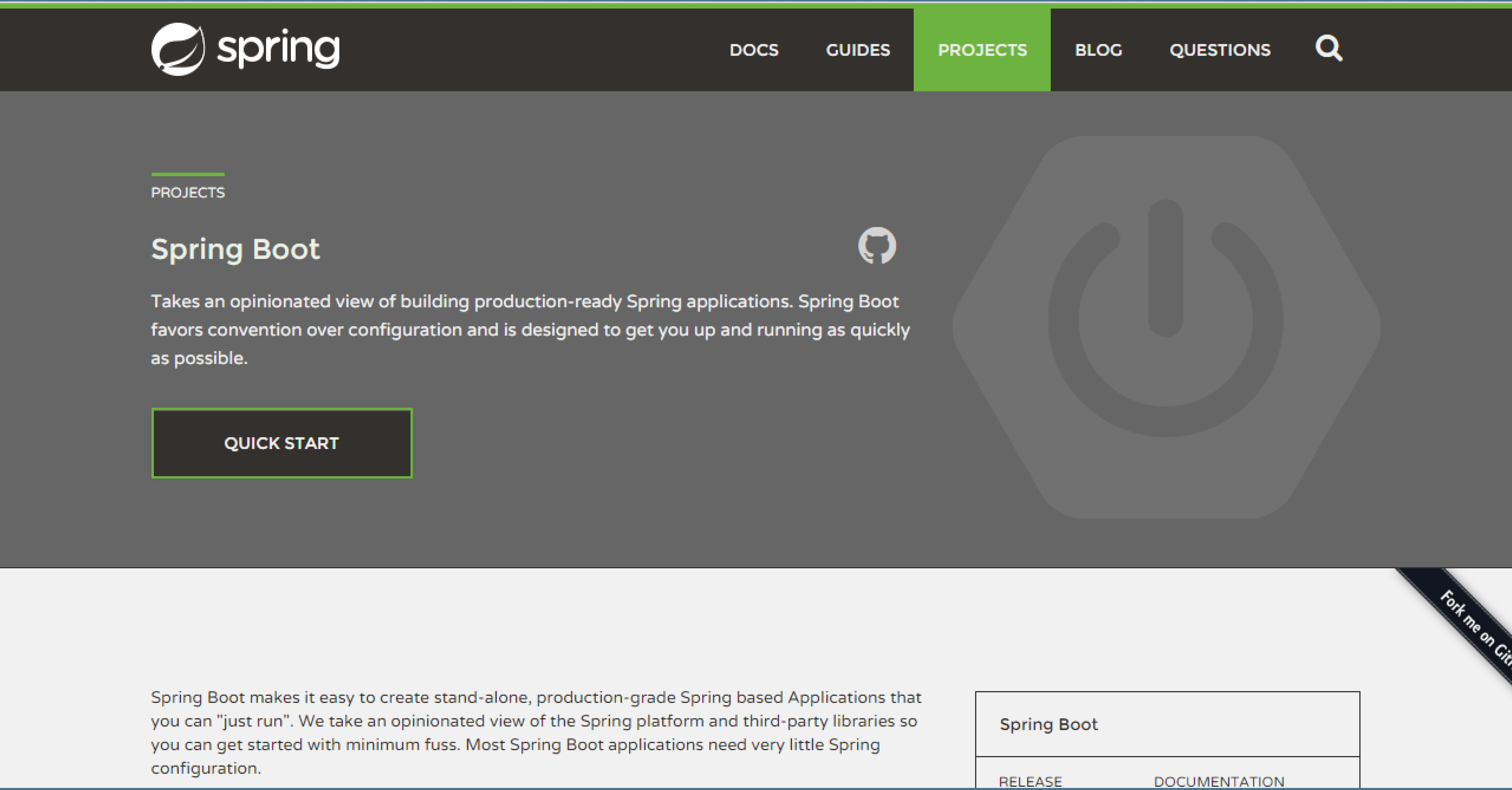
JAVA – Dia 05

Objetivos

- Spring Boot
- Thymeleaf vs JSP
- Spring MVC
  - @Controller e @RequestMapping
  - Model e ModelAndView
- JPA
  - Hibernate
  - Mapeamento Simples



