NOMURA
Connecting Markets East & West

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

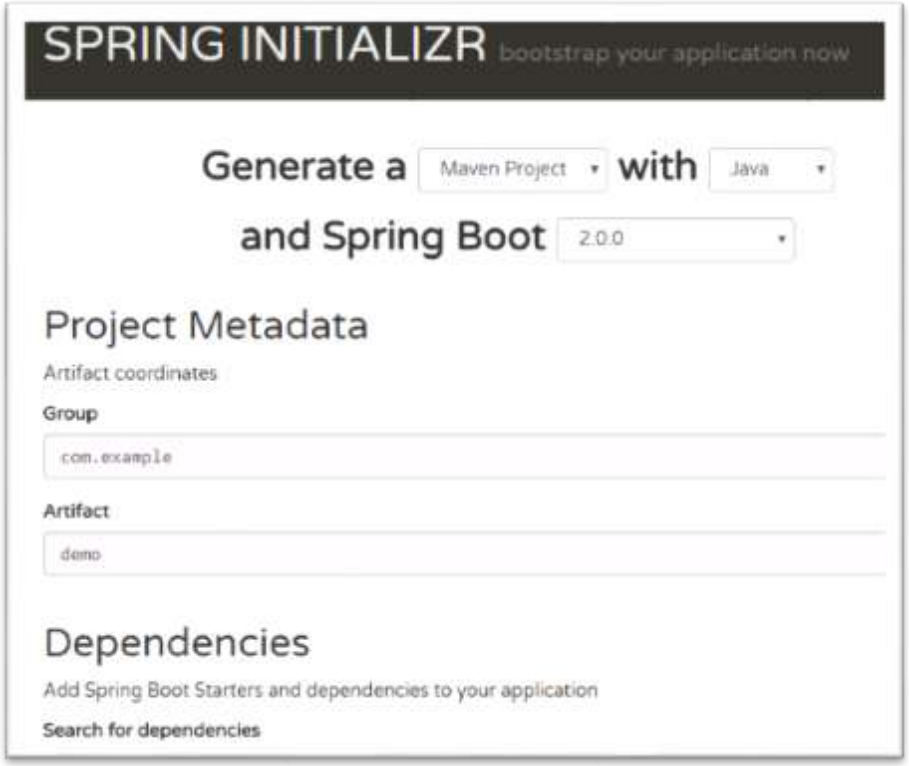
Spring Boot minimum requirements

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

Spring Boot “Initializr”

