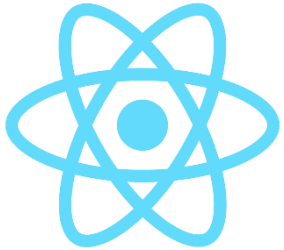
Day 11

An Application Stack

- A group of independent components that work together to support an application
- Each component
  - Implemented by a one or more technologies eg. Node, Spring, Flask
  - Requires resources and platform to run eg. physical hardware, cloud
- Common components include but not limited to
  - Client side application framework
  - Server side application framework
  - Persistence store
- Common application stack
  - MEAN
  - LAMP

Typical 3 Tier Application Stack

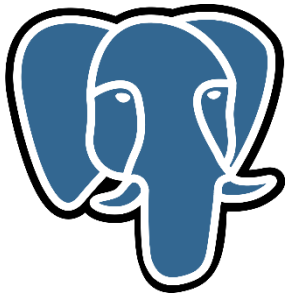




Client



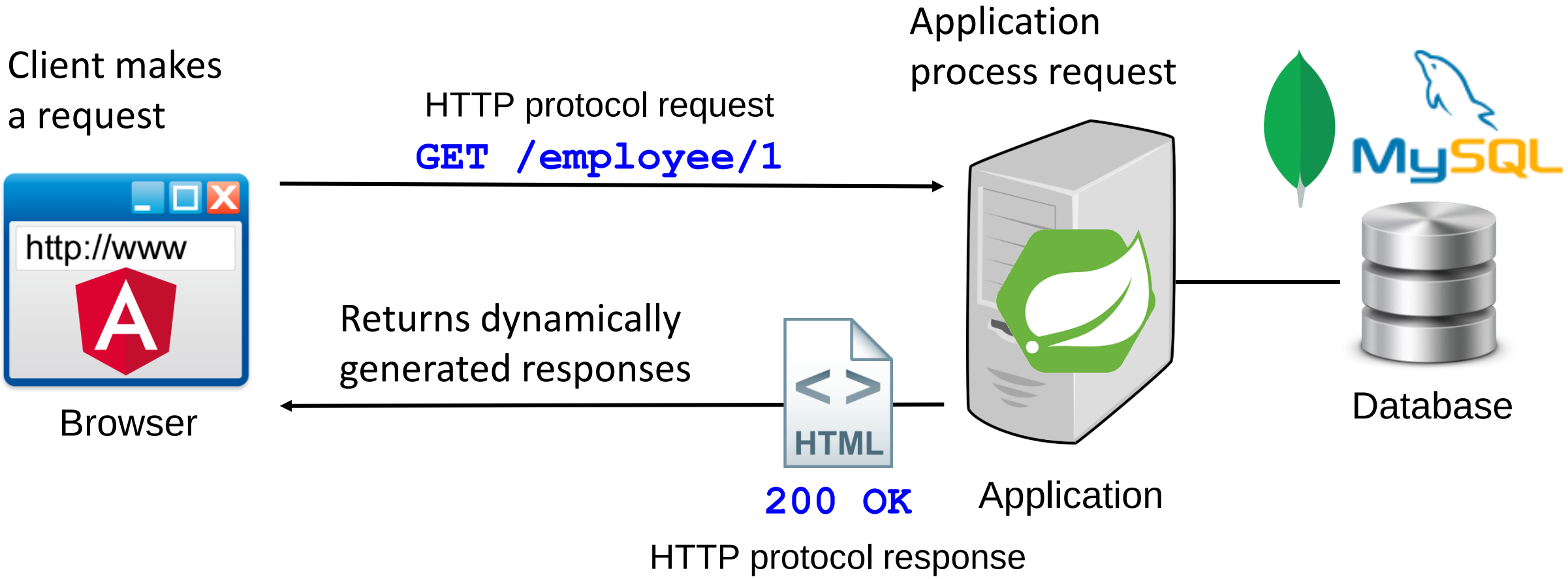

Server





Persistence

Web Application



URL - Uniform Resource Locator

https://iss.nus.edu.sg

https://iss.nus.edu.sg:80/index.html

Server, resolved to an IP address

Port number

Resource name

HDB - Blocks and Units

IP Addresses

- Unique identifiers assigned to any device that connects to the Internet
  - Usually assigned by the network provider that you connect to

