Spring Framework (http://github.com/spring-projects/spring-framework)

Core support for dependency injection, transaction management, web applications, data access, messaging, testing and more.

QUICK START

Introduction

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of

enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Features

- Dependency Injection
- Aspect-Oriented Programming including Spring's declarative transaction management
- Spring MVC web application and RESTful web service framework
- Foundational support for JDBC, JPA, JMS
- Much more...

All avaible features and modules are described in the Modules section of the reference documentation (http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/overview.html#overview-modules). Their maven/gradle coordinates are also described there (http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/overview.html#dependency-management).

Minimum requirements

- JDK 6+ for Spring Framework 4.x
- JDK 5+ for Spring Framework 3.x

Quick Start

Download 4.1.5 ≡ MAVEN GRADLE

The recommended way to get started using `spring-framework` in your project is with a dependency management system – the snippet below can be copied and pasted into your build. Need help? See our getting started guides on building with Maven (http://spring.io/guides/gs/maven/) and Gradle (http://spring.io/guides/gs/gradle/).

RELEASE	DOCUMENTATION
4.2.0	Reference (http://docs.spring.io/spring/docs/4.2.0.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/) API (http://docs.spring.io/spring/docs/4.2.0.BUILD-SNAPSHOT/javadoc-api/)
4.1.5	Reference (http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/) API (http://docs.spring.io/spring/docs/current/javadoc-api/)

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.1.5.RELEASE</version>
    </dependency>
</dependencies>
```

Spring Framework includes a number of different modules. Here we are showing `spring-context` which provides core functionality. Refer to the getting started guides on the right for other options.

Once you've set up your build with the `spring-context` dependency, you'll be able to do the following:

`hello/MessageService.java`

```
package hello;

public interface MessageService {
    String getMessage();
}
```

`hello/MessagePrinter.java`

4.0.9	Reference (http://docs.spring.io/spring/docs/4.0.9.RELEASE/spring-framework-reference/htmlsingle/) API (http://docs.spring.io/spring/docs/4.0.9.RELEASE/javadoc-api/)
3.2.13	Reference (http://docs.spring.io/spring/docs/3.2.13.RELEASE/spring-framework-reference/htmlsingle/) API (http://docs.spring.io/spring/docs/3.2.13.RELEASE/javadoc-api/)

(https://twitter.com/SpringFramework)
(https://github.com/spring-projects/spring-framework)
(http://jira.springsource.org/browse/SPR)
(https://build.springsource.org/browse/SPR)
(http://stackoverflow.com/questions/tagged/spring)

Getting Started Guides

```
package hello;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class MessagePrinter {

    final private MessageService service;

    @Autowired
    public MessagePrinter(MessageService service) {
        this.service = service;
    }

    public void printMessage() {
        System.out.println(this.service.getMessage());
    }
}
```

hello/Application.java

Building a RESTful Web Service (http://spring.io/guides/gs/rest-service)

Consuming a RESTful Web Service (http://spring.io/guides/gs/consuming-rest)

Managing Transactions (http://spring.io/guides/gs/managing-transactions)

Accessing Relational Data using JDBC with Spring (http://spring.io/guides/gs/relational-data-access)

Scheduling Tasks (http://spring.io/guides/gs/scheduling-tasks)

Serving Web Content (http://spring.io/guides/gs/serving-web-content)

Validating Form Input (http://spring.io/guides/gs/validating-form-input)

Messaging with JMS (http://spring.io/guides/gs/messaging-jms)

Tutorials

Designing and Implementing RESTful Web Services with Spring (http://spring.io/guides/tutorials/rest)

Screencasts

How to create a RESTful app in five minutes or less (http://spring.io/blog/2014/11/20/screencast-how-to-create-a-restful-app-in-five-minutes-or-less)

```
package hello;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.*;

@Configuration
@ComponentScan
public class Application {

    @Bean
    MessageService mockMessageService() {
        return new MessageService() {
            public String getMessage() {
              return "Hello World!";
            }
        };
    }

  public static void main(String[] args) {
      ApplicationContext context =
          new AnnotationConfigApplicationContext(Application.class);
      MessagePrinter printer = context.getBean(MessagePrinter.class);
      printer.printMessage();
  }
}
```

The example above shows the basic concept of dependency injection (http://stackoverflow.com/questions/24337486/how-to-properly-do-dependency-injection-in-spring/24363707#24363707), the `MessagePrinter` is decoupled from the `MessageService` implementation, with Spring Framework wiring everything together.

- Email Address
- SUBSCRIE