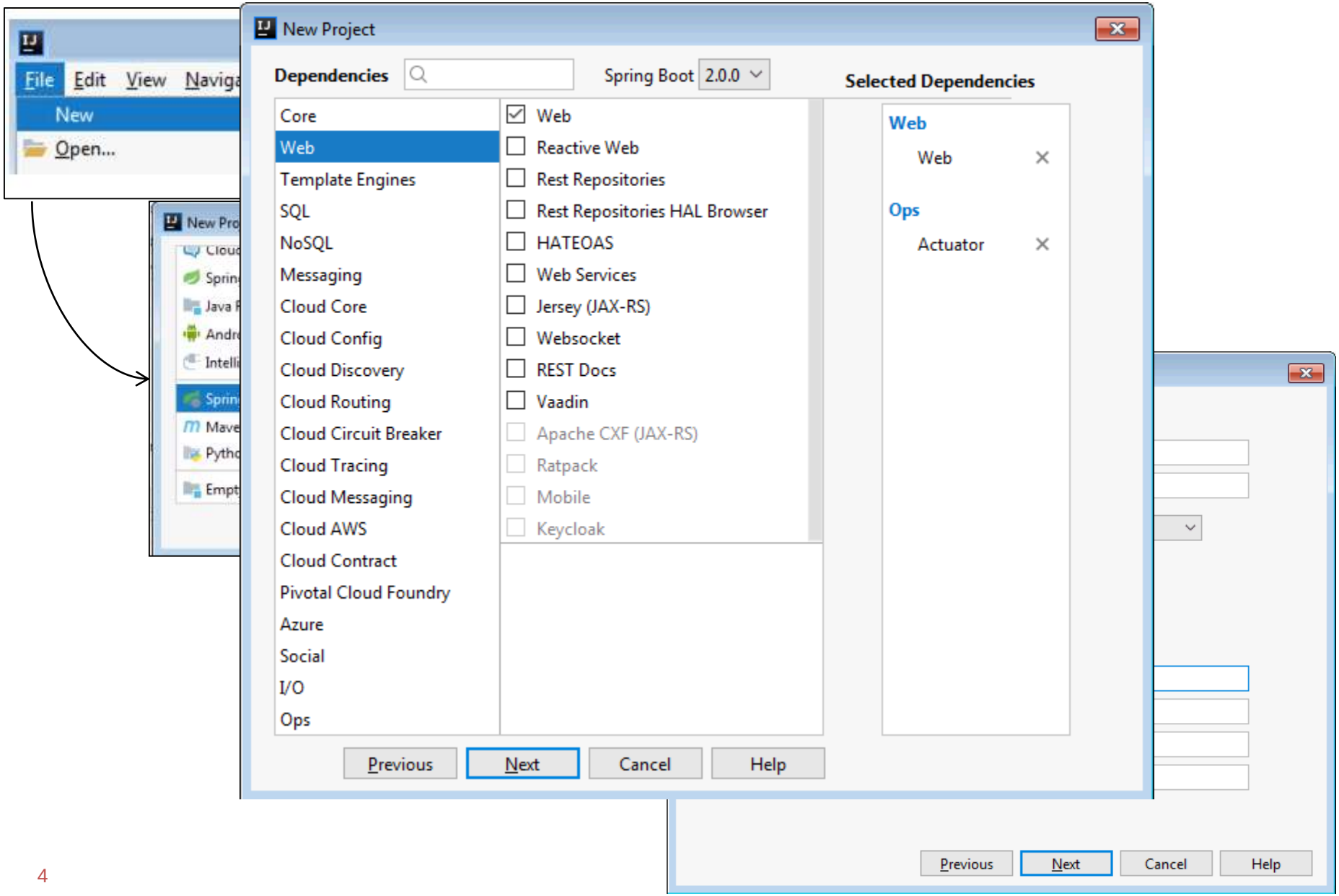
- Doesn’t require an IDE:

  https://start.spring.io



Initializr – Using IntelliJ

NOMURA



Spring Boot Initializr Generated Project

NOMURA

spring-boot-demo - [C:\dev\spring-boot-demo] - [spring-boot-demo] - ...\src\main\java\com\nomura\demo\springbootdemo\SpringBootDemoApplicati...

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

SpringBootDemoApplication

spring-boot-demo > src > main > java > com > nomura > demo > springbootdemo > SpringBootDemoApplication

Project

- spring-boot-demo 4 hidden C:\dev\spring-boot-
  - .mvn
  - src
    - main
      - java
        - com.nomura.demo.springbootden
          - SpringBootDemoApplication
      - resources
        - static
        - templates
        - application.properties
    - test
  - .gitignore
  - pom.xml
- External Libraries

SpringBootDemoApplication.java

```
1  package com.nomura.demo.springbootdemo;
2
3  import ...
5
6  @SpringBootApplication
7  public class SpringBootDemoApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(SpringBootDemoApplication.class, args);
11     }
12 }
13
```

FindBugs-IDEA Regex Tester Nexus IQ Terminal Spring 6: TODO Event Log JRebel Console