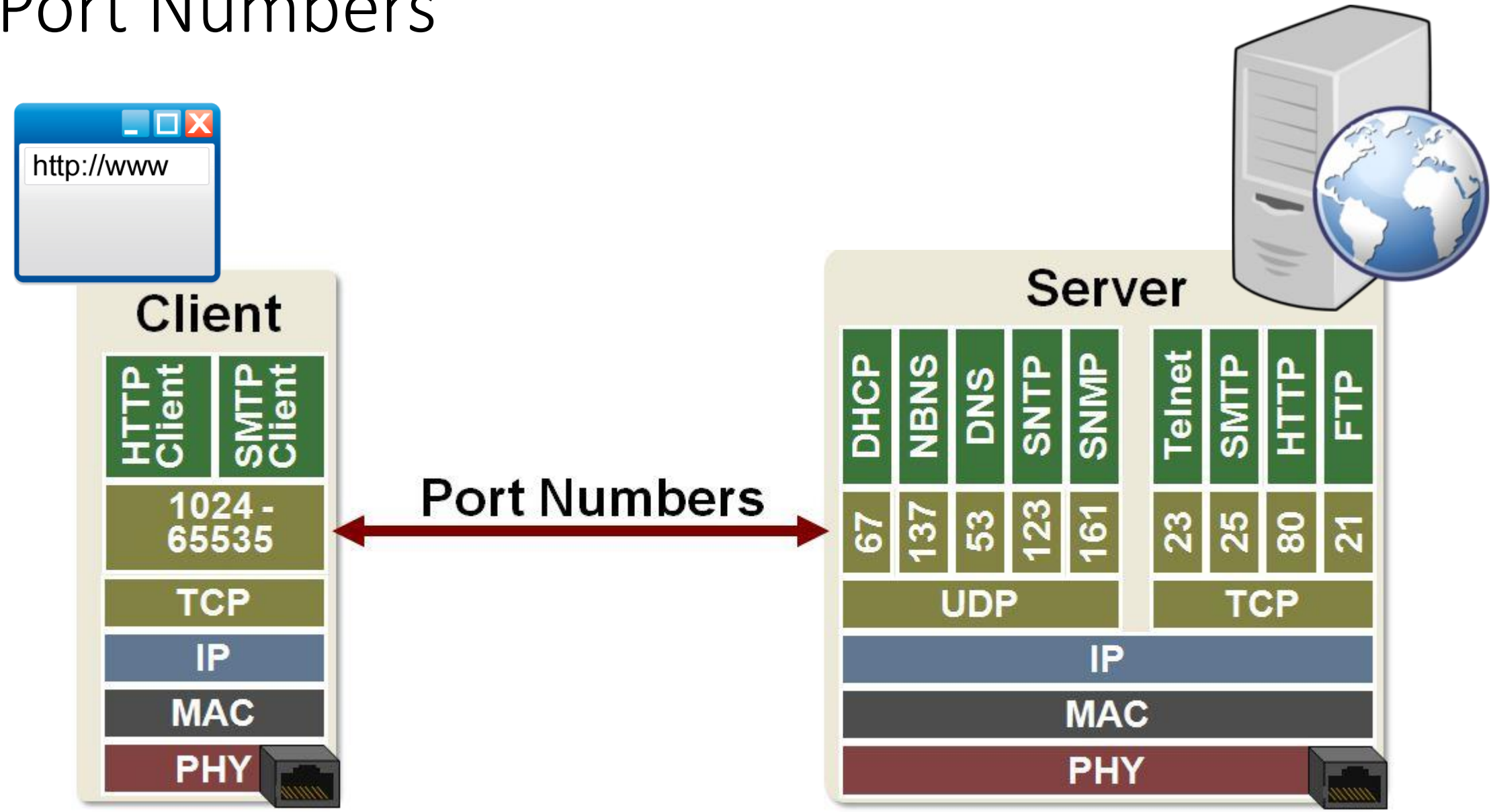
192 . 168 . 122 . 1

11000000.10101000.01111010.00000001

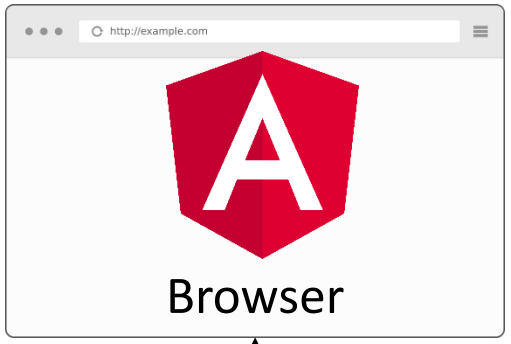
8 bits | 8 bits | 8 bits | 8 bits

32 bits == 4 bytes

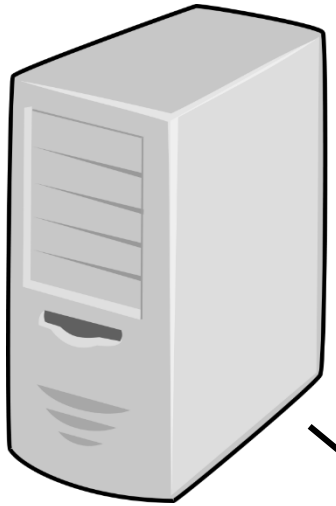
Port Numbers



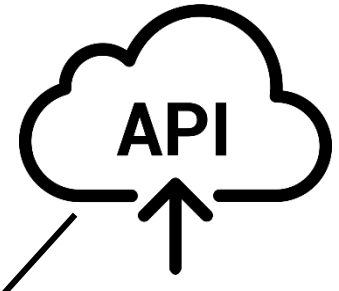
Web Application

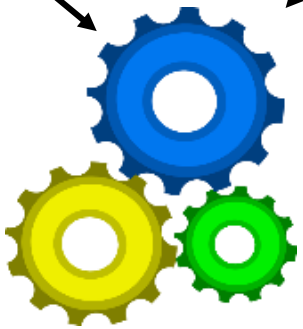


A request is sent to the server



Application framework receives the request, dispatches it to the application

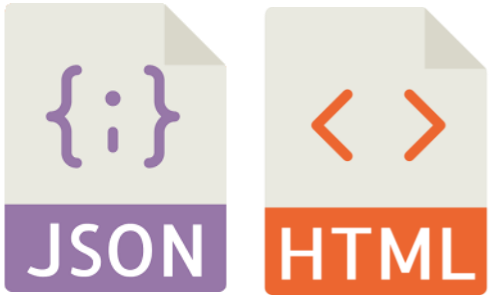




Web Application

Application generates HTML/JSON combining data from multiple sources

HTML/JSON is sent back to the browser



JavaScript runs on the browser to provide interactivity

Create and Initialize a SpringBoot Application

spring initializr

Project
- Maven Project
- Gradle Project

Language
- Java
- Kotlin
- Groovy

Spring Boot
- 2.6.0 (SNAPSHOT)
- 2.6.0 (M1)
- 2.5.4 (SNAPSHOT)
- 2.5.3
- 2.4.10 (SNAPSHOT)
- 2.4.9

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	Jar / War
Java	16 / 11 / 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

