



Day 11



The World Wide Web

Hypertext references other documents on the server

Hypertext document server



Browser to display the hypertext

Other hypertext documents on the server





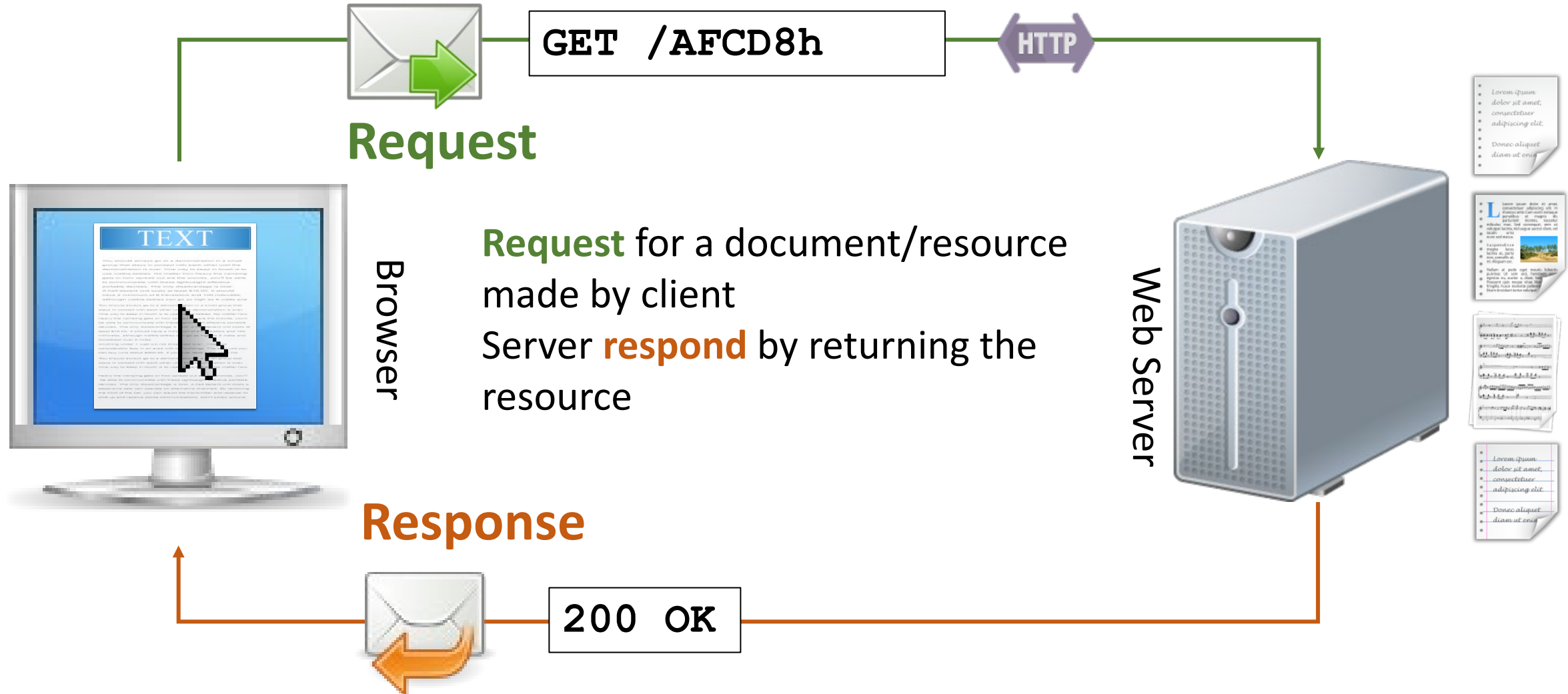
The World Wide Web

- Developed by Tim Berners-Lee in 1990 at CERN
 - To organize research documents available on the Internet
- Research documents are hypertext
 - Contains references to other research paper
 - Could go from paper to paper following a particular topic
- Research document (hypertext) would be authored in HTML
- References within the research document would linked with hyperlink
 - A hyperlink would reference a document on a server
 - When a link is activated (clicked), the document would be retrieved from the server and displayed
- Combining the idea of FTP/file server with hypertext/hyperlinks