PROJECTS

Spring Boot

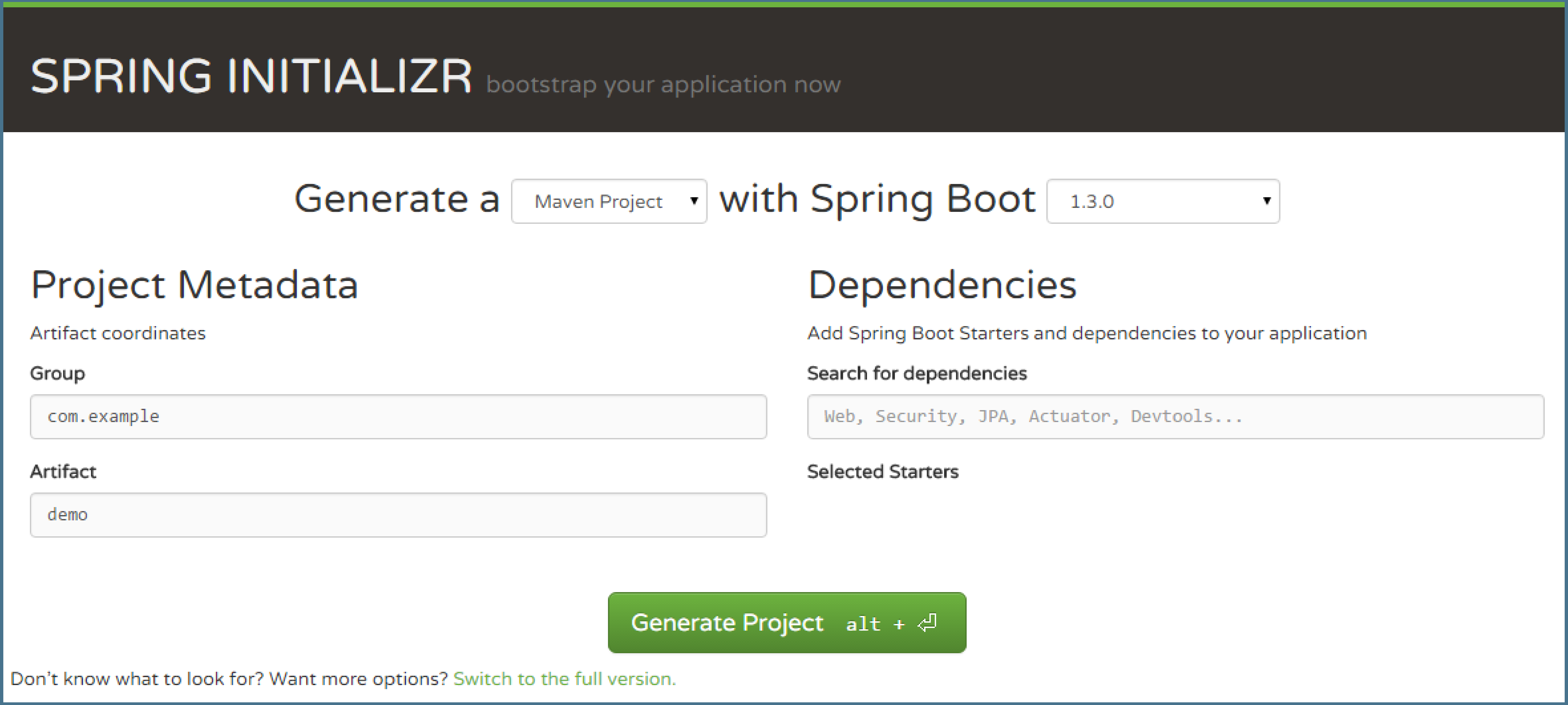
Takes an opinionated view of building production-ready Spring applications. Spring Boot favors convention over configuration and is designed to get you up and running as quickly as possible.