13:1 LF UTF-8

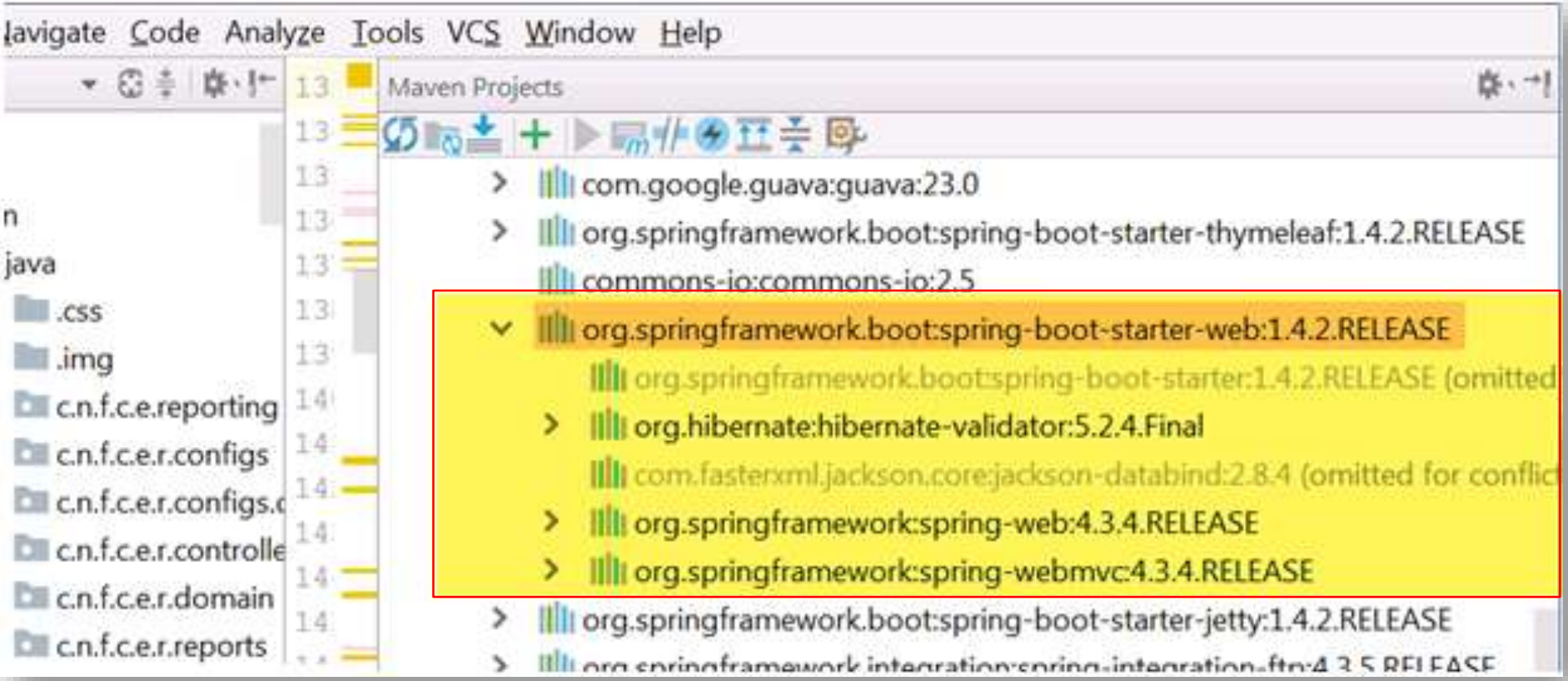
@SpringBootApplication:

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

```java
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);
    }
}
```

Starters

- 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

Project

spring-boot-demo 3 hidden C:\dev\spring-boo
- .mvn
- exportToHTML
- src
  - main
    - java
    - resources
      - static
      - templates
      - application.properties

spot-demo | application.properties | index.jsp | web.xml

```
server.port=8083
debug=false
spring.mvc.view.suffix=.jsp
logging.level.root=warn
logging.level.com.nomura=debug
db.type=sybase
```

Project

spring-boot-demo 3 hidden C:\dev\spring-boo
- .mvn
- exportToHTML
- src
  - main
    - java
    - resources
      - static
      - templates
      - application.yml
      - application-dev.properties

spring-boot-demo | application.yml

```
server:
  port: 8083
debug: false
spring:
  mvc:
    view:
      suffix: .jsp
logging:
  level:
    root: warn
    com.nomura: debug
db.type:  sybase
```

