Introdução ao Spring Framework

Veja nesse artigo uma introdução ao Spring Framework, um dos mais utilizados em projetos na linguagem Java. Conheça um pouco sobre o framework e implemente um exemplo de Hello World.

Muitos já ouviram falar nesse framework(Spring), mas a complexidade inicial afasta ou assusta bastante os iniciantes do mundo Java. Sim, a principio e principalmente para um iniciante o Spring pode parecer nada simples, mas para desenvolvedores mais experientes, logo pode-se perceber o encanto e a mágica do Spring Framework. Nesse artigo vamos desvendar um pouco desse completíssimo framework.

Configurando o Spring

Para esse artigo vamos utilizar o Eclipse e o Tomcat 7, certifique-se de tê-los instalados antes de continuar.

Primeiramente criaremos um “Dynamic Web Project” com o nome de “hello-spring”, como mostra a figura 1.

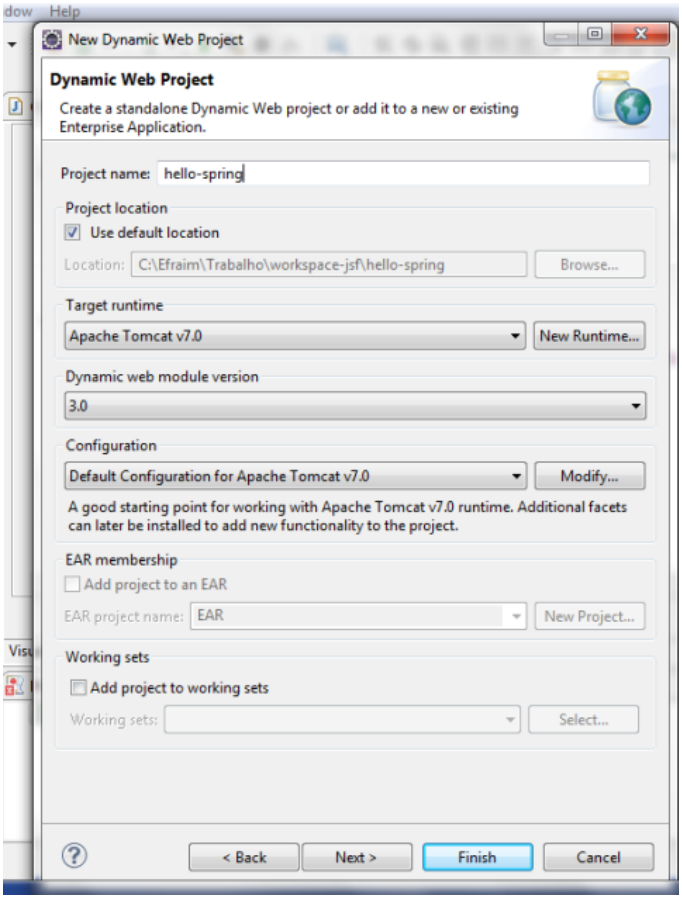

Figura 1: Criando o projeto

Agora vem a parte mais “complicada”, configura o Spring, a princípio algo bastante complexo para aqueles que ainda não possuem muita pratica, lembrando que todo framework é um bicho de sete cabeças a primeira vista.

Para o Spring funcionar, vamos precisar de suas libs, acessando os links abaixo, vamos encontrar tudo o que precisamos.

- http://www.springsource.org/spring-framework
- http://commons.apache.org/logging/download_logging.cgi

No canto direito da tela teremos as últimas versões lançadas, usaremos a mais

libs do framew o se assuste, não usaremos todos esses jar's , vamos copiar "spring-webmvc-X-X.jar" , "spring-web- X-X.jar", "spring-expression-X-X.jar", "spring-core- X-X.jar", "spring-context- X-X.jar", "spring-beans- X-X.jar" e colar na pasta "WEB-INF/lib". Também é necessária a biblioteca "commons-logging-1.1.1.jar" para nosso projeto, ver figura 2.

Observação: X-X trata-se da versão do framework que foi baixada.