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Spring Boot

RELEASE	DOCUMENTATION

http://start.spring.io



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Spring Boot 1.3.0

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

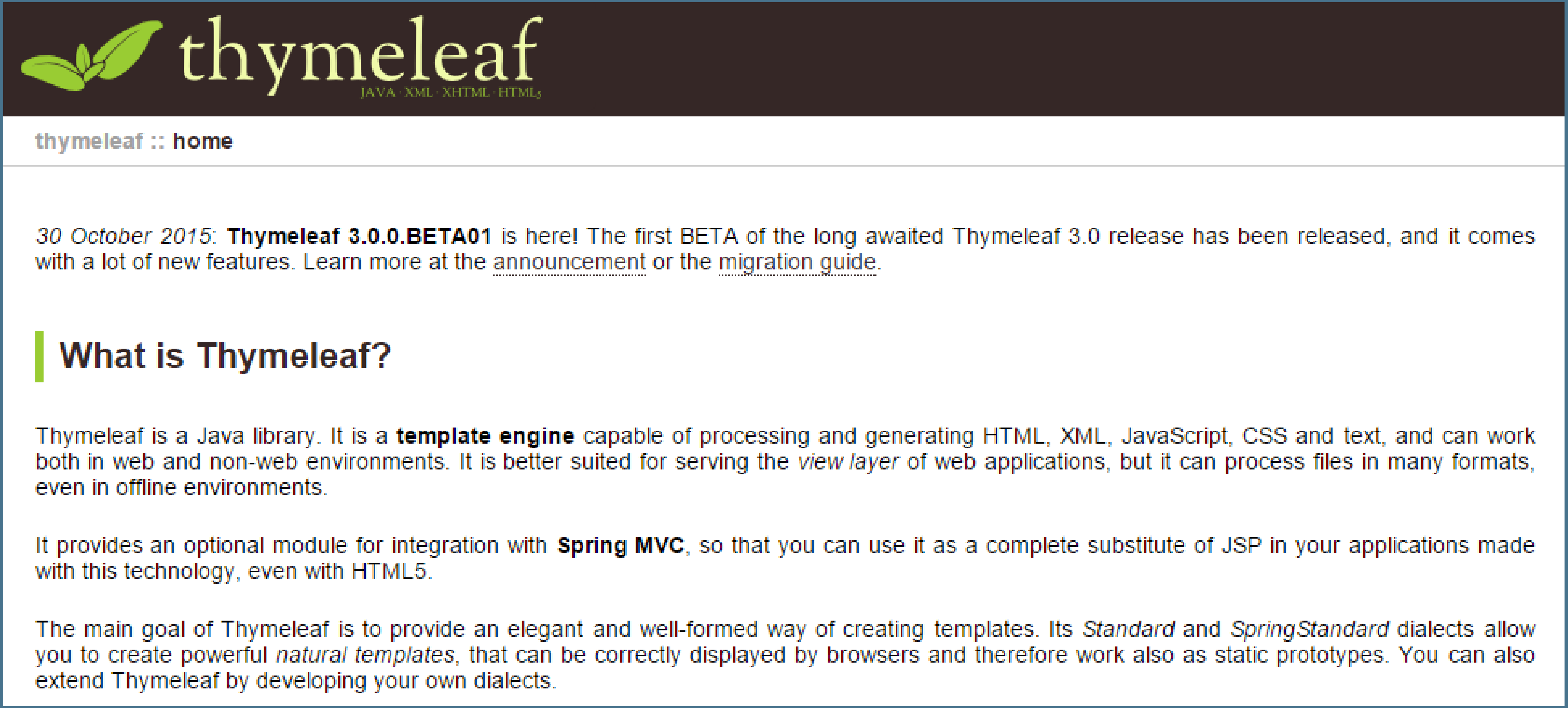
Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Starters

Generate Project alt + ⏎

Don't know what to look for? Want more options? Switch to the full version.



thymeleaf
JAVA · XML · XHTML · HTML5

thymeleaf :: home

30 October 2015: **Thymeleaf 3.0.0.BETA01** is here! The first BETA of the long awaited Thymeleaf 3.0 release has been released, and it comes with a lot of new features. Learn more at the announcement or the migration guide.

