Spring WebFlux

Introduction

By Sergey Kargopolov

What is Spring WebFlux?

- A non-blocking web framework from Spring,
- Handles large number requests with fewer resources,
- Supports reactive programming model.

Spring WebFlux (Reactive)

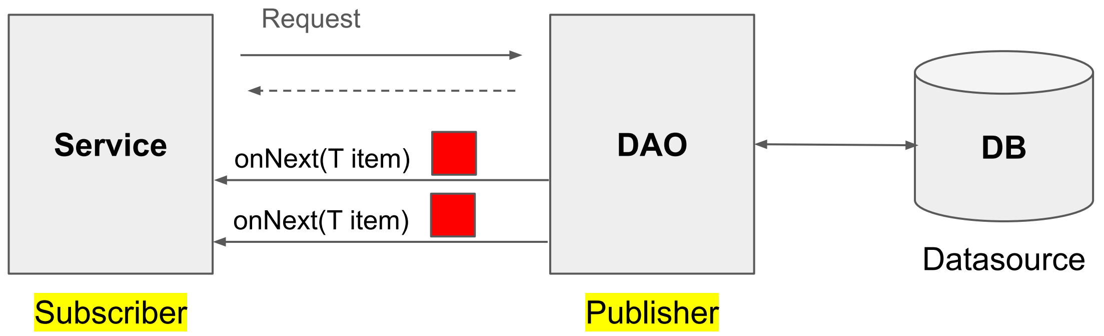
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
```

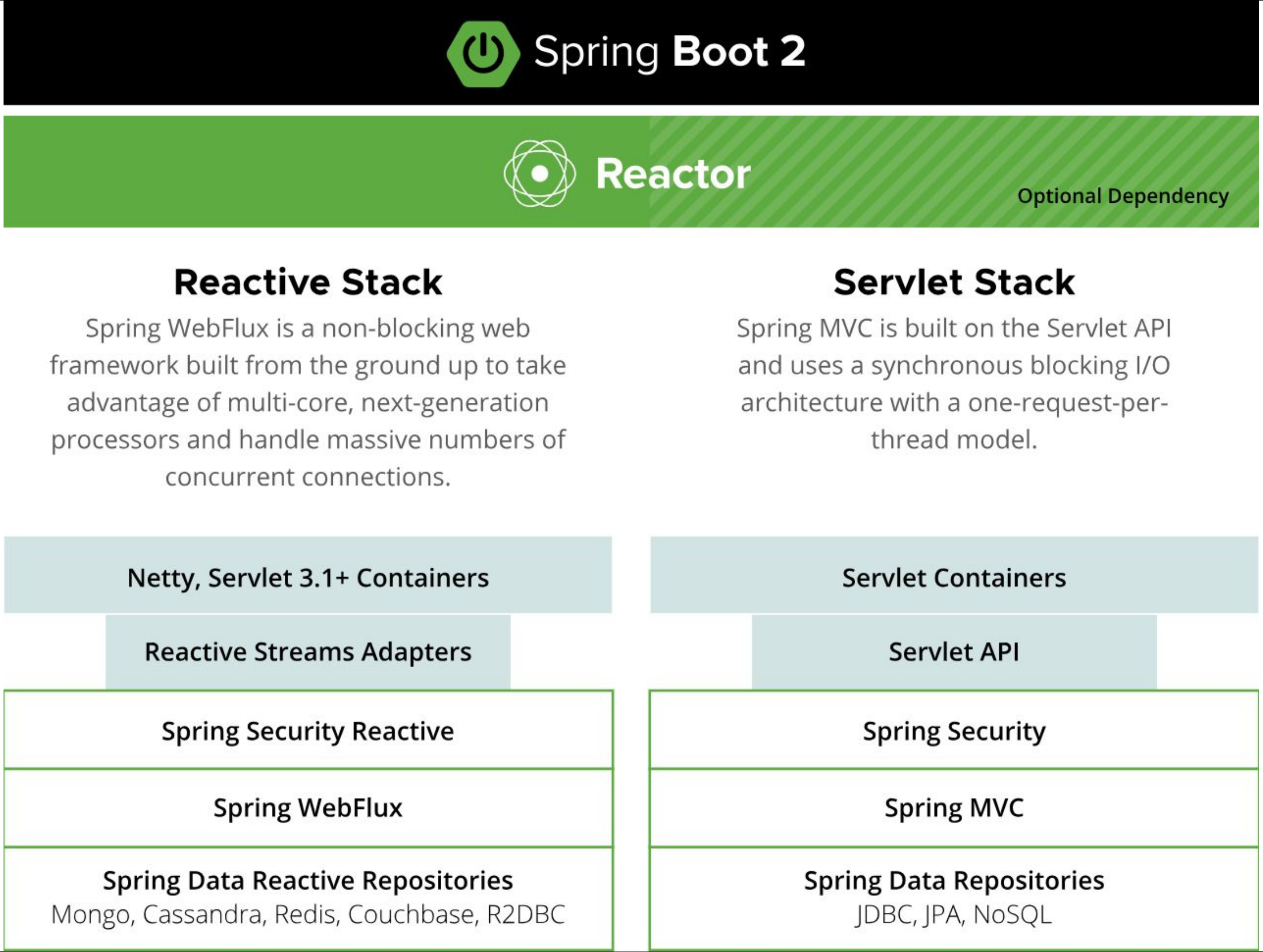
Spring WebMVC (Traditional/Blocking)

```
<dependency>
  <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

How Spring WebFlux is different?

