

El uso de aplicaciones de Spring Boot Kotlin se irá incrementando poco a poco porque Kotlin como lenguaje va adquiriendo mas cuota de mercado. Todos tenemos la necesidad de construir aplicaciones más rápido y Kotlin nos puede ayudar a ello. Vamos a ver un ejemplo sencillo de hola mundo. El primer paso que tenemos que realizar es acceder a [Spring Initalizr](#) y configurar nuestra aplicación para Spring Boot Kotlin. Para ello nos conectamos a su web y elegimos crear un proyecto con Java 11 , Gradle y Kotlin como lenguaje de programación por defecto.

The screenshot shows the Spring Initializr web application in a browser. The interface is dark-themed and includes a sidebar with a menu icon. The main content area is divided into several sections:
 - **Project:** Radio buttons for 'Maven Project' and 'Gradle Project' (selected).
 - **Language:** Radio buttons for 'Java' and 'Kotlin' (selected).
 - **Spring Boot:** Radio buttons for various versions, with '2.6.6' selected.
 - **Project Metadata:** Text input fields for 'Group' (com.arquitecturajava), 'Artifact' (kotlin), 'Name' (kotlin), 'Description' (kotlin hola mundo), and 'Package name' (com.arquitecturajava.kotlin).
 - **Packaging:** Radio buttons for 'Jar' (selected) and 'War'.
 - **Java:** Radio buttons for versions 18, 17, 11 (selected), and 8.
 - **Dependencies:** A section with a 'ADD DEPENDENCIES...' button and a 'Spring Web' dependency selected, with a 'WEB' tag.
 - **Buttons:** 'GENERATE', 'EXPLORE', and 'SHARE...' buttons are at the bottom.
 - **Footer:** A 'Mostrar todo' button is in the bottom right corner.

spring boot kotlin configuracion

Spring Boot Kotlin y Gradle

Como siempre para poder diseñar un proyecto de Spring Boot necesitamos añadir dependencias. En esta situación solo abordaremos el manejo de Spring Web para diseñar unos servicio REST de holamundo el resto las añadirá Spring Boot al fichero de build.gradle.kts.

```
import org.jetbrains.kotlin.gradle.tasks.KotlinCompile

plugins {
    id("org.springframework.boot") version "2.6.6"
    id("io.spring.dependency-management") version "1.0.11.RELEASE"
    kotlin("jvm") version "1.6.10"
    kotlin("plugin.spring") version "1.6.10"
    kotlin("plugin.jpa") version "1.6.10"
}

group = "com.arquitecturajava"
version = "0.0.1-SNAPSHOT"
java.sourceCompatibility = JavaVersion.VERSION_17

repositories {
    mavenCentral()
}

dependencies {
    implementation("org.springframework.boot:spring-boot-starter-web")
    implementation("com.fasterxml.jackson.module:jackson-module-kotlin")
    implementation("org.jetbrains.kotlin:kotlin-reflect")
    implementation("org.jetbrains.kotlin:kotlin-stdlib-jdk8")
    testImplementation("org.springframework.boot:spring-boot-starter-test")
}

tasks.withType<KotlinCompile> {
    kotlinOptions {
        freeCompilerArgs = listOf("-Xjsr305=strict")
    }
}
```

```
jvmTarget = "17"  
}  
}
```

```
tasks.withType<Test> {  
    useJUnitPlatform()  
}
```

Una vez descargado y descomprimido el proyecto . Podemos importar de forma directa su contenido en IntelliJ. Dispondremos de la clase `KotlinApplication` que se encarga de arrancar la aplicación y es muy similar a la que se usa con `SpringBoot`.

```
package com.arquitecturajava.kotlin
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication  
import org.springframework.boot.runApplication
```

```
@SpringBootApplication  
class KotlinApplication
```

```
fun main(args: Array<String>) {  
    runApplication<KotlinApplication>(*args)  
}
```

Es momento de construir una clase `Persona` con Kotlin cuyo código es muy reducido:

```
package com.arquitecturajava.kotlin
```

```
class Persona constructor (val nombre:String, val apellidos:String ,  
val edad:Int) {}
```

Nos queda construir un `@RestController` que use la clase `Persona` y genere una lista de personas que serán publicadas como JSON a través del servicio REST que vamos a

construir.

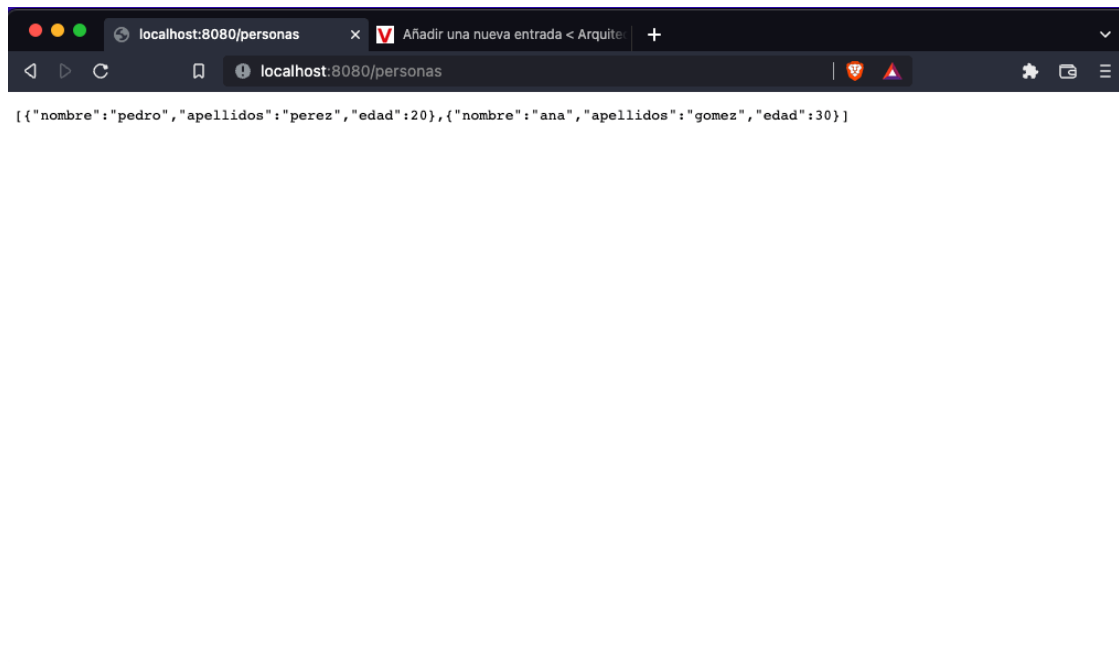
```
package com.arquitecturajava.kotlin

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

@RestController
class PersonaRest {

    @GetMapping("/personas")
    fun buscarTodas():MutableList<Persona> {
        return mutableListOf<Persona>(
Persona("pedro","perez",20),Persona("ana","gomez",30));
    }
}
```

Realizada esta operación es momento de desplegar la aplicación y ejecutarla en un navegador.



Acabamos de desplegar una aplicación de Spring Boot Kotlin de Hola Mundo . Como se puede observar sus similitudes con Java son enormes con la única diferencia de que el código es más compacto.

Otros artículos relacionados

- [Kotlin vs Java y el manejo de clases](#)
- [Kotlin Ranges y sentencia de Control](#)
- [Kotlin Destructuring](#)
- [Kotlin Delegación buenas prácticas.](#)
- [Spring Boot JPA y su configuración](#)
- [¿Qué es Spring Boot?](#)
- [¿Qué es Gradle?](#)
- [Curso Kotlin y buenas prácticas](#)