ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Spring Workshop

1 - Basics

Agenda

- Gradle
- Controller
- DI / IoC
- Configuration
- Profiles
- Logging

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Gradle

- Automation Tool
  - Building
  - Download dependencies
  - Running various scripts
- Scalable (Multi-Project)
- Runs on Groovy or Kotlin
- Alternative to Maven
- Default for Spring since v6

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Gradle Example - Plugins

```
plugins {
    id 'org.springframework.boot' version '3.0.0'
    id 'io.spring.dependency-management' version '1.1.0'
    id 'java'
}
```


Gradle Example - Metadata

```
plugins {
    id 'org.springframework.boot' version '3.0.0'
    id 'io.spring.dependency-management' version '1.1.0'
    id 'java'
}

group = 'com.softarc'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Gradle Example - Repositories

```
plugins {
    id 'org.springframework.boot' version '3.0.0'
    id 'io.spring.dependency-management' version '1.1.0'
    id 'java'
}

group = 'com.softarc'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

repositories {
    mavenCentral()
    maven { url 'https://repo.spring.io/milestone' }
}
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Gradle Example - Dependencies

```
plugins {
    id 'org.springframework.boot' version '3.0.0'
    id 'io.spring.dependency-management' version '1.1.0'
    id 'java'
}

group = 'com.softarc'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

repositories {
    mavenCentral()
    maven { url 'https://repo.spring.io/milestone' }
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter'
  compileOnly 'org.projectlombok:lombok'
  annotationProcessor 'org.projectlombok:lombok'
  testImplementation 'org.springframework.boot:spring-boot-starter-webflux'
  developmentOnly 'org.springframework.boot:spring-boot-devtools'
}
```

Gradle Example - Task Customization

```
plugins {
    id 'org.springframework.boot' version '3.0.0'
    id 'io.spring.dependency-management' version '1.1.0'
    id 'java'
}

group = 'com.softarc'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

repositories {
    mavenCentral()
    maven { url 'https://repo.spring.io/milestone' }
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter'
  compileOnly 'org.projectlombok:lombok'
  annotationProcessor 'org.projectlombok:lombok'
  testImplementation 'org.springframework.boot:spring-boot-starter-webflux'
  developmentOnly 'org.springframework.boot:spring-boot-devtools'
}

tasks.named('test') {
    useJUnitPlatform()
}
```


private void executeLoad(long timeout, int usersCount) {
showDebugInfo(timeout);
Load.setPages(URL, parsingTimeout);
Load.setTimeout(timeout);
List<Load> threads = new ArrayList<>();
for (int i = 0; i < usersCount; i++) {
threads.add(new Load(this.URL));
}
logger.info(s: usersCount +
for (Load thread : threads)
thread.start();
}
logger.info(s: "All threads are started");
progressInfo(timeout);
System.out.print(".....DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout,
}
Demo

Useful Tools

- IDE
- Lombok
- MapStruct
  - Mapping Library
  - DTOs, Domain Models <-> Entity Classes
- SDKMAN! (requires WSL)
- google-code-format prettier

A First Controller

```
package com.softarc.eternal;

public class CustomersController {
}
```


A First Controller: Assigning the "Entry Url"

```
package com.softarc.eternal;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController()
@RequestMapping("/api/customers")
public class CustomersController {
}
```

A First Controller: Adding a Method

```
package com.softarc.eternal;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/customers")
public class CustomersController {
  public String index() {
    return "This is a customer";
  }
}
```

A First Controller: /api/customers/show

```
package com.softarc.eternal;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/customers")
public class CustomersController {

  @RequestMapping(method = RequestMethod.GET, path = "/show")
  public String index() {
    return "This is a customer";
  }
}
```

A First Controller: /api/customers/show (modern)

```
package com.softarc.eternal;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController("/api/customers")
public class CustomersController {

  @GetMapping("/show")
  public String index() {
    return "This is a customer";
  }
}
```

A First Controller: /api/customers

```
package com.softarc.eternal;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController("/api/customers")
public class CustomersController {

  @GetMapping()
  public String index() {
    return "This is a customer";
  }
}
```

