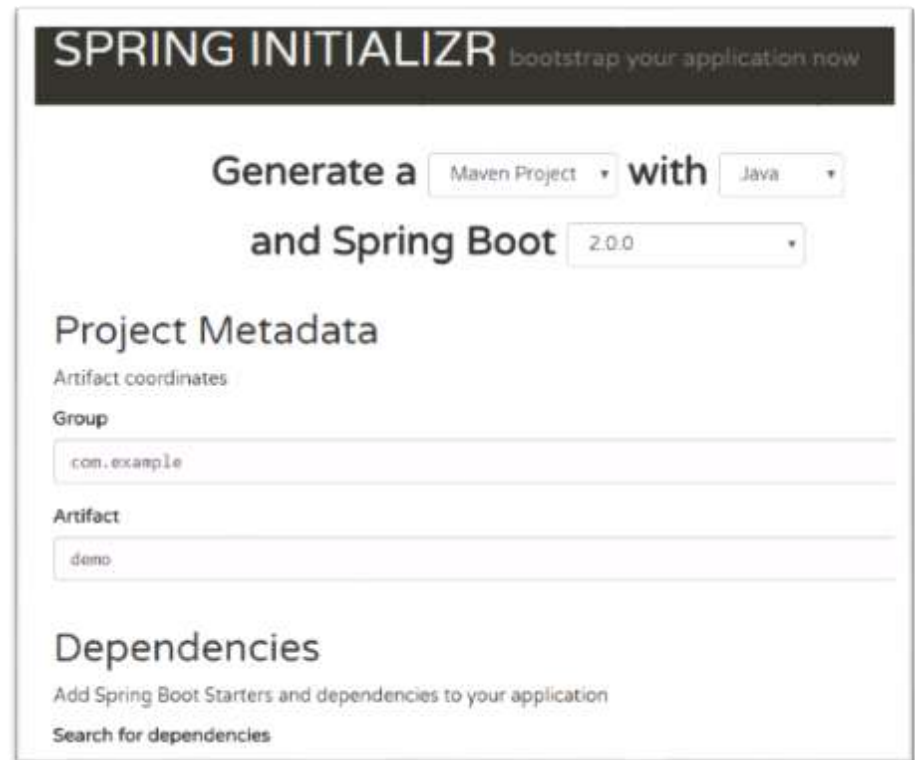
Spring Boot minimum requirements

- Java 8 or greater
- Maven 3 or Gradle 4
- Spring 4 or greater

Spring Boot “Initializr”

- Doesn't require an IDE:

<https://start.spring.io>



The screenshot shows the Spring Initializr web application interface. At the top, there is a dark header with the text "SPRING INITIALIZR" and a subtitle "bootstrap your application now". Below the header, the main form is titled "Generate a" followed by a dropdown menu set to "Maven Project", then "with" followed by another dropdown menu set to "Java". Below this, it says "and Spring Boot" followed by a dropdown menu set to "2.0.0". The form is divided into sections: "Project Metadata" which includes "Artifact coordinates" with sub-fields for "Group" (containing "com.example") and "Artifact" (containing "demo"). Below this is a section titled "Dependencies" with the instruction "Add Spring Boot Starters and dependencies to your application" and a link "Search for dependencies".

Initializr – Using IntelliJ

The screenshot shows the IntelliJ IDEA 'New Project' dialog. The 'Dependencies' section is active, displaying a list of categories on the left and a list of dependencies on the right. The 'Web' category is selected, and the 'Web' dependency is checked. The 'Selected Dependencies' section shows the 'Web' and 'Actuator' dependencies. The 'Next' button is highlighted.