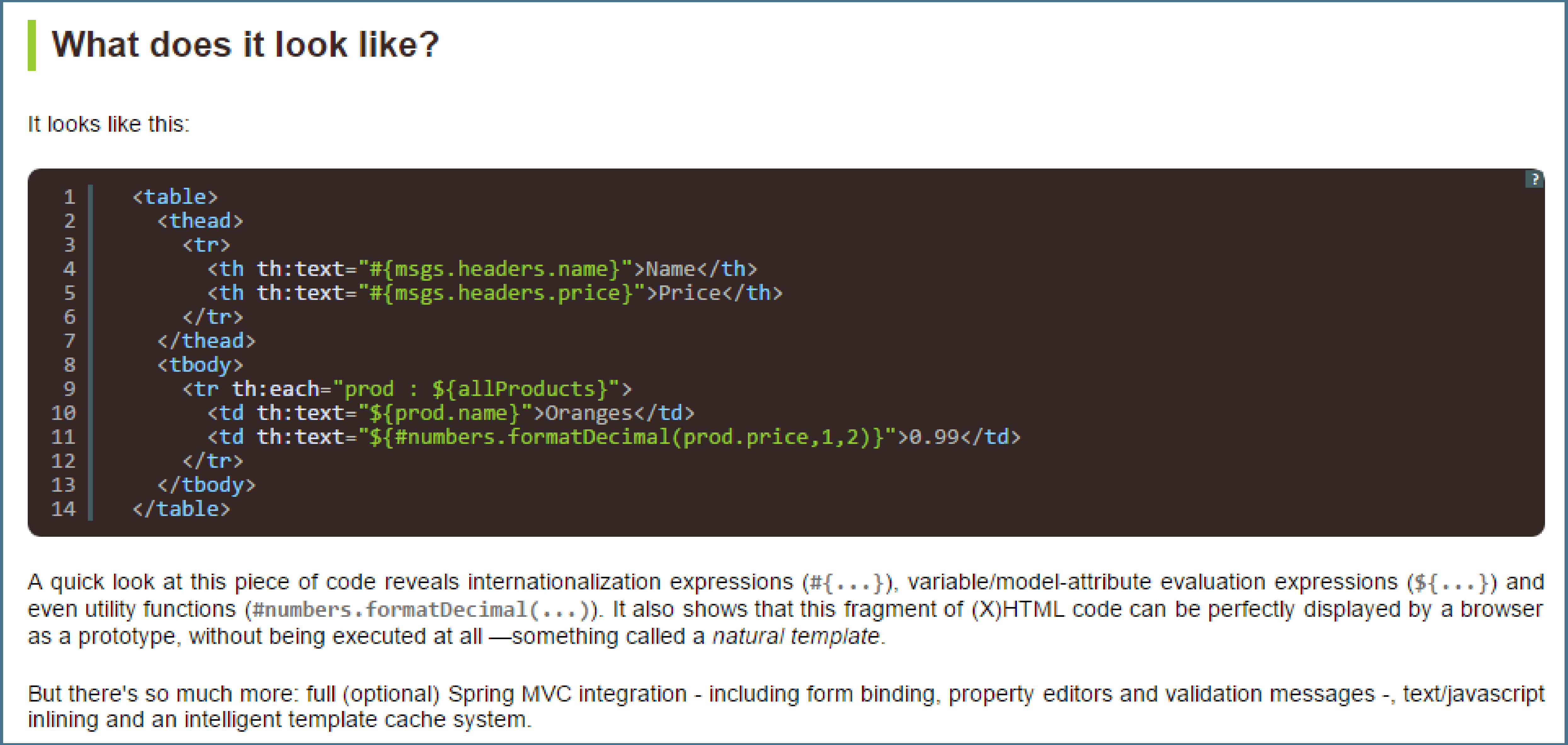
What is Thymeleaf?

Thymeleaf is a Java library. It is a **template engine** capable of processing and generating HTML, XML, JavaScript, CSS and text, and can work both in web and non-web environments. It is better suited for serving the *view layer* of web applications, but it can process files in many formats, even in offline environments.

It provides an optional module for integration with **Spring MVC**, so that you can use it as a complete substitute of JSP in your applications made with this technology, even with HTML5.

The main goal of Thymeleaf is to provide an elegant and well-formed way of creating templates. Its *Standard* and *SpringStandard* dialects allow you to create powerful *natural templates*, that can be correctly displayed by browsers and therefore work also as static prototypes. You can also extend Thymeleaf by developing your own dialects.

http://www.thymeleaf.org/doc/articles/standarddialect5minutes.html



What does it look like?

It looks like this:

```
<table>
  <thead>
    <tr>
      <th th:text="#{msgs.headers.name}">Name</th>
      <th th:text="#{msgs.headers.price}">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="prod : ${allProducts}">
      <td th:text="${prod.name}">Oranges</td>
      <td th:text="${#numbers.formatDecimal(prod.price,1,2)}">0.99</td>
    </tr>
  </tbody>
</table>
```

A quick look at this piece of code reveals internationalization expressions (`#{...}`), variable/model-attribute evaluation expressions (`${...}`) and even utility functions (`#numbers.formatDecimal(...)`). It also shows that this fragment of (X)HTML code can be perfectly displayed by a browser as a prototype, without being executed at all —something called a *natural template*.

But there's so much more: full (optional) Spring MVC integration - including form binding, property editors and validation messages -, text/javascript inlining and an intelligent template cache system.

Thymeleaf vs JSP

http://www.thymeleaf.org/doc/articles/thvsjsp.html

http://www.thymeleaf.org/features.html

http://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html

http://www.thymeleaf.org/documentation.html

http://www.thymeleaf.org/whoisusingthymeleaf.html

http://www.thymeleaf.org/ecosystem.html