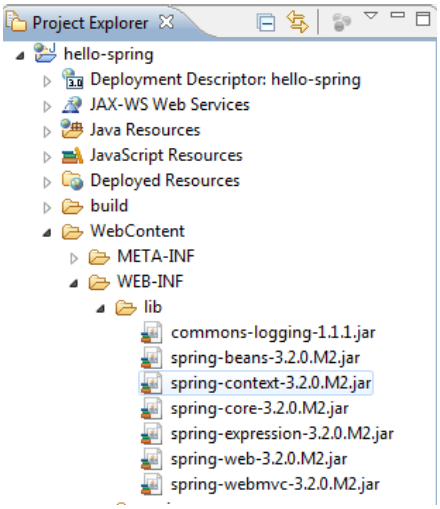

Figura 2: Monstrando JAR's do prejeto

Com posse da biblioteca do framework, agora vamos criar uma pasta chamada "spring" dentro do diretório WEB-INF, e dentro da pasta criaremos um arquivo xml chamado "application-context.xml", lembrando que o Spring é um framework "container-based", ou seja, ele vai conter e carregar o que você informá-lo. A raiz do nosso xml é tag <beans></beans> e dentro conterá toda a configuração do Spring

Observação: não é obrigatória a configuração do Spring em apenas um ".xml", é possível separar vários arquivos de configuração, exemplo: persistence-context.xml mvc-context.xml etc... , porém nesse artigo isso não será abordado.

Listagem 1: Como deve ficar nosso "application-context.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xsi:schemaLocation="http://www.springframework.org/schema/mvc http:/
                http://www.springframework.org/schema/beans http://www.sprin
                http://www.springframework.org/schema/context http://www.spr

        <!-- Informa o pacote onde o Spring ira buscar as classes anotadas (
        <context:component-scan base-package="br.com.devmedia" />
```

```
        <!-- Define pagina inicial (ingnora a configuração do web.xml)-->
        <mvc:view-controller path="/" view-name="helloworld"/>

        <!-- Define onde está localizada as views da aplicação, e qual a ext
        <!--
         Estão configuradas dentro da WEB-INF para que o usuário não possa a
         não por meio do mapeamento
         -->
        <bean class="org.springframework.web.servlet.view.InternalResourceVi
                <property name="prefix" value="/WEB-INF/views/"/>
                <property name="suffix" value=".jsp"/>
        </bean>

</beans>
```

Arquivo de configuração bem simples não? Claro, é uma aplicação simples, e a medida que sua aplicação cresce, basta ir acrescentando módulos de acordo com sua necessidade, evitando assim um consumo gigantesco de memória com coisas que você nunca vai utilizar na sua aplicação.

Agora vamos dizer a nossa aplicação Web para carregar o Spring, mas como? O bom e velho arquivo “web.xml”, será necessária apenas a configuração do Servlet do Spring, nada sobrenatural , veja como deve ficar o nosso arquivo.

Listagem 2: Arquivo web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/
  <display-name>hello-spring</display-name>

        <!-- Configura o Spring Servlet -->
        <servlet>
                <servlet-name>Spring-Servlet</servlet-name>
                <servlet-class>org.springframework.web.servlet.DispatcherSer
                <init-param>
                        <param-name>contextConfigLocation</param-name>
                        <param-value><!-- Especifica que arquivo de configur
                                /WEB-INF/spring/application-context.xml
                        </param-value>
                </init-param>
                <load-on-startup>1</load-on-startup>
        </servlet>

        <servlet-mapping><!-- Mapeia o Servlet -->
                <servlet-name>Spring-Servlet</servlet-name>
                <url-pattern>/</url-pattern>
        </servlet-mapping>

</web-app>
```

Então é só isso? Podemos rodar a aplicação? Calma, ainda não existe nenhuma página, então criaremos uma dentro da pasta “WEB-INF/view” com o nome de "helloworld.jsp”. Veja que é o mesmo nome o qual configuramos como página inicial no “application-context.xml”.

Listagem 3: Definindo a página inicial

```
<!-- Define pagina inicial (ingnora a configuração do web.xml)-->
        <mvc:view-controller path="/" view-name="helloworld"/>
```

Nossa página ficará assim.