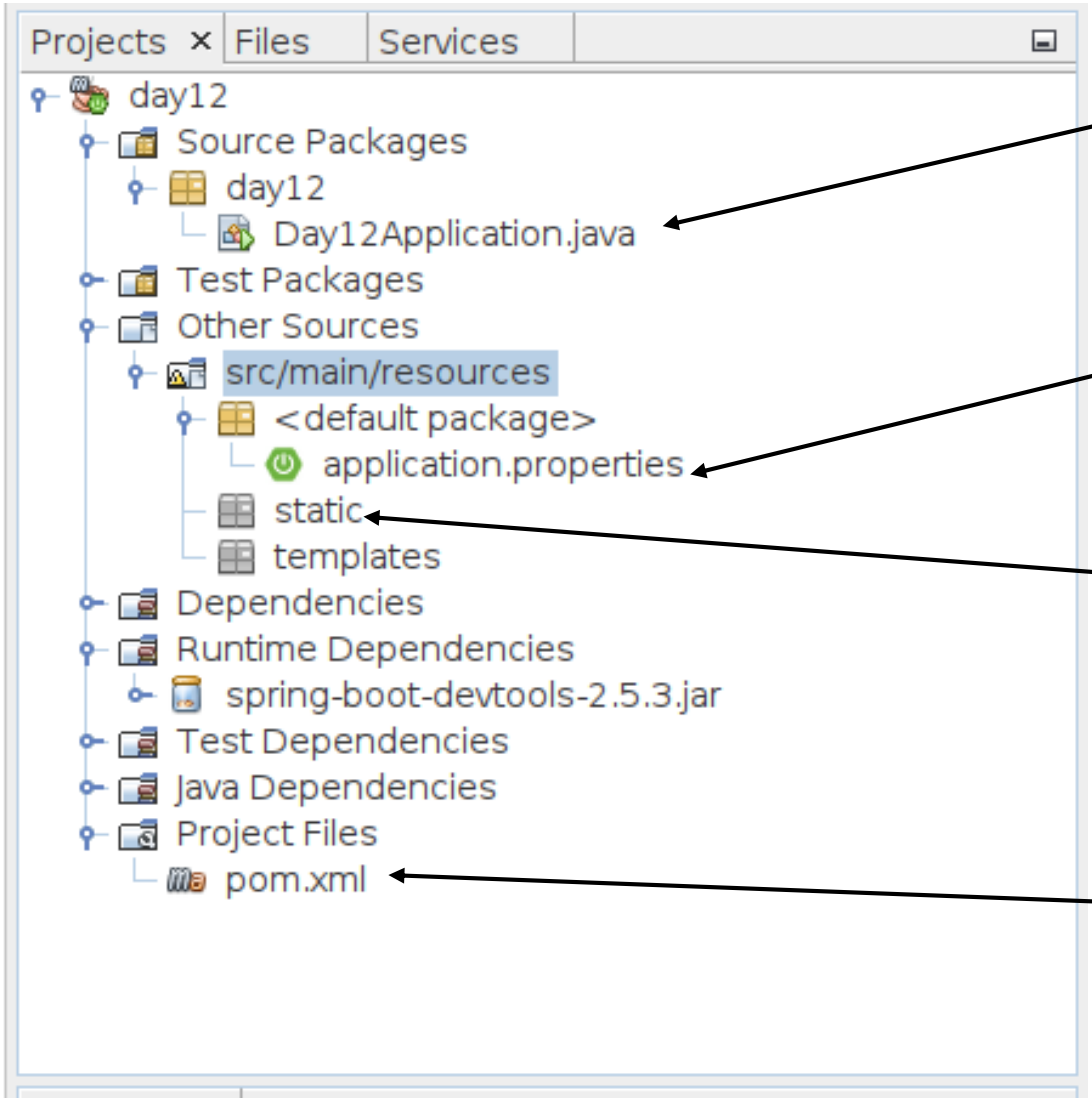
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

https://start.spring.io/

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | SHARE...