Dependencies

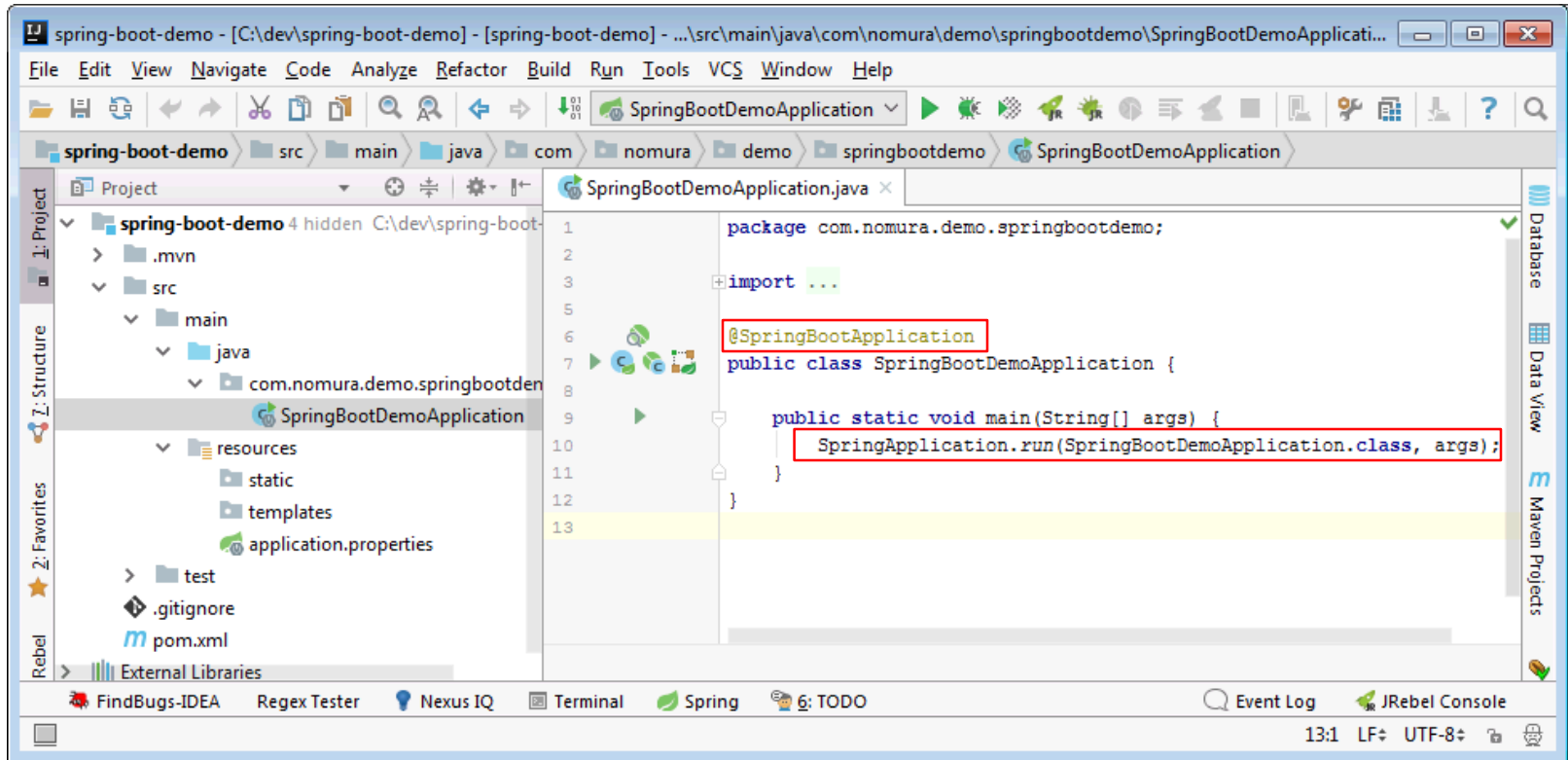
Category	Dependency	Selected
Core	Web	<input checked="" type="checkbox"/>
	Reactive Web	<input type="checkbox"/>
Template Engines	Rest Repositories	<input type="checkbox"/>
	Rest Repositories HAL Browser	<input type="checkbox"/>
SQL	HATEOAS	<input type="checkbox"/>
NoSQL	Web Services	<input type="checkbox"/>
Messaging	Jersey (JAX-RS)	<input type="checkbox"/>
Cloud Core	Websocket	<input type="checkbox"/>
Cloud Config	REST Docs	<input type="checkbox"/>
Cloud Discovery	Vaadin	<input type="checkbox"/>
Cloud Routing	Apache CXF (JAX-RS)	<input type="checkbox"/>
Cloud Circuit Breaker	Ratpack	<input type="checkbox"/>
Cloud Tracing	Mobile	<input type="checkbox"/>
Cloud Messaging	Keycloak	<input type="checkbox"/>
Cloud AWS		
Cloud Contract		
Pivotal Cloud Foundry		
Azure		
Social		
I/O		
Ops		