Listagem 4: Página helloword.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Dev Media Hello Spring</title>
</head>
<body>
        <p>Hello World Spring</p>
        <br/><br/>
        <a href="bemvindo">Ir para pagina bem vindo</a>
</body>
</html>
```

Agora sim, você já pode rodar o projeto no tomcat, veja o resultado na figura 3.

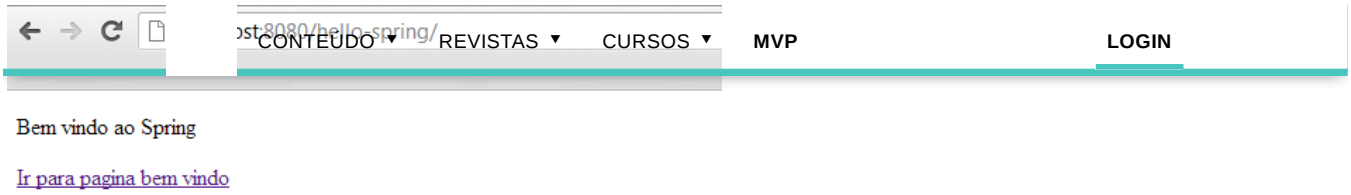

Figura 3: Exibindo um Hello World

Controllers

Exibir uma página é muito simples, mas qual a diferença do Spring? Vamos falar sobre o controle de navegação da aplicação. O Spring utiliza os chamados Controllers, que são classes mapeadas através de annotations que servem para dizer à aplicação o que exibir quando for requisitada uma página, ou envio de informações, uma espécie de Servlet do JSP, mas bem mais fácil de se trabalhar.

Vamos a um exemplo, criaremos uma classe chamada “HelloController” no pacote “br.com.devmedia.controllers”, se o pacote não existir, crie-o. Vamos anotar essa classe com o @Controller, que diz ao Spring que essa classe vai funcionar como uma espécie de Servlet para a aplicação, ou seja, vai receber requisições tratá-las e responder ao usuário.

Listagem 5: Anotação @Controller

```
@Controller
public class HelloController{
...
}
```

Espera ai, mas como minha aplicação vai saber qual requisição deve ser tratada pelo meu @Controller? Simples, através da anotação @RequestMapping, ela vai verificar qual url está sendo solicitada e enviar para o @Controller que contém a anotação com a url específica, veja.

Listagem 6: Método bemVindo

```
public class      ontroller {

        @RequestMapping("/bemvindo")
        public ModelAndView bemVindo(Model model){
                model.addAttribute("bemvindo" , "Olha só que facil dizer bem
                return new ModelAndView("bemvindo");
        }

}
```

Explicando o código, o @RequestMapping diz que quando for requisitada a url /bemvindo, será executado o método “bemVindo()” da classe “HelloController”. Mas o que é esse “Model” e de onde ele vem? Por que ele está ali? O Model não é um parâmetro obrigatório, e com um pouco mais de experiência verá que muita coisa não é. O “Model” vai servir para adicionar atributos para serem usados na tela, não se preocupe com detalhes, o Spring vai fazer isso por você. Devem ter notado também o “ModelAndView” que está sendo retornado, ele na verdade vai dizer ao Spring que página ele vai exibir, no caso a “bemvindo.jsp”, mas poderia ser qualquer página que você tiver dentro da pasta “WEB-INF/views” que é o diretório que foi mapeado para conter nossas “views”.

Listagem 7: Mapeamento do diretório das views

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolv
                <property name="prefix" value="/WEB-INF/views/"/>
                <property name="suffix" value=".jsp"/>
        </bean>
```

Agora que temos o @Controller que vai tratar uma requisição, precisamos também da página que vai ser exibida, no caso a “bemvindo.jsp”. Criaremos ela dentro de “WEB-INF/views”, e nessa página exibiremos o atributo passado pelo nosso @Controller, veja o código da JSP.

Listagem 8: Código da página jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Dev Media Hello Spring</title>
</head>
<body>
        <p>Pagina de boas vindas que tal ?</p>
        <br/>