HTTP Protocol

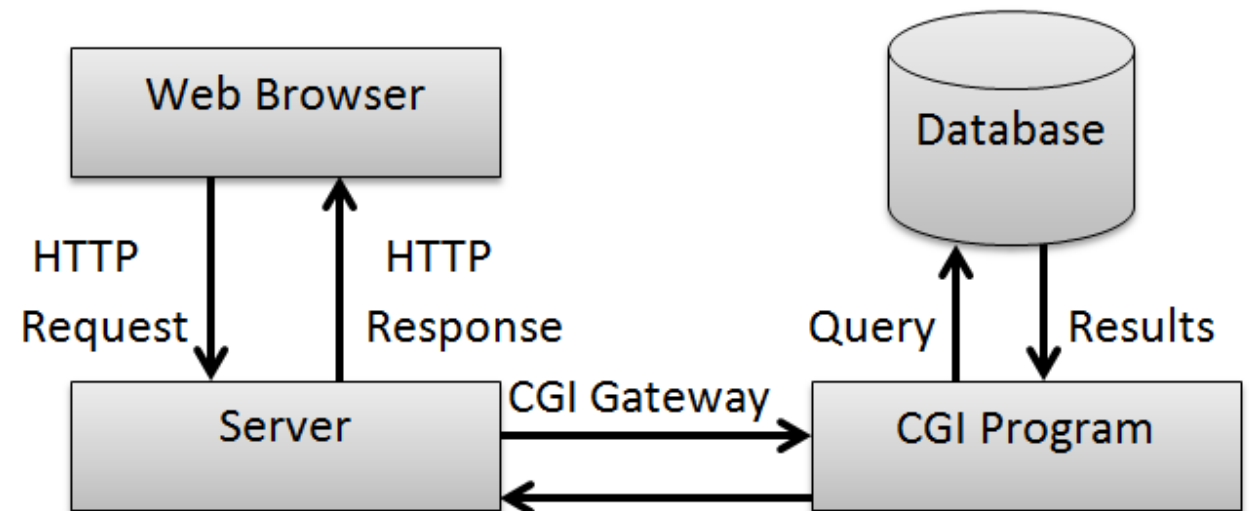




Web Application

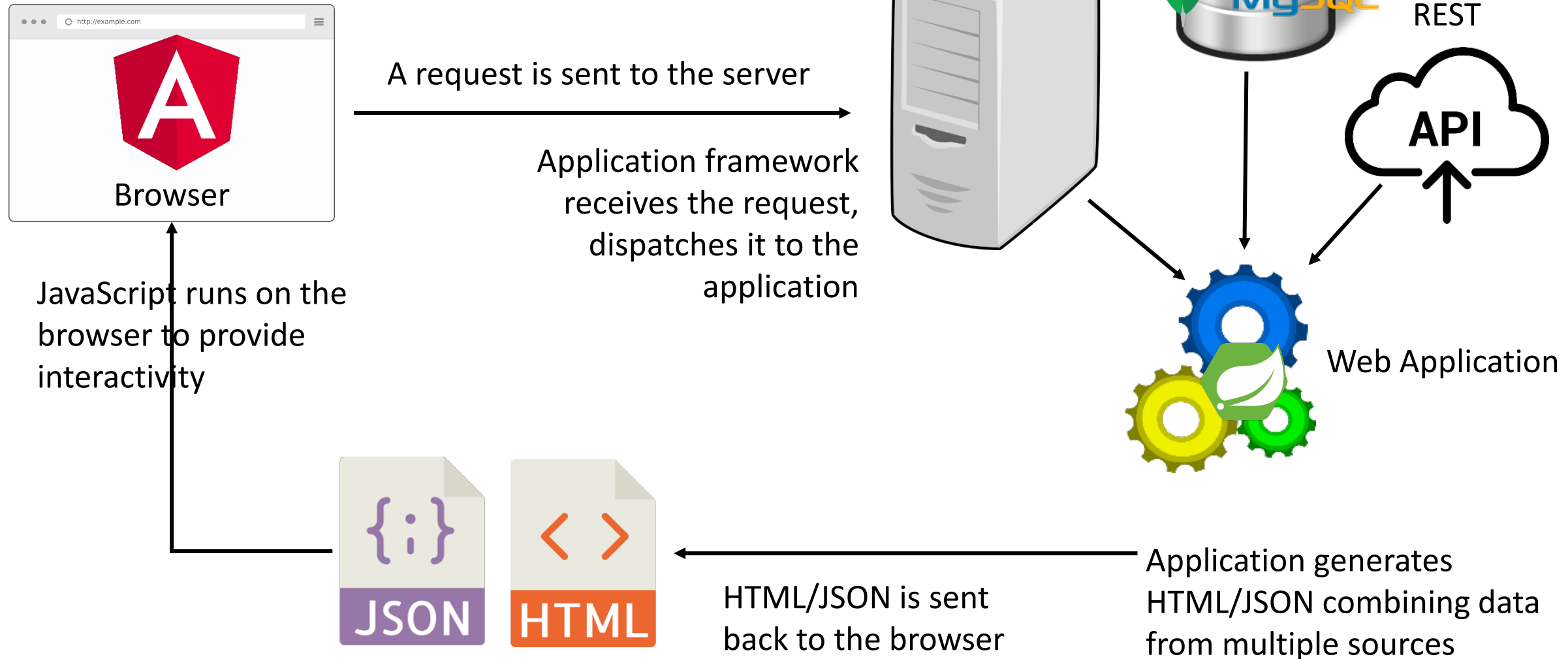
- NCSA (National Center for Supercomputing Applications) create a way for programs to be executed inside a web server
 - The application could return non static dynamically generated hypertext documents!
 - Content no longer limited by what is uploaded to the web server
- This environment for executing applications inside a web server is call CGI (Common Gateway Interface)

