(/)

¿Por qué elegir Spring como su marco de Java?

Última modificación: 29 de julio de 2019

por Kumar Chandrakant (https://www.baeldung.com/author/kumar-chandrakant/)
(https://www.baeldung.com/author/kumar-chandrakant/)

Primavera (https://www.baeldung.com/category/spring/) +

Spring Core Basics (https://www.baeldung.com/tag/spring-core-basics/)

Si tiene algunos años de experiencia en **Linux** y está interesado en compartirlo con la comunidad (y recibir un pago por su trabajo, por supuesto), eche un vistazo a la página "Escribir para nosotros" (/contribution-guidelines) . Aclamaciones. Eugen

1. Información general

En este artículo, veremos la propuesta de valor principal de Spring (https://spring.io/) como uno de los frameworks Java más populares.

Más importante aún, trataremos de entender las razones por las cuales Spring es nuestro marco de elección. Los detalles de Spring y sus componentes han sido ampliamente cubiertos en nuestros tutoriales anteriores (https://www.baeldung.com/spring-intro) . Por lo tanto, omitiremos las partes introductorias de "cómo" y nos centraremos principalmente en "por qué".

2. ¿Por qué usar cualquier marco?

Antes de comenzar cualquier discusión en particular sobre la primavera, primero comprendamos por qué necesitamos usar cualquier marco en primer lugar.

Un **lenguaje de programación de propósito general como Java es capaz de soportar una amplia variedad de aplicaciones** . Sin mencionar que Java se está trabajando y mejorando activamente todos los días.

Además, existen innumerables bibliotecas de código abierto y propietarias para admitir Java a este respecto.

Entonces, ¿por qué necesitamos un marco después de todo? Honestamente, no es absolutamente necesario usar un marco para realizar una tarea. Pero, a menudo es recomendable usar uno por varias razones:

- Nos ayuda a **centrarnos en la tarea central en lugar de la repetitiva** asociada a ella
- Reúne años de sabiduría en forma de patrones de diseño.
- Nos ayuda a cumplir con los estándares regulatorios y de la industria.
- Reduce el costo total de propiedad de la aplicación

Acabamos de arañar la superficie aquí y debemos decir que los beneficios son difíciles de ignorar. Pero no puede ser todo positivo, entonces, ¿cuál es el problema?

- Nos obliga a **escribir una solicitud de manera específica**
- Se une a una versión específica de lenguaje y bibliotecas.
- Se agrega a la huella de recursos de la aplicación

Francamente, no hay balas de plata en el desarrollo de software y los marcos ciertamente no son una excepción a eso. Por lo tanto, la elección de qué marco o ninguno debe basarse en el contexto.

Con suerte, estaremos en mejores condiciones para tomar esta decisión con respecto a Spring in Java al final de este artículo.

3. Breve descripción del ecosistema de primavera

Antes de comenzar nuestra evaluación cualitativa de Spring Framework, echemos un vistazo más de cerca a cómo se ve

Spring surg
rápidamente

so.

Spring comenzó como un contenedor de Inversión de Control (IoC) para Java (https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring) . Todavía relacionamos Spring principalmente con él y, de hecho, forma el núcleo del marco y otros proyectos que se han desarrollado sobre él.

3.1. Spring Framework

Spring Framework está dividido en módulos, lo (https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html) que hace que sea realmente fácil elegir y elegir partes para usar en cualquier aplicación:

- **Núcleo (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/core.html#spring-core)** : proporciona funciones básicas como DI (inyección de dependencia), internacionalización, validación y AOP (programación orientada a aspectos)
- **Acceso a datos (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/data-access.html#spring-data-tier)** : admite el acceso a datos a través de JTA (Java Transaction API), JPA (Java Persistence API) y JDBC (Java Database Connectivity)

- **Web (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/web.html#spring-web)** : admite tanto Servlet API (**Spring MVC (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/web.html#spring-web)**) como la recientemente reactiva API (**Spring WebFlux (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/web-reactive.html#spring-webflux)**), y además admite WebSockets, STOMP y WebClient
- **Integración (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/integration.html#spring-integration)** : admite la integración a Enterprise Java a través de JMS (Java Message Service), JMX (Java Management Extension) y RMI (Remote Method Invocation)
- **Pruebas (https://docs.spring.io/spring/docs/5.1.8.RELEASE/spring-framework-reference/testing.html#testing)** : amplio soporte para pruebas de unidad e integración a través de objetos simulados, accesorios de prueba, gestión de contexto y almacenamiento en caché

3.2. Proyectos de primavera

Pero lo que hace que Spring sea mucho más valiosa es **un ecosistema fuerte que ha crecido a su alrededor a lo largo de los años y que continúa evolucionando activamente** . Estos se estructuran como proyectos de Spring (https://spring.io/projects) que se desarrollan sobre el marco de Spring.