Selected Dependencies

Category	Dependency	Remove
Web	Web	<input checked="" type="checkbox"/>
Ops	Actuator	<input checked="" type="checkbox"/>

Buttons: Previous, Next, Cancel, Help

Spring Boot Initializr Generated Project



@SpringBootApplication:

- @Configuration – Designates this as a config class
- @EnableAutoConfiguration – Configures using sensible defaults
- @ComponentScan – Scans classpath for Spring classes

```
package com.nomura.demo.springbootdemo;

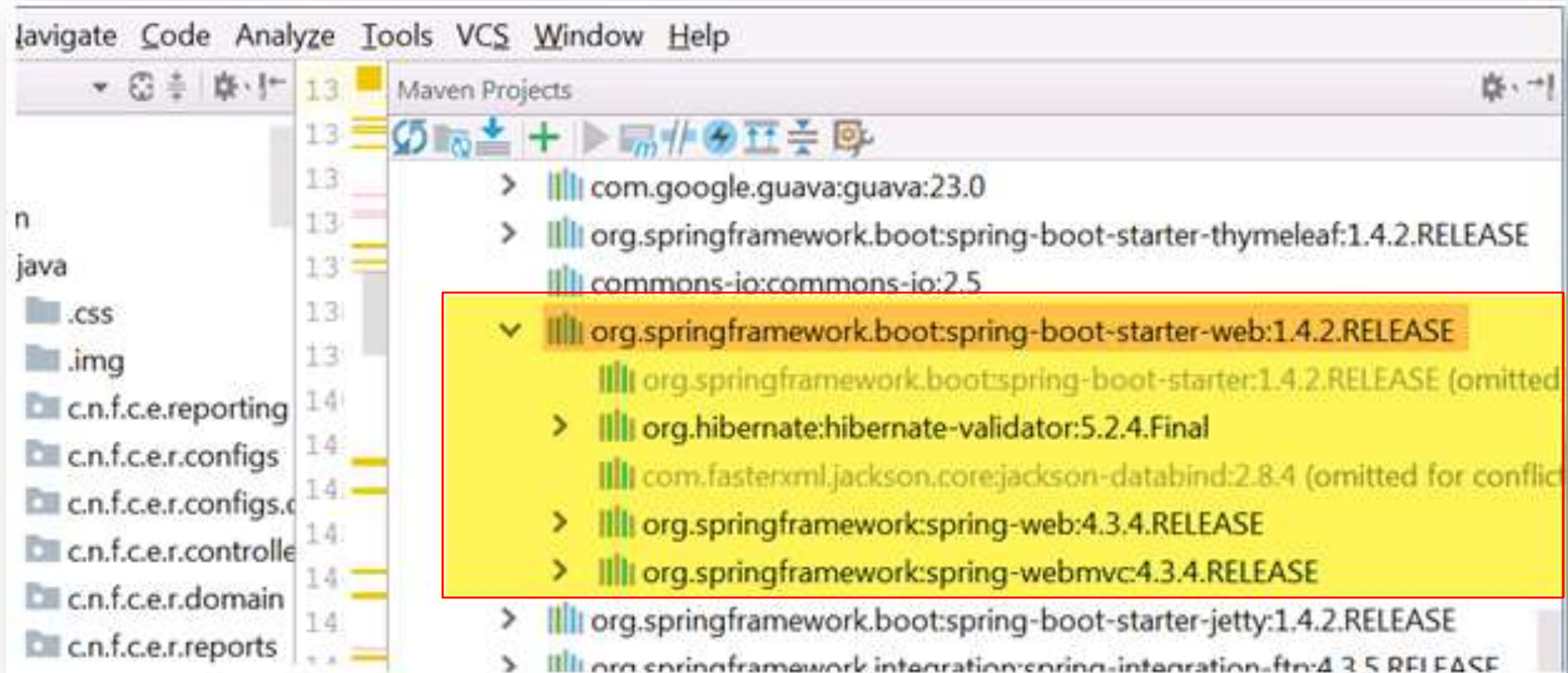
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;

@RestController
public class MyRestController
{
    @GetMapping("/hello/{name}")
    public String helloWorld(@PathVariable("name") String name) throws IOException
    {
        return "Hello, " + name ;
    }

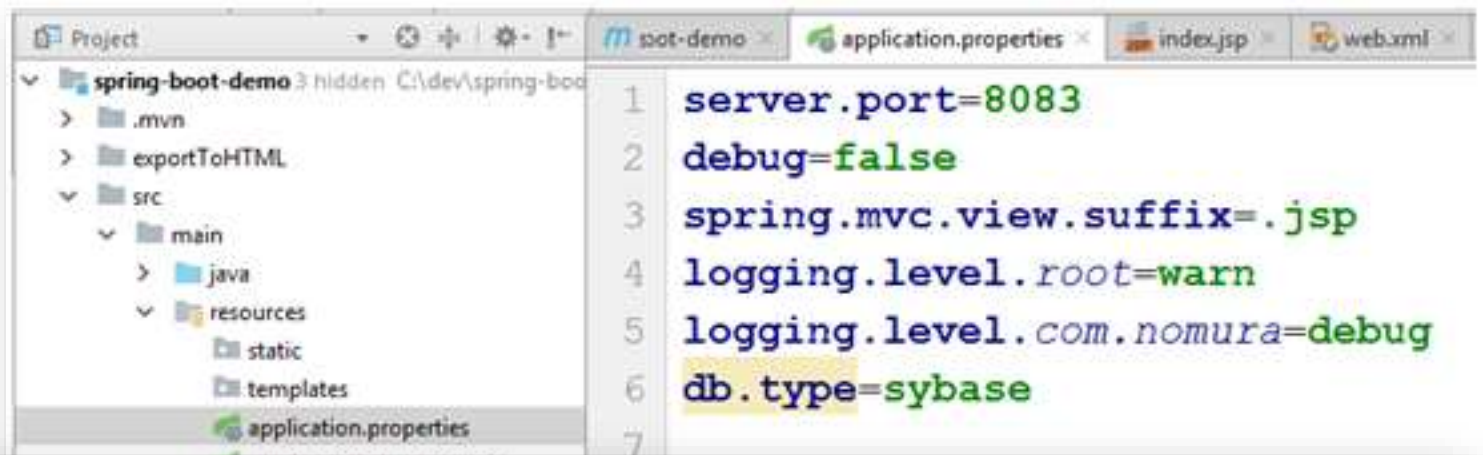