Generated SpringBoot Project



Application startup

Application properties, initially empty

For static resources eg. HTML, images

pom.xml

View in NetBeans 12.x with NB SpringBoot plugin

SpringBoot Startup Class

Annotation to indicate that this is a Spring Boot application.
This annotation enables auto configuration and scanning for other components

```
@SpringBootApplication
public class MyApplication {

   public static void main(String[] args) {
      SpringApplication app =
            new SpringApplication(MyApplication.class);

      System.out.println("Starting application on port 8080");

      app.run(args);
   }
}
```

Create an instance of Spring application

Run the application with the command line arguments

The above is a modified version of a generated SpringBoot startup class

Build and Run

- Compile application

```
mvn compile
```

- Package application including compile
  - JAR file is in target directory

```
mvn package
```

- Run application

```
mvn spring-boot:run
```

  - Run JAR file

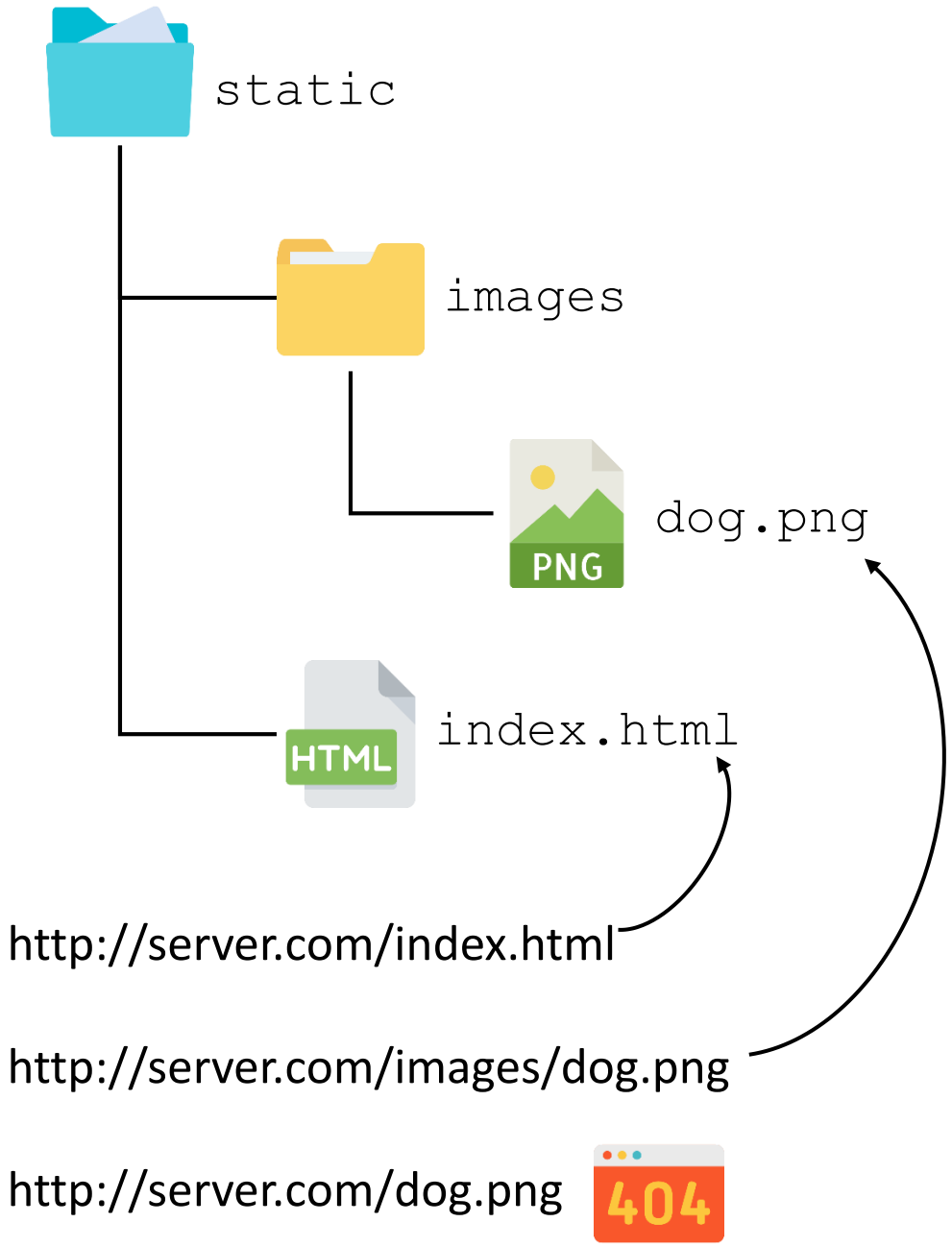
```
java -jar day12-0.0.1-SNAPSHOT.jar
```