Aunque la lista de proyectos de Spring es larga y sigue cambiando, hay algunos que vale la pena mencionar:

- **Arranque (https://www.baeldung.com/spring-boot)** : nos proporciona un conjunto de plantillas altamente obstinadas pero extensibles para crear varios proyectos basados en Spring en muy poco tiempo. Hace que sea muy fácil crear aplicaciones Spring independientes con Tomcat incorporado o un contenedor similar.
- **Nube (https://www.baeldung.com/spring-cloud-tutorial)** : Proporciona soporte para desarrollar fácilmente algunos de los patrones comunes del sistema distribuido como descubrimiento de servicios, disyunt nes repetiti
- **Seguri** o para desarrollar autenticación y autorización para proyectos basados en Spring de una manera altamente personalizable. Con un soporte declarativo mínimo, obtenemos protección contra ataques comunes como la fijación de sesiones, clics y falsificación de solicitudes entre sitios.
- **Móvil (https://www.baeldung.com/spring-mobile)** : proporciona capacidades para detectar el dispositivo y adaptar el comportamiento de la aplicación en consecuencia. Además, admite la administración de vista compatible con el dispositivo para una experiencia de usuario óptima, administración de preferencias del sitio y cambio de sitio.
- **Lote (https://www.baeldung.com/introduction-to-spring-batch)** : Proporciona un marco ligero para desarrollar aplicaciones por lotes para sistemas empresariales como el archivo de datos. Tiene soporte intuitivo para programar, reiniciar, omitir, recopilar métricas e iniciar sesión. Además, admite la ampliación para trabajos de gran volumen a través de la optimización y la partición.

No hace falta decir que esta es una introducción bastante abstracta a lo que Spring tiene para ofrecer. Pero nos proporciona suficiente terreno con respecto a la organización y amplitud de Spring para llevar nuestra discusión más allá.

4. Primavera en acción

Es costumbre agregar un programa hello-world para comprender cualquier tecnología nueva.

Veamos cómo **Spring puede convertirlo en un juego de niños para escribir un programa que haga más que solo hello-world** . Crearemos una aplicación que expondrá las operaciones CRUD como API REST para una entidad de dominio como Empleado respaldada por una base de datos en memoria. Además, protegeremos nuestros puntos finales de mutación utilizando la autenticación básica. Finalmente, ninguna aplicación puede estar realmente completa sin buenas y viejas pruebas unitarias.

4.1. Configuración del proyecto

Configuraremos nuestro proyecto Spring Boot usando Spring Initializr (https://start.spring.io/) , que es una herramienta en línea conveniente para arrancar proyectos con las dependencias correctas. Agregaremos Web, JPA, H2 y Seguridad como dependencias del proyecto para obtener la configuración de configuración de Maven correctamente.

Más detalles sobre bootstrapping (https://www.baeldung.com/spring-boot-start) están disponibles en uno de nuestros artículos anteriores.

4.2. Modelo de dominio y persistencia

Con tan poco por hacer, ya estamos listos para definir nuestro modelo de dominio y persistencia.

Primero definamos al *Empleado* como una simple entidad JPA:

```
@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @NotNull
    private String firstName;
    @NotNull
    private String lastName;
    // Standard constructor, getters and setters
}
```

Tenga en cuenta la identificación autogenerada que hemos incluido en nuestra definición de entidad.

Ahora tenemos que definir un repositorio JPA para nuestra entidad. Aquí es donde Spring lo hace realmente simple:

```
public interface EmployeeRepository
  extends CrudRepository<Employee, Long> {
    List<Employee> findAll();
}
```

Todo lo que tenemos que hacer es definir una interfaz como esta, y **Spring JPA nos proporcionará una implementación desarrollada con operaciones predeterminadas y personalizadas** . ¡Muy ordenado! Encuentre más detalles sobre cómo trabajar con Spring Data JPA (https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa) en nuestros otros artículos.

4.3. Controlador

Ahora tenemos que definir un controlador web para enrutar y manejar nuestras solicitudes entrantes:

```
@RestController
public class EmployeeController {
    @Autowired
    private EmployeeRepository repository;
    @GetMapping("/employees")
    public List<Employee> getEmployees() {
        return repository.findAll();
    }
    // Other CRUD endpoints handlers
}
```

Realmente, todo lo que teníamos que hacer era **anotar la clase y definir la metainformación de enrutamiento** junto con cada método de controlador.

Trabajar con controladores Spring REST (https://www.baeldung.com/building-a-restful-web-service-with-spring-and-java-based-configuration) está cubierto con gran detalle en nuestro artículo anterior.