    @ExceptionHandler
    public ResponseEntity handle(IOException e)
    {
        return new ResponseEntity(e.getStackTrace(), HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

- Starter: Curated bills of materials of dependencies and versions, guaranteed to be compatible.
- All versions are inherited from the parent POM, just specify artifact.
- List of starters <https://github.com/spring-projects/spring-boot/tree/master/spring-boot-project/spring-boot-starters>



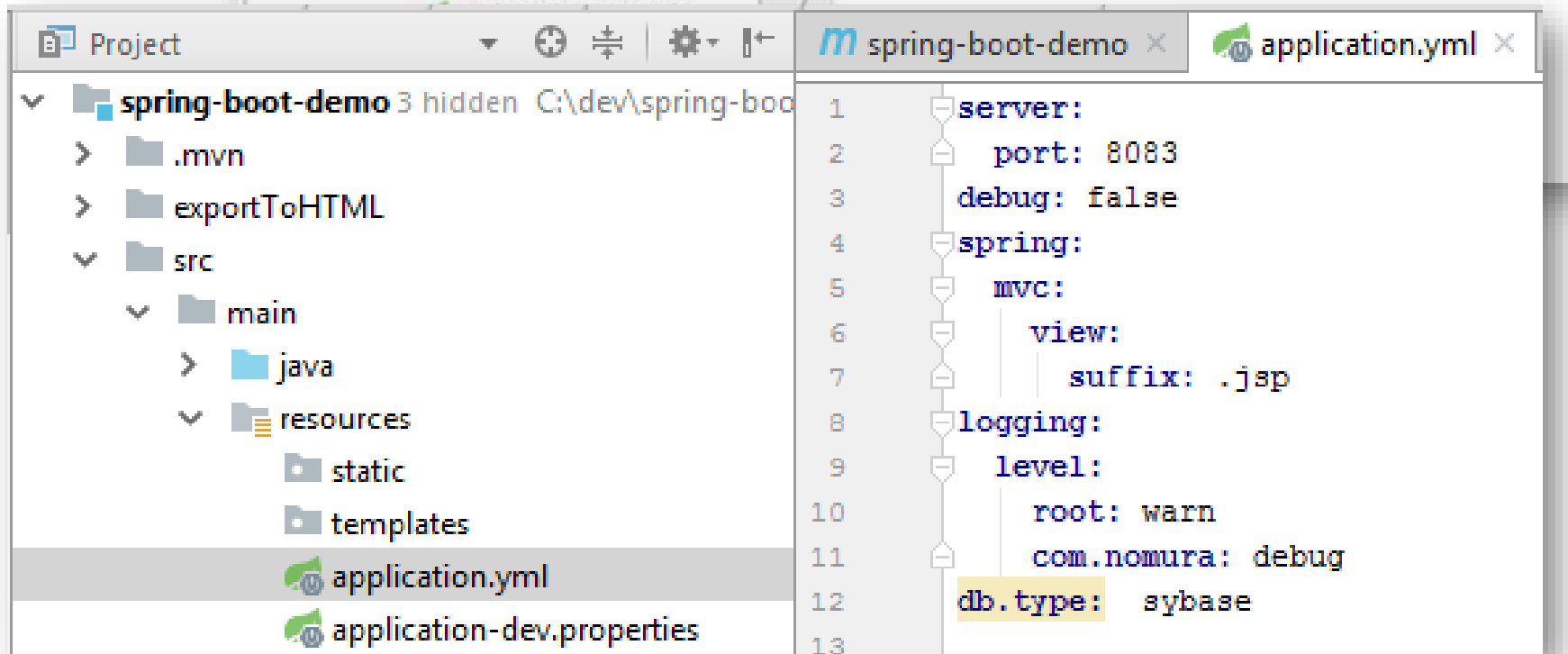
application.properties/ application.yml

NOMURA



This screenshot shows an IDE window with the file `application.properties` open. The left sidebar displays the project structure for `spring-boot-demo`, with the `resources` folder selected. The main editor area contains the following properties:

```
1 server.port=8083
2 debug=false
3 spring.mvc.view.suffix=.jsp
4 logging.level.root=warn
5 logging.level.com.nomura=debug
6 db.type=sybase
```



This screenshot shows the same IDE with the file `application.yml` open. The left sidebar shows the project structure, with `application.yml` selected under the `resources` folder. The main editor area displays the YAML configuration:

```
1 server:
2   port: 8083
3   debug: false
4 spring:
5   mvc:
6     view:
7       suffix: .jsp
8 logging:
9   level:
10    root: warn
11    com.nomura: debug
12 db.type: sybase
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Profile;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Component;

@Component
@Profile("app1")
@Order(1)
public class SomeClass1 implements CommandLineRunner
{
    private static final Logger logger = LoggerFactory.getLogger(SomeClass1.class);

    public static void main(String[] args)
    {
        new SomeClass1().launch();
    }

    private void launch()
    {
        logger.info("This is some class!");
    }

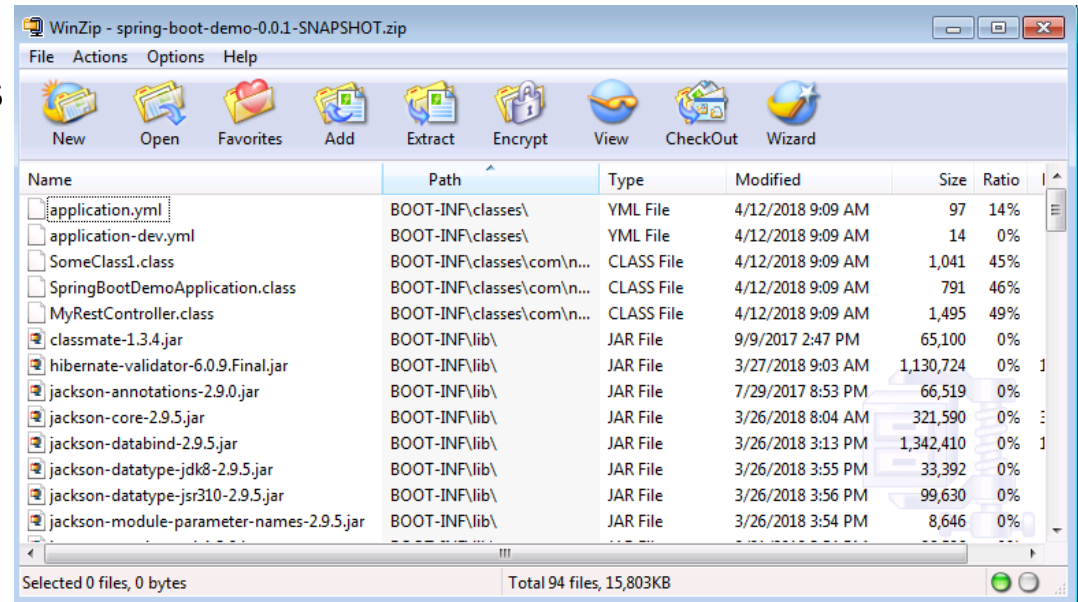
    @Override
    public void run(String... args) throws Exception
    {
        logger.info("Yay, you hit SomeClass1");
    }
}
```

Build and run

- Build the project as a jar or war, as with any Maven or Gradle build