- Early CGIs were Perl and Bash shell scripts





Web Application Today



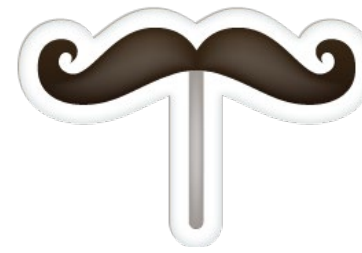
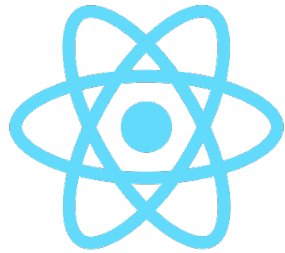


An Application Stack

- 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



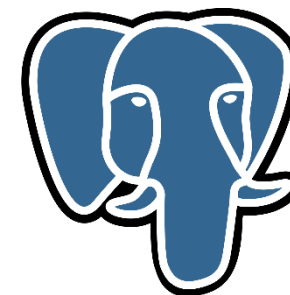
Typical Web Application Stack



Client




Server



Persistence



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.

GENERATE CTRL + G

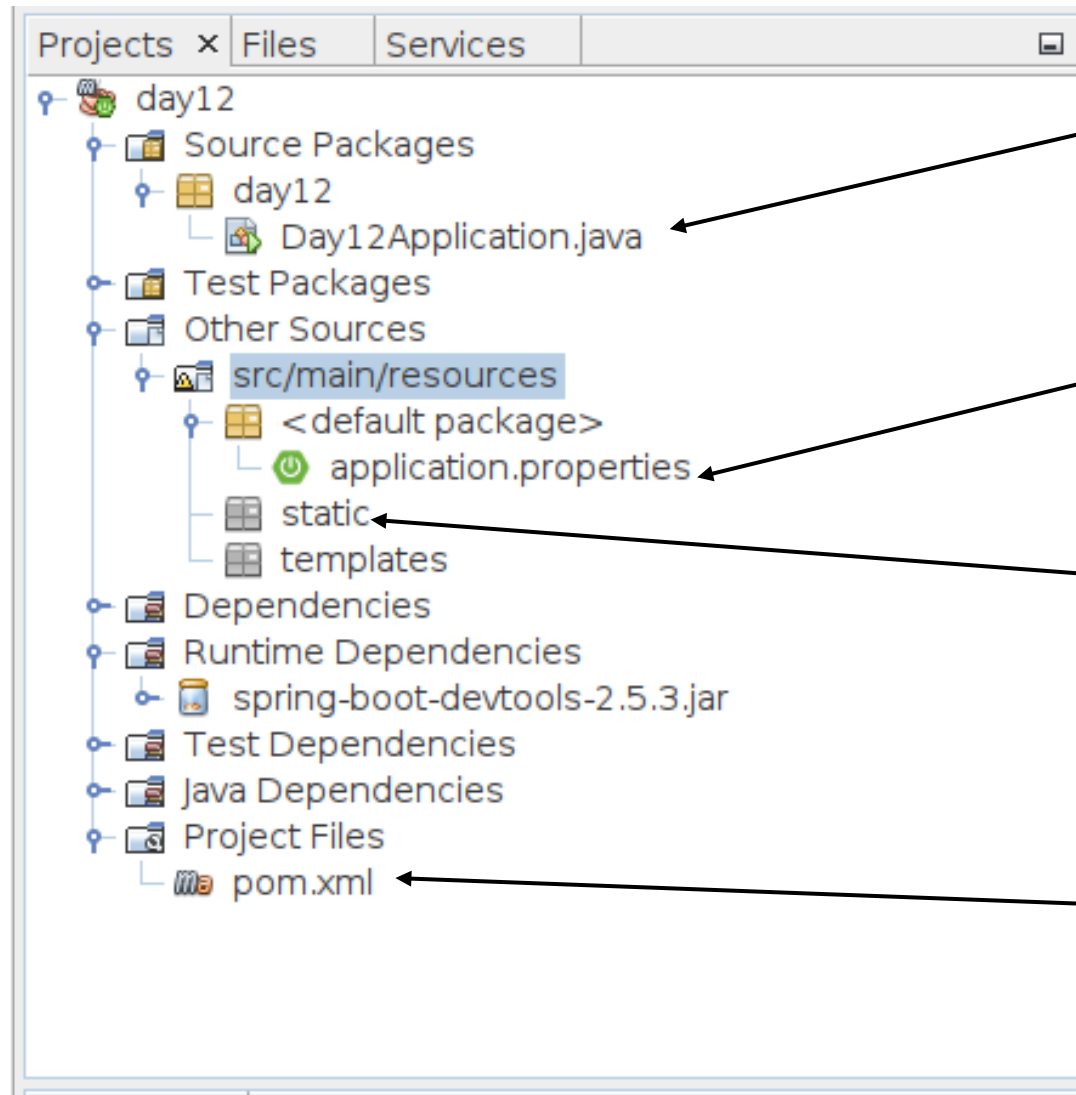
EXPLORE CTRL + SPACE

SHARE...