NOMURA

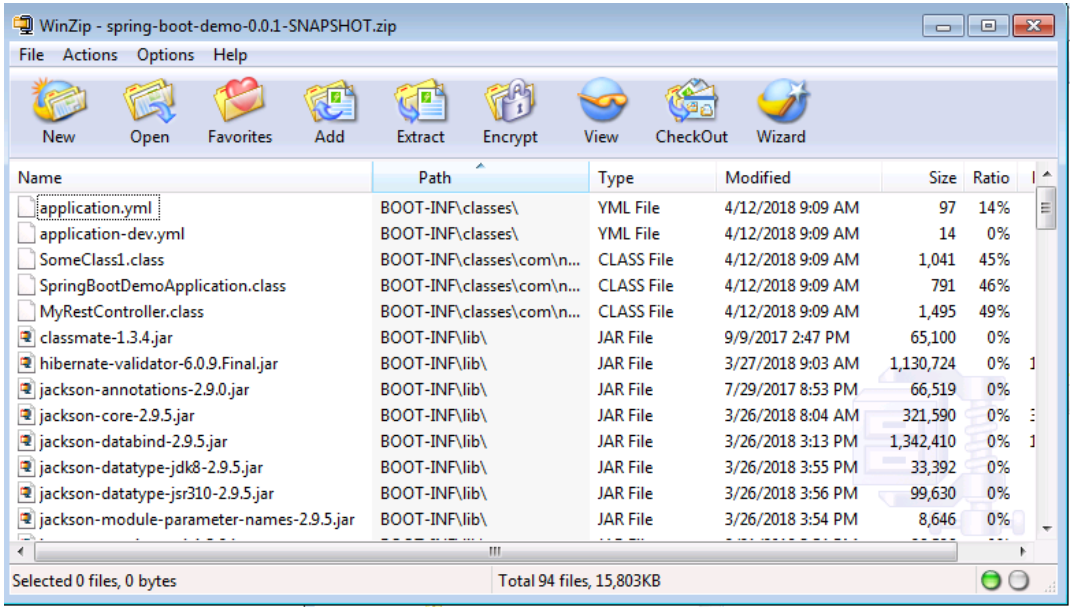
```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Profile;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Component;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

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

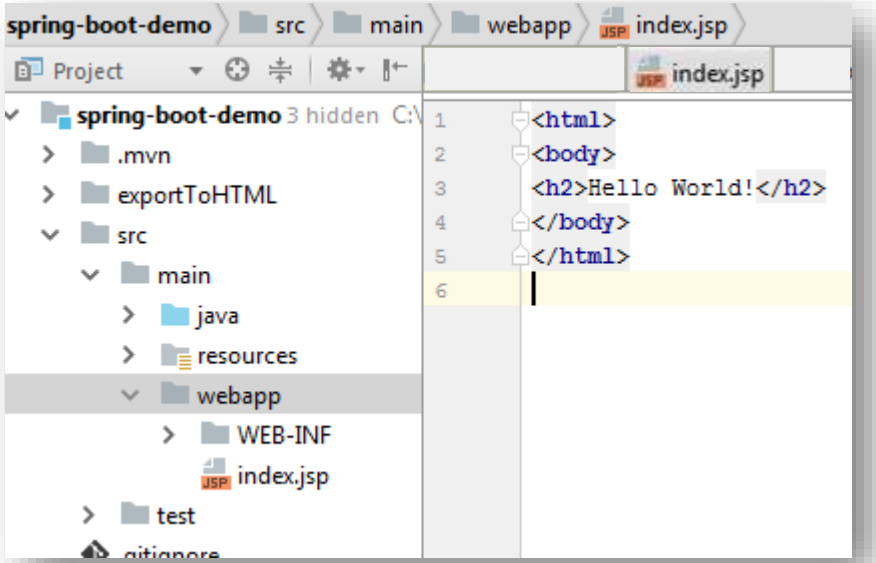
```