```
mvn install
```

- Jar contains all dependencies



- To execute:

- `java -jar spring-boot-demo-0.0.1-SNAPSHOT.jar`

- `mvn spring-boot:run`

- `java -cpxxx`

`com.nomura.demo.springbootdemo.SpringBootDemoApplication`

EOS/LPS Integration

```
<plugin>
  <groupId>com.nomura.fid.core</groupId>
  <artifactId>maven-lps-plugin</artifactId>
  <version>1.16.0</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>package</goal>
      </goals>
      <configuration>
        <mainClass>org.springframework.boot.loader.JarLauncher</mainClass>
        <applicationArgs>
          --env=@LPS_ENV@ --hostname.full=@LPS_CANONICAL_HOSTNAME@
          --install.dir=@INSTALL_DIR@
          --region=@LPS_REGION@ --appInstance=all
        </applicationArgs>
        <packageType>zip</packageType>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Specify environment specific details
using LPS env placeholders

Propstore Integration

- Add a maven dependency:

```
<dependency>
```

```
<groupId>com.nomura.fid.core</groupId>
```

```
<artifactId>propstore-spring-boot-client</artifactId>
```

```
<version>2.0.4</version>
```

```
</dependency>
```

- Add a query property in application.properties:

env property

```
[propstore.property.sources=\\
```

property source

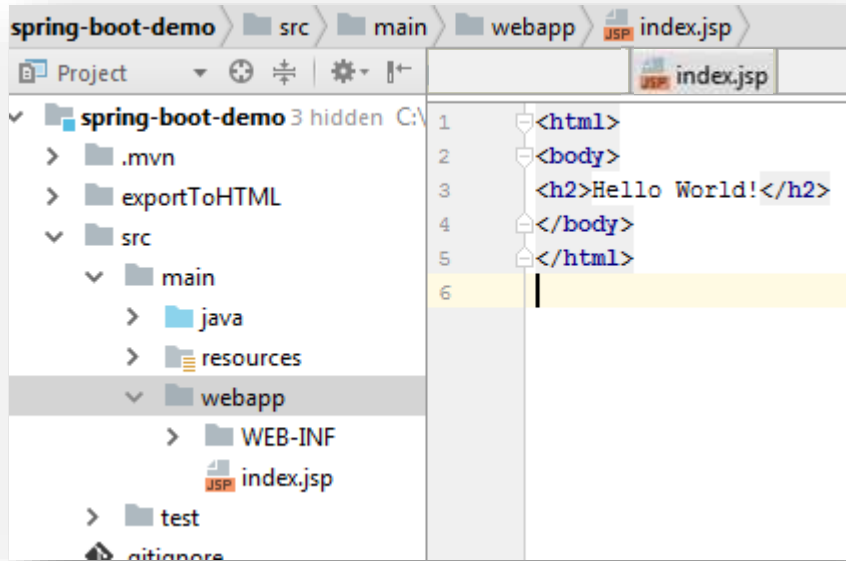
```
[propertySourceName=MyProperties,env=dev] || \\
```

Propstore query

```
[system=demo,app=my-app,region=eu]
```

Converting a Web App

- Move the webapp dir into main directory (No longer need WEB-INF)