4.4. Seguridad

Así que hemos definido todo ahora, pero ¿qué hay de asegurar operaciones como crear o eliminar empleados? ¡No queremos acceso no autenticado a esos puntos finales!

Spring Security realmente brilla en esta área:

```
@EnableWebSecurity
public class WebSecurityConfig
  extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http)
      throws Exception {
        http
          .authorizeRequests()
            .antMatchers(HttpMethod.GET, "/employees", "/employees/**")
            .permitAll()
          .anyRequest()
            .authenticated()
          .and()
            .httpBasic();
    }
    // other necessary beans and definitions
}
```

Aquí hay más detalles que requieren atención (https://www.baeldung.com/spring-security-basic-authentication) para comprender, pero el punto más importante a tener en cuenta es **la manera declarativa en la que solo hemos permitido operaciones GET sin restricciones** .

4.5. Pruebas

Ahora hemos hecho todo, pero espera, ¿cómo probamos esto?

Veamos si Spring puede facilitar la escritura de pruebas unitarias para controladores REST:

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@AutoConfigureMockMvc
public class EmployeeControllerTests {
    @Autowired
    private MockMvc mvc;
    @Test
    @WithMockUser()
    public void givenNoEmployee_whenCreateEmployee_thenEmployeeCreated() throws Exception {
        mvc.perform(post("/employees").content(
            new ObjectMapper().writeValueAsString(new Employee("First", "Last"))
            .with(csrf()))
          .contentType(MediaType.APPLICATION_JSON)
          .accept(MediaType.APPLICATION_JSON))
          .andExpect(MockMvcResultMatchers.status()
            .isCreated())
          .andExpect(jsonPath("$.firstName", is("First")))
          .andExpect(jsonPath("$.lastName", is("Last")));
    }
    // other tests as necessary
}
```

Como podemos ver, **Spring nos proporciona la infraestructura necesaria para escribir pruebas simples de unidad e integración** que, de lo contrario, dependen del contexto de Spring que se inicializará y configurará.

4.6. Ejecutando la aplicación

Finalmente, ¿cómo ejecutamos esta aplicación? Este es otro aspecto interesante de Spring Boot. Aunque podemos empaquetar esto como una aplicación normal e implementarlo tradicionalmente en un contenedor de Servlet.

Pero donde es divertido esto que! **Spring Boot viene con un servidor Tomcat incorporado** :

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Esta es una clase que viene pre-creada como parte del bootstrap y tiene todos los detalles necesarios para iniciar esta aplicación usando el servidor incorporado.

Además, esto es altamente personalizable (https://www.baeldung.com/spring-boot-application-configuration) .

5. Alternativas a la primavera

Si bien elegir usar un marco es relativamente más fácil, elegir entre marcos a menudo puede ser desalentador con las opciones que tenemos. Pero para eso, debemos tener al menos una comprensión aproximada de qué alternativas hay para las características que Spring tiene para ofrecer.

Como discutimos anteriormente, **el marco Spring junto con sus proyectos ofrecen una amplia variedad para que un desarrollador empresarial elija** . Si hacemos una evaluación rápida de los marcos de trabajo contemporáneos de Java, ni siquiera se acercan al ecosistema que nos proporciona Spring.

Sin embargo, para áreas específicas, forman un argumento convincente para elegir como alternativas:

- **Guice (https://www.baeldung.com/guice)** : ofrece un contenedor de IoC robusto para aplicaciones Java
- **Reproducir (https://www.baeldung.com/java-intro-to-the-play-framework)** : encaja perfectamente como un marco web con soporte reactivo
- **Hibernate (https://www.baeldung.com/hibernate-4-spring)** : un marco establecido para el acceso a datos con soporte JPA

Aparte de estos, hay algunas adiciones recientes que ofrecen un soporte más amplio que un dominio específico, pero aún no cubren todo lo que Spring tiene para ofrecer:

- **Micronaut (https://www.baeldung.com/micronaut)** : un marco basado en JVM diseñado para microservicios nativos de la nube
- **Quarkus (https://www.baeldung.com/quarkus-io)** : una nueva pila de Java que promete ofrecer un tiempo de arranque más rápido y una huella más pequeña

Obviamente, no es necesario ni factible recorrer la lista por completo, pero aquí tenemos la idea general.

6. Entonces, ¿por qué elegir la primavera?

Finalmente, hemos construido todo el contexto requerido para abordar nuestra pregunta central, ¿por qué Spring? Entendemos las formas en que un marco puede ayudarnos a desarrollar aplicaciones empresariales complejas.

Además, entendemos las opciones que tenemos para preocupaciones específicas como web, acceso a datos, integración en términos de framework, especialmente para Java.

Ahora, ¿dónde brilla la primavera entre todos estos? Vamos a explorar.