- Clean build artifacts

```
mvn clean
```

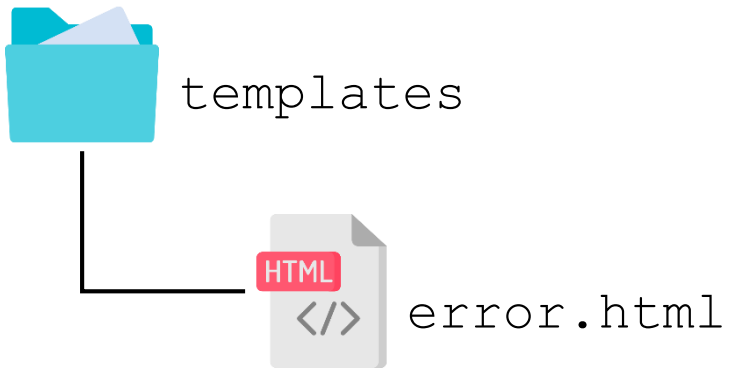
Serving Static Resources

- SpringBoot serve static resources from `resources/static` directory
  - Place HTML, video, images, CSS, JavaScript into this directory
- `static` is document root
  - All resources are rooted under this directory
- Files in static are bundled into the final application JAR!



Custom Error Page

- Display default error page whenever resource is not found
  - Whitelabel Error Page
- Custom error page
  - Create a file call `error.html` in `resources/templates` directory
  - Error file must be called `error.html`
- Require Thymeleaf to be installed



Processing Command Line Arguments

- Process command line arguments by instantiating `DefaultApplicationArugments`
  - Use `String[]` from `main()` as parameter
- Options are passed as `--optionName=value`, eg. `--port=3000`
- Processing command line arguments eg. `--port=3000`
  - Check if argument is set, `cliArgs.containsOption("port")`
    - Returns boolean
  - Read the value, `cliArgs.getOptionValues("port")`
    - Returns `List<String>`, multiple values if the same argument is used multiple times
- Passing arguments
  - `java -jar MyApp.jar --port=3000`
  - `mvn spring-boot:run -Dspring-boot.run.arguments="--port=3000 <space>--logLevel=TRACE"`

Setting Port With Command Line Argument

```
public static void main(String[] args) {
    SpringApplication app = new StringApplication(MyApplication.class);

    String port = "8080";
    ApplicationArguments cliOpts = new DefaultApplicationArguments(args);
    if cliOpts.containsOption("port")
        port = cliOpts.getOptionValues("port").get(0); // get the first value

    app.setDefaultProperties(
        Collections.singletonMap("server.port", port)
    );

    System.out.printf("Application started on port %d\n", port);
    app.run(args);
}
```

Parse the command line arguments

Get the value of port if it is set from command line

Set the port to listen before starting the application

Start the application

Enable Logging

Configure a method to create an instance of logger

```
@SpringApplication
public class MyApplication {
    ...
    @Bean
    public CommonsRequestLoggingFilter log() {
        CommonRequestLoggingFilter logger =
              new CommonRequestLoggingFilter();
        logger.setIncludeClientInfo(true);
        logger.setIncludeQueryString(true);
        return logger;
    }
}
```

Enable logging with the following key in `application.properties`

Other values include `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`

```
application.properties
logging.level.org.springframework.web.filter.CommonsRequestLoggingFilter=DEBUG
```