        <p><%= request.getAttribute("bemvindo") %></p>
```

Rode novamente a aplicação e veja o resultado, como na figura 4.

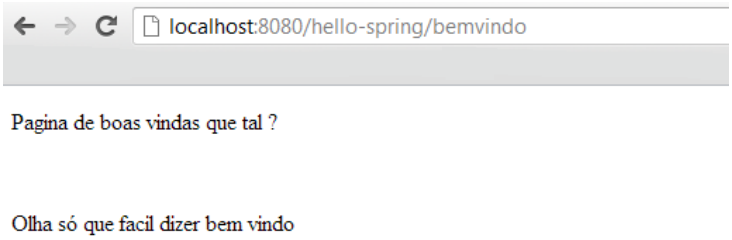

Figura 4: Apresentando pagina com atributo do @Controller

Concluindo

O que foi apresentado é apenas a ponta do iceberg, a maior dificuldade é justamente a configuração e o passo inicial, com essa dificuldade superada, agora é estudar e usar tudo o que esse incrível framework oferece.

Até a próxima.

Efraim Gentil

Estudante de Analise e Desenvolvimento de Sistemas, trabalha com desenvolvimento Web a mais de 2 anos com JAVA e outras linguagens.

[há +1 ano] - Responder

Sandro Rizzo Oliveira Almeida
Como faço para receber a continuação? Nem sempre vejo os post no twitter.

[há +1 ano] - Responder

[autor] Efraim Gentil De Goes Filho
Sandro, também existe o RSS da DevMedia, http://www.devmedia.com.br/feed/
ou ficar de olho no Facebook!

[há +1 ano] - Responder

Leonardo Mendonça
Muito bom, fica fácil começar.

[há +1 ano] - Responder

Flavio Pinto
muito bom, extremamente didático!!!

[há +1 ano] - Responder

Joel Medeiros
Muito bom... parabéns!

[há +1 ano] - Responder

Veodo Cara De Pau
E o que seria "ModelAndView" ????

[há +1 mês] - Responder

Douglas Claudio
Olá Veodo, obrigado pelo seu comentário.

Enviamos sua solicitação ao Efraim e estamos no aguardo de um feedback do mesmo.

Um abraço.

[há +1 mês] - Responder

[autor] Efraim Gentil
O ModelAndView é um facade do Spring juntado duas classes o Model e a View (também classes do Spring) , onde a groso modo o Model serão os atributos que a minha pagina terá acesso, e a View minha pagina em si, com essa facade, em uma unica instancia você pode trabalhar com essas duas informações, e como o controller retorna o ModelAndView, está ao mesmo tempo dizendo para qual pagina (View) e que atributos(Model) poderão ser acessados nessa view.

Não necessariamente é obrigado o uso do ModelAndView no retorno, caso você retorne uma String ele ira buscar a pagina parava você da mesma forma que faz com o ModelAndView, no exemplo não ficou claro por estar usando um Model separado, obrigado pela observação "veodo cara de pau".

[há +1 mês] - Responder

Estou tentando achar os jars para baixar no site do Spring conforme indicado mas parece ter mudado o site como eu faço?

[há +1 mês] - Responder

Douglas Claudio
Olá Aldecir, obrigado pelo seu comentário.

Enviamos sua solicitação ao Efraim e estamos no aguardo de um feedback do mesmo.

Um abraço.

[há +1 mês] - Responder

[autor] Efraim Gentil
Aldecir, opa verdade o site foi alterado a algum tempo, e foi retirada a opção de download, e sugerem o uso de um gerenciador de dependência como maven ou gradle, mas ainda é possível fazer o download pelo repositório deles, segue link "http://repo.spring.io", basta buscar pelo nome do jar e você vai ter a opção de download.

[há +1 mês] - Responder

Nélio De Souza Santos
Faltou informar que precisa da lib spring-aop-xxx
Abraço!

[há +1 mês] - Responder

[autor] Efraim Gentil
Não é necessária a dependência do spring-aop para rodar o exemplo do artigo

[há +1 mês] - Responder

Nélio De Souza Santos
Efraim, boa tarde!
Bom... o meu só rodou com esta lib :/

[há +1 mês] - Responder

[autor] Efraim Gentil
Nélio, qual versão do spring está usando?
fiz os testes aqui com a 3.2.0-RELEASE e está tudo ok.

[há +1 mês] - Responder

Mais posts

Pocket Video

Introdução às Certificações OCAJP e OCPJP

Testes do UserService - Curso de Java EE: Construa uma aplicação completa Java EE - Aula 125

Video aula

Testes do ServiceService - Curso de Java EE: Construa uma aplicação completa Java EE - Aula 124

Artigo

Arquitetura de Software: Avaliando a arquitetura de seus sistemas

Artigo

Elasticsearch: realizando buscas no Big Data

Artigo

Criando uma aplicação corporativa em Java – Parte 1

Artigo

Spring MVC: Construa aplicações responsivas com Bootstrap – Parte 1

Artigo

Spring Framework: Criando uma aplicação web – Parte 2

Artigo

Programação funcional com Java

Listar mais conteúdo