propertySourceName=MyProperties,env=dev || \
```

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)

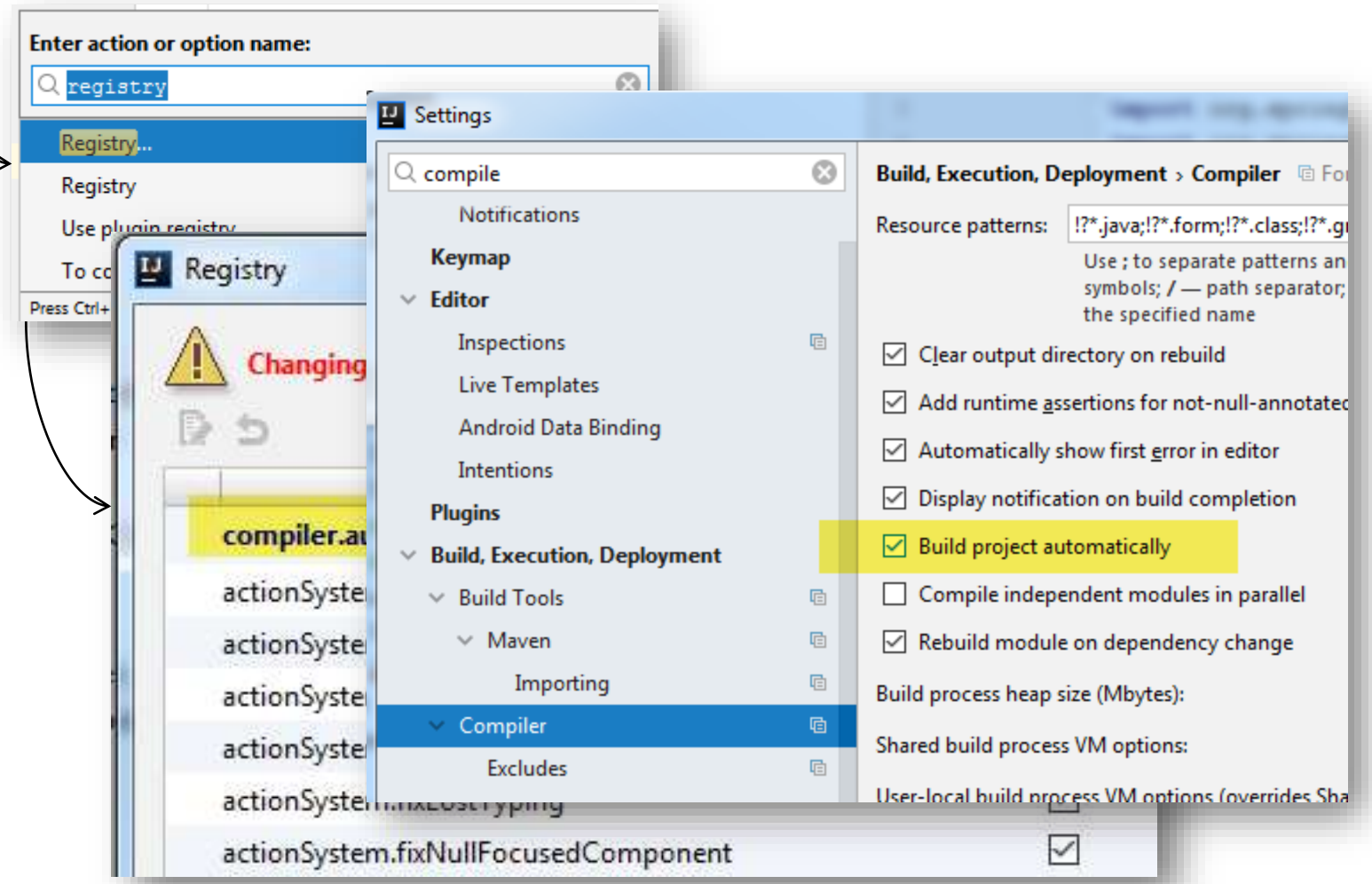


- Add @Controller annotation to the controller class
- Map a suffix property: `spring.mvc.view.suffix=.jsp`
- Create your RequestMappings:

```
@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