A First Controller: /api/customers/1

```
package com.softarc.eternal;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController("/api/customers")
public class CustomersController {

  @GetMapping("/{id}")
  public String index(@PathVariable("id") Long id {
    return "This is a customer";
  }
}
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE


private void executeLoad(long timeout, int usersCount) {
showDebugInfo(timeout);
Load.setPages(URL, parsingTimeout);
Load.setTimeout(timeout);
List<Load> threads = new ArrayList<>();
for (int i = 0; i < usersCount; i++) {
threads.add(new Load(this.URL));
}
logger.info(s: usersCount +
for (Load thread : threads)
thread.start();
}
logger.info(s: "All threads are started");
progressInfo(timeout);
System.out.print(".....DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout,
}
Demo

Spring DI / IoC Container

- Bean: By Spring, instantiated and managed classes
- Mostly Singletons
- Bean Creation:
  - XML
  - Annotation (since Spring 2.5)
    - Component, Repository, Service,
  - Configuration (since Spring 3.0)
- Common pattern to customise Spring's behaviour

Dependency Injection: Register a Bean

```
public class CustomersRepository {

  public List<Customer> findAll() {
    List<Customer> customers = new ArrayList<>();
    customers.add(new Customer(1L, "Max", "Mustermann"));
    customers.add(new Customer(2L, "Anna", "Schneider"));

    return customers;
  }
}
```

Dependency Injection: Register a Bean

```
@Service
public class CustomersRepository {

  public List<Customer> findAll() {
    List<Customer> customers = new ArrayList<>();
    customers.add(new Customer(1L, "Max", "Mustermann"));
    customers.add(new Customer(2L, "Anna", "Schneider"));

    return customers;
  }
}
```

Singleton

Dependency Injection: Consume a Bean I

```
@RestController
@RequestMapping("/api/customers")
public class CustomersController {

  @Autowired
  public CustomersRepository repository;

  @GetMapping
  public List<Customer> index() {
    return this.repository.findAll();
  }
}
```

Injection
(hard to test)

Dependency Injection: Consume a Bean II

```
@RestController
@RequestMapping("/api/customers")
public class CustomersController {

  private final CustomersRepository repository;

  public CustomersController(CustomersRepository repository) {
    this.repository = repository;
  }