6.1. Usabilidad

Uno de los aspectos clave de la popularidad de cualquier framework es lo fácil que es para los desarrolladores usarlo. Salta a través de múltiples opciones de configuración y Convención sobre Configuración hace que sea **realmente fácil para los desarrolladores comenzar y luego configurar exactamente lo que necesitan** .

Proyectos como **Spring Boot han hecho que el arranque de un complejo proyecto Spring sea casi trivial** . Sin mencionar que tiene excelente documentación y tutoriales para ayudar a cualquiera a subir a bordo.

6.2. Modularidad

Otro aspecto clave de la popularidad de Spring es su naturaleza altamente modular. Tenemos opciones para usar todo el framework Spring o solo los módulos necesarios. Además, **opcionalmente** podemos **incluir uno o más proyectos Spring** dependiendo de la necesidad.

¡Además, tenemos la opción de usar otros marcos como Hibernate o Struts también!

6.3. Conformidad

Aunque Spring **no admite todas las especificaciones de Java EE, admite todas sus tecnologías** , a menudo mejorando el soporte sobre la especificación estándar cuando sea necesario. Por ejemplo, Spring admite repositorios basados en JPA y, por lo tanto, es trivial cambiar de proveedor.

Además, Spring admite especificaciones de la industria como Reactive Stream (https://www.reactive-streams.org/) en Spring Web Reactive y HATEOAS en Spring HATEOAS (https://www.baeldung.com/spring-hateoas-tutorial) .

6.4. Testabilidad

La adopción de cualquier marco también depende en gran medida del hecho de lo fácil que es probar la aplicación construida sobre él. Spring at the core **aboga y apoya el desarrollo impulsado por pruebas** (TDD).

La aplicación Spring se compone principalmente de POJO, lo que naturalmente hace que las pruebas unitarias sean relativamente mucho más simples. Sin embargo, Spring proporciona Mock Objects para escenarios como MVC donde las pruebas unitarias se complican de lo contrario.

6.5 Madurez

Spring tiene una larga historia de innovación, adopción y estandarización. Con los años, se ha vuelto lo **suficientemente maduro como para convertirse en una solución predeterminada para los problemas más comunes que se** enfrentan en el desarrollo de aplicaciones empresariales a gran escala.

Lo que es aún más emocionante es cuán activamente se está desarrollando y manteniendo. El soporte para nuevas funciones de lenguaje y soluciones de integración empresarial se están desarrollando todos los días.

6.6. Soporte comunitario

Por último, pero no menos importante, cualquier marco o incluso biblioteca sobrevive a la industria a través de la innovación y no hay mejor lugar para la innovación que la comunidad. Spring es un código abierto **liderado por Pivotal Software y respaldado por un gran consorcio de organizaciones y desarrolladores individuales** .

Esto ha significado que sigue siendo contextual y a menudo futurista, como lo demuestra la cantidad de proyectos bajo su paraguas.

7. Razones para *no* usar Spring

Existe una amplia variedad de aplicaciones que pueden beneficiarse de un nivel diferente de uso de Spring, y eso está cambiando tan rápido como Spring está creciendo.

Sin embargo, debemos entender que Spring, como cualquier otro marco, es útil para administrar la complejidad del desarrollo de aplicaciones. Nos ayuda a evitar dificultades comunes y mantiene la aplicación mantenible a medida que crece con el tiempo.

Esto **tiene el costo de una huella de recursos adicional y una curva de aprendizaje** , por pequeña que sea. Si realmente hay una aplicación que es lo suficientemente simple y no se espera que se vuelva compleja, ¡tal vez sea más beneficioso no utilizar ningún marco en absoluto!

8. Conclusión

En este artículo, discutimos los beneficios de usar un marco en el desarrollo de aplicaciones. Discutimos más brevemente Spring Framework en particular.

Mientras hablamos del tema, también analizamos algunos de los marcos alternativos disponibles para Java.

Finalmente, discutimos las razones que pueden obligarnos a elegir Spring como el marco de elección para Java.

Sin embargo, deberíamos terminar este artículo con una nota de consejo. Por más convincente que parezca, **generalmente no existe una solución única y única** para el desarrollo de software.

Por lo tanto, debemos aplicar nuestra sabiduría al seleccionar las soluciones más simples para los problemas específicos que buscamos resolver.

¡Los comentarios están cerrados en este artículo!

CATEGORÍAS

PRIMAVERA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
HTTP DEL LADO DEL CLIENTE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL DE JAVA "VOLVER A LO BÁSICO" (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
RESTO CON SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
TUTORIAL SPRING PERSISTENCE (/PERSISTENCE-WITH-SPRING-SERIES)
SEGURIDAD CON PRIMAVERA (/SECURITY-SPRING)