- Reactive programming,



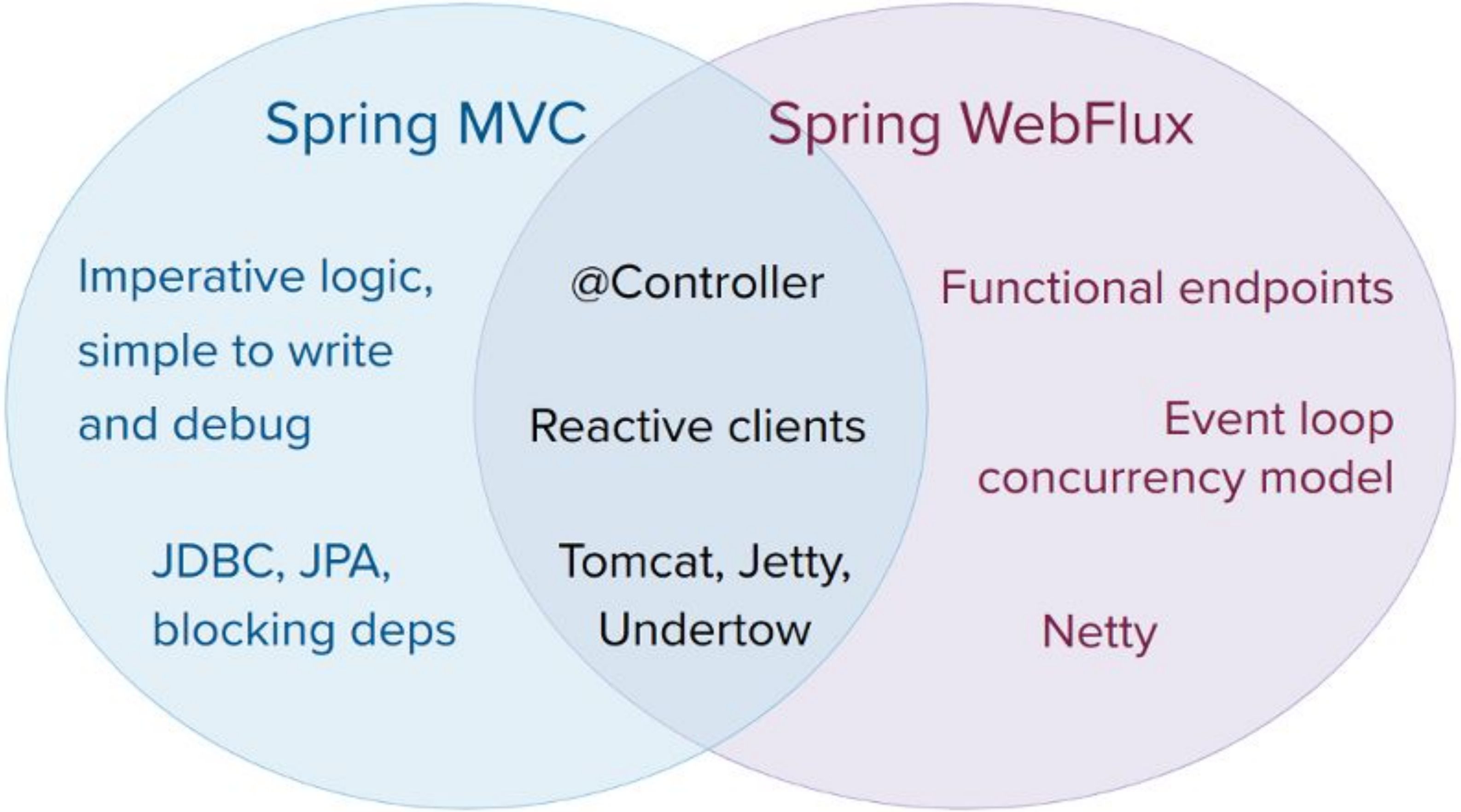


Reactive Stack

Spring WebFlux is a non-blocking web framework built from the ground up to take advantage of multi-core, next-generation processors and handle massive numbers of concurrent connections.

Servlet Stack

Spring MVC is built on the Servlet API and uses a synchronous blocking I/O architecture with a one-request-per-thread model.



By Sergey Kargopolov

WebFlux Functional Endpoints

Traditional (Annotation-based):

```
@RestController
@RequiredArgsConstructor

public class UsersController {

   private final UserService userService;

   @PostMapping("/users")

   public Mono<UserDto> createUser(@RequestBody @Validated CreateUserDTO user) {

       return userService.save(user);

   }
```

WebFlux Functional Endpoints

Functional Endpoints(functional programming):

```
@Configuration

public class RoutesConfig {

   @Bean

   RouterFunction<ServerResponse> usersRoutes(UsersRouteHandler usersRouteHandler) {

       return route(POST("/users"), usersRouteHandler::handleCreateUser);

   }

}
```

How Spring WebFlux is different?

- Reactive programming,
- Supports both imperative and reactive programming styles,

Imperative Programming (Spring WebMVC)

```
@RestController
public class UserController {
    ...
    @GetMapping("/users")
    public List<User> getUsers() {
        return userService.getUsers();
    }
}
```

Reactive Programming (Spring WebFlux)

```
@RestController
public class UserController {
    ...
    @GetMapping("/users")
    public Flux<User> getUsers() {
        return userService.getUsers();
    }
}
```

By Sergey Kargopolov