<https://start.spring.io/>



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
```

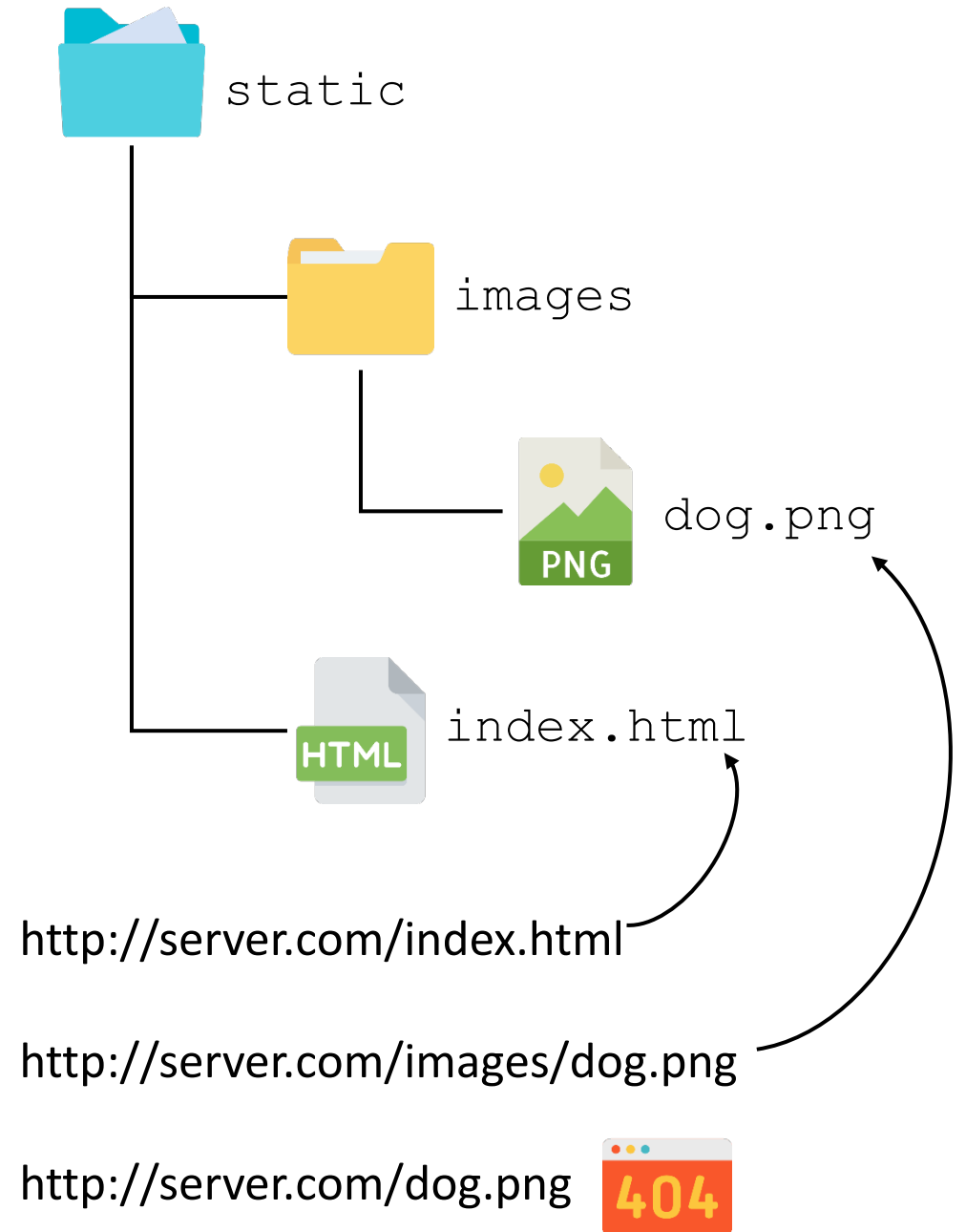
- Clean build artifacts

```
mvn clean
```



Serving Static Resources

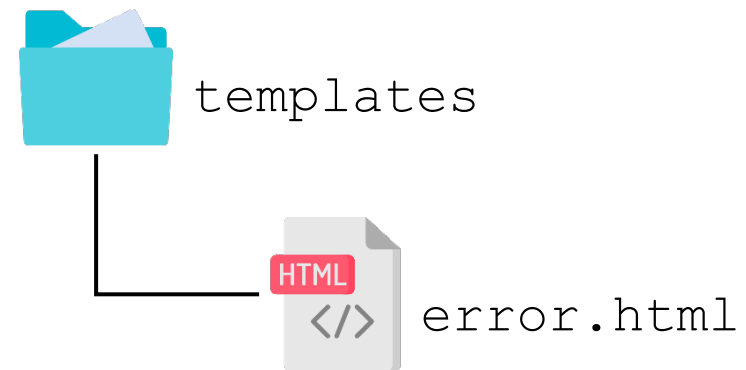
- SpringBoot serve static resources from `resources/static` directory
 - Place HTML, video, images, CSS, JavaScript into this directory
- `static` is document root
 - All resources are rooted under this directory
- Files in static are bundled into the final application JAR!





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





Changing the Port Number

- In `application.properties` file

