```
1) Synchronus based execution ( Blocking Thread )

2) Asynchronus based execution ( Non Blocking Thread )


=> Spring 5.x introduced Reactive Programming

=> In Spring 5.x 'starter-webflux' introduced


==================
Old Approach
==================
@RestController
public class WelcomeRestController{

    @GetMapping("/msg")
        public String getMsg(){
                return "Hello";
        }

}

===============
New Approach
===============

@Component
public class MessageRequestHandler{

        public Mono<ServerResponse> handle(ServletRequest request){
                return new ServerRespoinse.ok()

.contentType(MediaType.APPLICATION_JSON)

.body(BodyInserters.fromValue(data));
        }

}

@Configuration
public MsgRouter {

                @Bean
                pubilc RouterFunction<ServerResponse> route(MessageRequestHandler requestHandler){

                                return RouterFunctions.route(GET("/hello"))

.and(accept(MediaType.APPLICATION_JSON), MessageRequestHandler::handle);

                }

}


=============================
SprinBoot Reactive Example
============================

1) Create Boot application with 'Reactive Web' dependency

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

Note: Reactive Web dependency means 'starter-webflux' dependency. It will provide 'Netty' as default embedded container.

2) Create Binding class to response

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Greeting {

        private String msg;

}
```

3) Create Request Handler class like below

```
@Component
public class GreetingHandler {

        public Mono<ServerResponse> hello(ServerRequest request){

                return ServerResponse.ok()
                                           .contentType(MediaType.APPLICATION_JSON)
                                           .body(BodyInserters.fromValue(new Greeting("Hello
World")));
        }
}
```

4) Create Router class

```
@Configuration
public class GreetingRouter {

        @Bean
        public RouterFunction<ServerResponse> route(GreetingHandler greeting){
                return RouterFunctions
                              .route(GET("/hello")
                              .and(accept(MediaType.APPLICATION_JSON)), greeting::hello);
        }
}
```

5) Run the application and test it.