  @GetMapping
  public List<Customer> index() {
    return this.repository.findAll();
  }
}
```

Injection
(Testable)

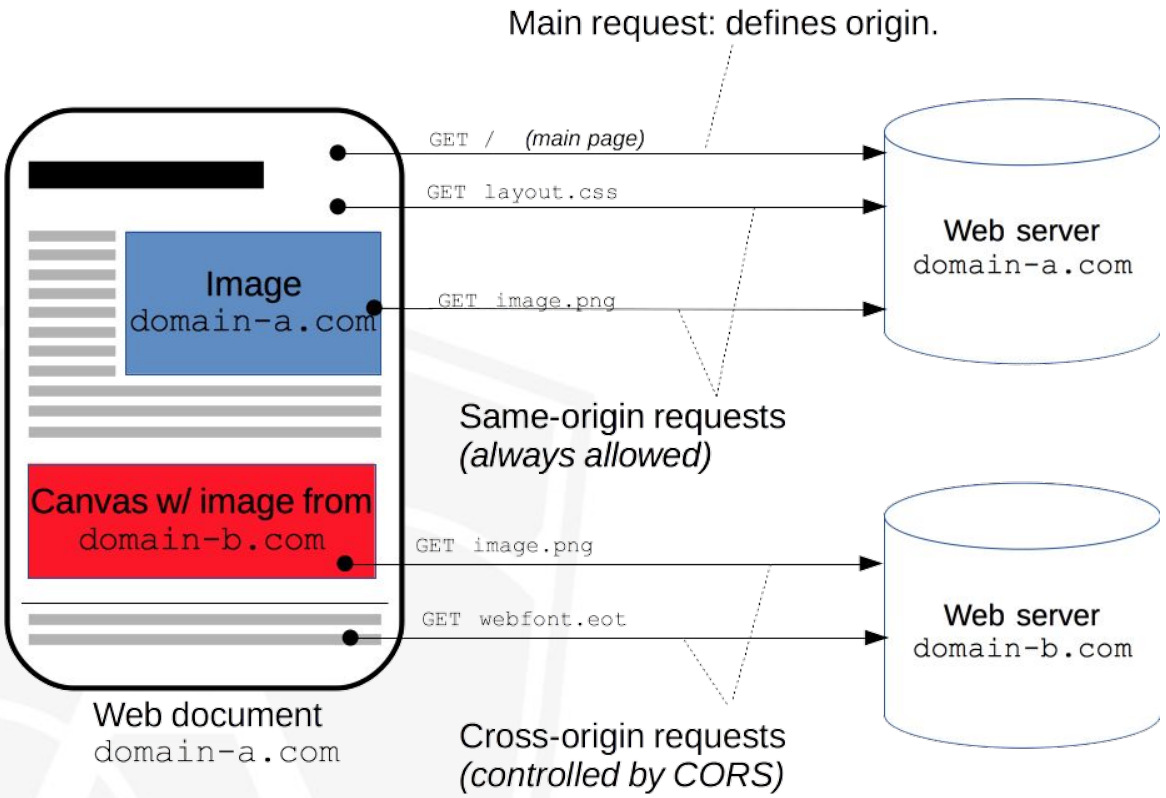

private void executeLoad(long timeout, int usersCount) {
showDebugInfo(timeout);
Load.setPages(URL, parsingTimeout);
Load.setTimeout(timeout);
List<Load> threads = new ArrayList<>();
for (int i = 0; i < usersCount; i++) {
threads.add(new Load(this.URL));
}
logger.info(s: usersCount +
for (Load thread : threads)
thread.start();
}
logger.info(s: "All threads are started");
progressInfo(timeout);
System.out.print(".....DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout,
}
Demo

CORS: Cross-Origin Resource Sharing

- Cross-Site Requests not permitted by default
- Security Feature
  - Also for the Server
- Needs to be enabled on the Server
- xhr and fetch use it automatically

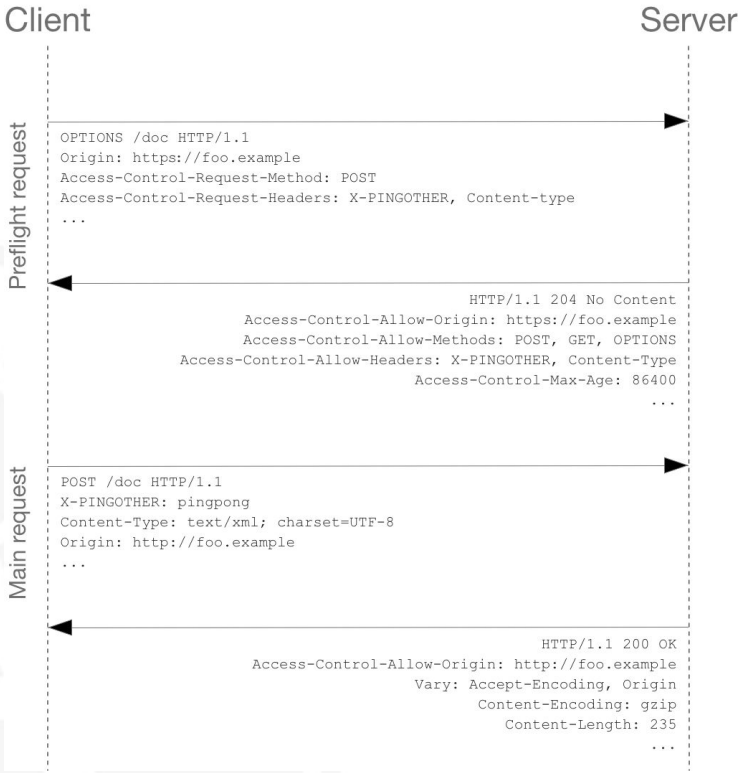
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

CORS: Cross-Origin Resource Sharing



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/cors_principle.png

CORS: Preflight on POST, PUT,…



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/preflight_correct.png

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Enabling CORS

```
package com.softarc.eternal;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig {

  @Bean
  public WebMvcConfigurer corsConfigurer() {
    return new WebMvcConfigurer() {
      @Override
      public void addCorsMappings(CorsRegistry registry) {
        registry
          .addMapping("/api/**")
          .allowedOrigins("http://localhost:4200")
          .allowedMethods("GET", "POST", "OPTIONS", "PUT", "DELETE")
          .allowCredentials(true);
      }
    };
  }
}
```

Lab Time

@Configuration & @Bean

- "FactoryBean"
- Useful for conditional Bean registration
- In conjunction with Application Properties
- Provides central Overview of Beans

@Configuration

```
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfiguration {
}
```

@Configuration

```
import com.softarc.eternal.data.CustomersRepository;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfiguration {

  CustomersRepository getCustomersRepository() {
    return new CustomersRepository();
  }
}
```

@Configuration

```
import com.softarc.eternal.data.CustomersRepository;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfiguration {

  @Bean
  CustomersRepository getCustomersRepository() {
    return new CustomersRepository();
  }
}
```


private void executeLoad(long timeout, int usersCount) {
showDebugInfo(timeout);
Load.setPages(URL, parsingTimeout);
Load.setTimeout(timeout);
List<Load> threads = new ArrayList<>();
for (int i = 0; i < usersCount; i++) {
threads.add(new Load(this.URL));
}
logger.info(s: usersCount +
for (Load thread : threads)
thread.start();
}
logger.info(s: "All threads are started");
progressInfo(timeout);
System.out.print(".....DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout,
}
Demo

Application Properties

- Configuration files
- Can be overridden by environment variables
- Profiles can override default properties
- Property types
  - Spring-based
  - Application-specific
- Injectable via
  - @Value: unsafe
  - @ConfigurationProperties
- YAML or default format
- Can include encrypted data

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

application.yml

```yaml
#application.yml
app:
  customers:
    showGdpr: true
```

```yaml
#application.yml - Alternative
app.customers.showGdpr: true
```

```
# application.properties
app.customers.showGdpr = true
```

Configuration Reference

https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Loading Properties via @Value

```
@Service
public class CustomersRepository {

  @Value("${app.customers.showGdpr}")
  private final Boolean showGdpr;

  public List<Customer> findAll() {
    List<Customer> customers = new ArrayList<>();
    customers.add(new Customer(1L, "Max", "Mustermann", true));
    customers.add(new Customer(2L, "Anna", "Schneider", false));

    return customers
      .stream()
      .filter(customer -> this.showGdpr || customer.getSignedGdpr())
      .toList();
  }
}
```

Loading Properties via @Value & @ConfigurationProperties I

```
@ConfigurationProperties("app.customers")
@Configuration
@Data
public class CustomersProperties {

  private Boolean showGdpr;
}
```

Loading Properties via @Value & @ConfigurationProperties II

```
public class CustomersRepository {

  private final CustomersProperties properties;

  public CustomersRepository(CustomersProperties properties) {
    this.properties = properties;
  }

  public List<Customer> findAll() {
    List<Customer> customers = new ArrayList<>();
    customers.add(new Customer(1L, "Max", "Mustermann", true));
    customers.add(new Customer(2L, "Anna", "Schneider", false));

    return customers
      .stream()
      .filter(customer ->
        this.properties.getShowGdpr() || customer.getSignedGdpr()
      )
      .toList();
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Loading Properties via @Value & @ConfigurationProperties III

```
@Configuration
public class AppConfiguration {

  @Bean
  CustomersRepository getCustomersRepository(
    CustomersProperties customersProperties
  ) {
    return new CustomersRepository(customersProperties);
  }
}
```


private void executeLoad(long timeout, int usersCount) {
showDebugInfo(timeout);
Load.setPages(URL, parsingTimeout);
Load.setTimeout(timeout);
List<Load> threads = new ArrayList<>();
for (int i = 0; i < usersCount; i++) {
threads.add(new Load(this.URL));
}
logger.info(s: usersCount +
for (Load thread : threads)
thread.start();
}
logger.info(s: "All threads are started");
progressInfo(timeout);
System.out.print(".....DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout,
}
Demo

Profiles

- Mostly used different configuration files
  - application-{profile}.yml
- Only override default configuration
- Can be used to
  - Save Credentials
  - Replace Implementation Classes
  - ...
  - Alternative to @ConditionalOn
- Groupable
- Environment Variable: SPRING_PROFILES_ACTIVE

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Logging

- Multiple Logging Mechanisms available
- Default is Logback
- Supports profile-based configuration
  - logback-stage.xml
- Different logging levels
- Logging levels configurable in application.yml
- Use Lombok's @Log (static property)


private void executeLoad(long timeout, int usersCount) {
showDebugInfo(timeout);
Load.setPages(URL, parsingTimeout);
Load.setTimeout(timeout);
List<Load> threads = new ArrayList<>();
for (int i = 0; i < usersCount; i++) {
threads.add(new Load(this.URL));
}
logger.info(s: usersCount +
for (Load thread : threads)
thread.start();
}
logger.info(s: "All threads are started");
progressInfo(timeout);
System.out.print(".....DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout,
}
Demo