```
server.port=3000
```

- With environment variable

- Windows

```
set SERVER_PORT=3000
```

- OSX/Linux

```
export SERVER_PORT=3000
```

- From command line

```
mvn spring-boot:run -Dserver.port=3000
```

```
java -Dserver.port=3000 -jar myapp.jar
```

- A port number of 0 means random unused port



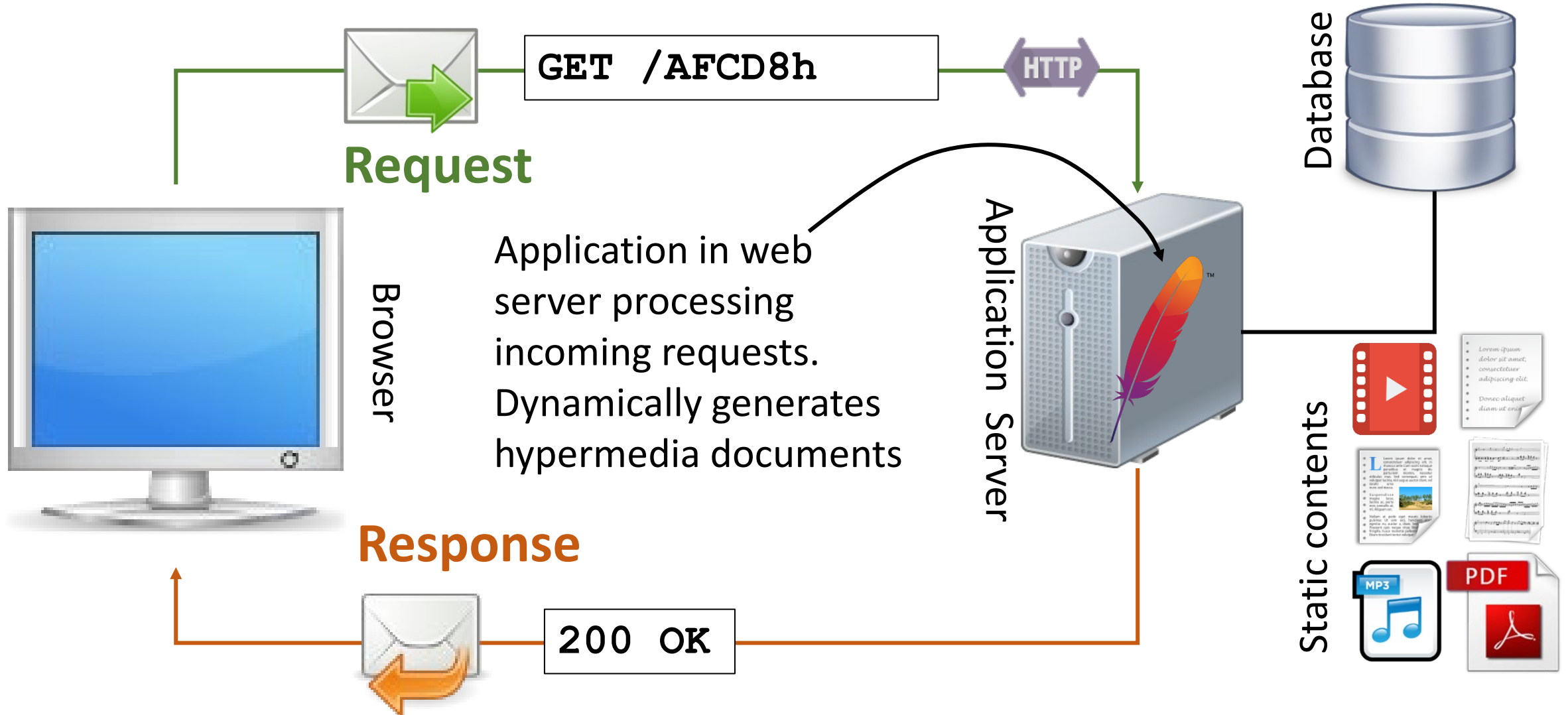
Deployment Demo

Deploy to Railway



Unused

Web Applications

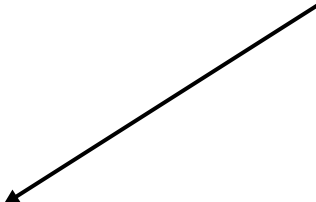




Enable Logging

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

Configure a method to
create an instance of 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
```



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 SpringApplication(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