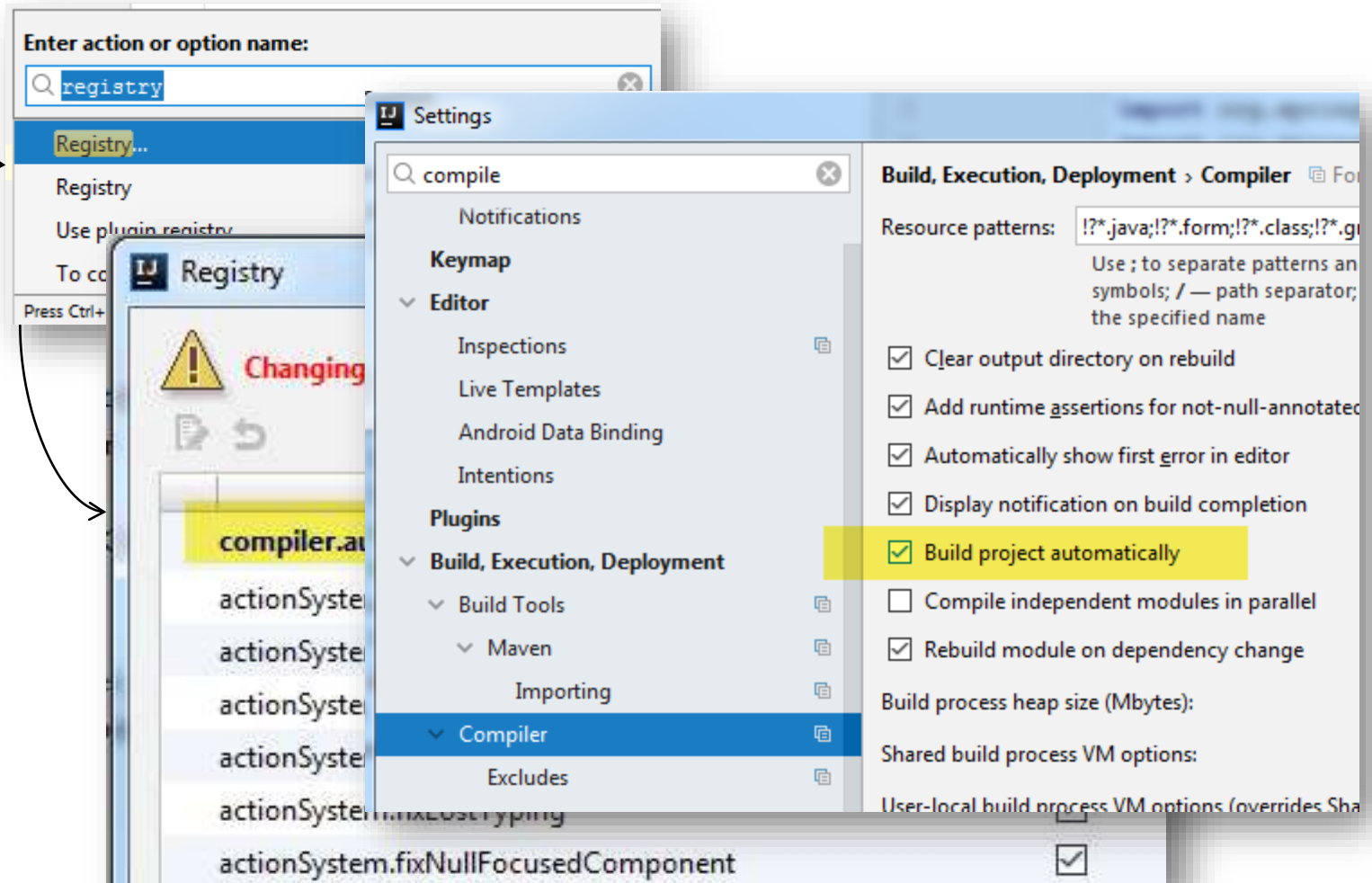


- Add @Controller annotation to the controller class
- Map a suffix property: `spring.mvc.view.suffix=.jsp`
- Create your RequestMappings:

```
@RequestMapping("/")
public String landing()
{
    return "index";
}
```

IntelliJ Settings for Spring Boot Dev Tools

Registry – Ctrl-Shift-A (then enter “registry”)



logging

CommandLineRunner

Execution:

- `java -jar SpringBootDemo.jar`
- `java SpringBootDemoApplication`

Ordering @Order(1)

Profiles

Nested profiles

Testing

Propstore

EOS integration