Spring Framework

Muitos já ouviram falar nesse framework (Spring), mas a complexidade inicial afasta ou assusta bastante os iniciantes do mundo Java. Sim, a princípio 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.

Spring é um framework de código aberto (open source), criado por Rod Johnson, em meados de 2002, e apresentado no seu livro Expert One-on-One: JEE Design and Development. Foi criado com o intuito simplificar a programação em Java, possibilitando construir aplicações que antes só era possível utilizando EJB's.

O Spring atualmente possui diversos módulos como Spring Data (trata da persistência), Spring Security (trata da segurança da aplicação) entre outros módulos. Mas o principal (core) pode ser utilizado em qualquer aplicação Java, as principais funcionalidades são a injeção de dependência (CDI) e a programação orientada a aspectos (AOP), cabe ao desenvolvedor dizer ao Spring o que quer usar. O que faz dele uma poderosa ferramenta, pois não existe a necessidade de se arrastar todas as ferramentas do framework para criar uma aplicação simples.

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.

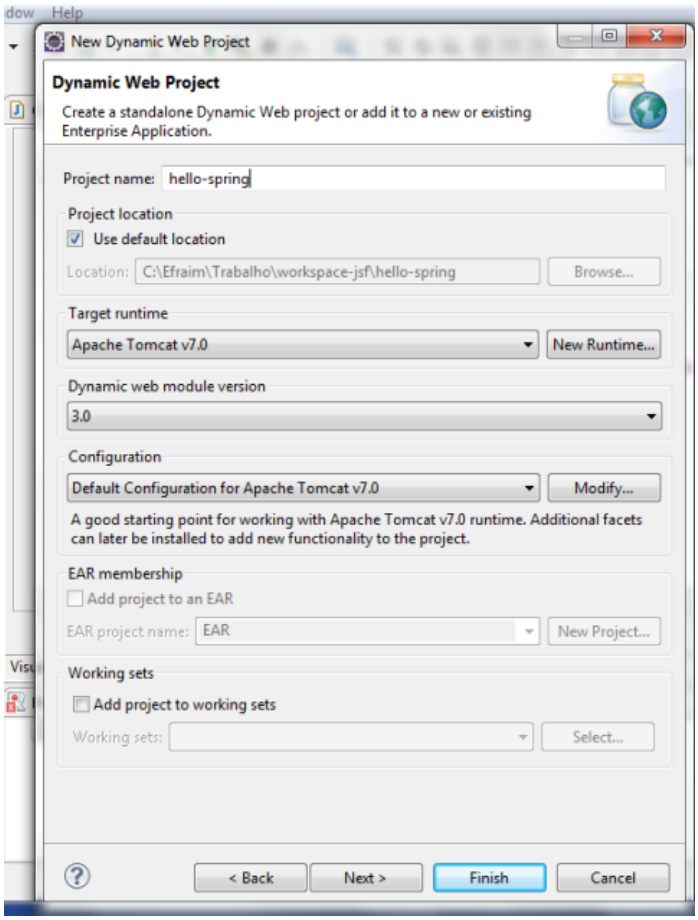

Figura 1: Criando o projeto

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.

No canto direito da tela teremos as últimas versões lançadas, usaremos a mais atual. Após baixar o zip do framework, vamos descomprimi-lo, e acessar a pasta libs do framework, não 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.

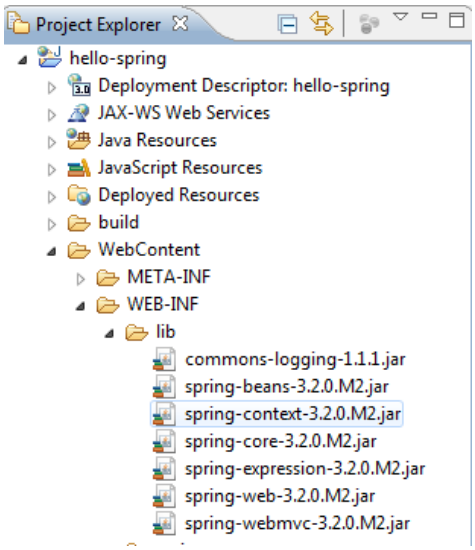

Figura 2: Monstrando JAR’s do prejeto

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://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd

                http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
```

```
                http://www.springframework.org/schema/context http://www.springframe-
work.org/schema/context/spring-context-3.0.xsd">
  <!-- Informa o pacote onde o Spring ira buscar as classes anotadas (@Controller, @Service...) -->
        <context:component-scan base-package="br.com.devmedia" />
        <!-- Diz ao Spring que ele deve usar a configuração das annotations -->
        <mvc:annotation-driven />
        <!-- 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 extenção das mesmas -->
        <!--
         Estão configuradas dentro da WEB-INF para que o usuário não possa acessalas, se
         não por meio do mapeamento
         -->
        <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <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://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/ja-
vaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

 <display-name>hello-spring</display-name>
        <!-- Configura o Spring Servlet -->
        <servlet>
                <servlet-name>Spring-Servlet</servlet-name>
                <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
                <init-param>
```

```
                    <param-name>contextConfigLocation</param-name>
                    <param-value><!-- Especifica que arquivo de configuração sera chamado, junto a instanciação -->
                        /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>
```

Exibindo uma pagina

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.w3.org/TR/html4/loose.dtd">
<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.

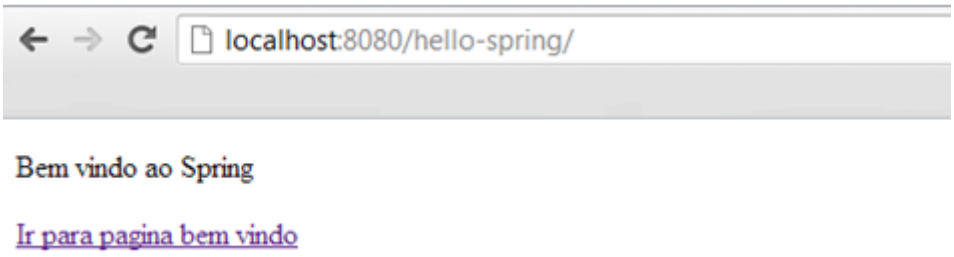

Figura 3: Exibindo um Hello World

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

```
@Controller
public class HelloController {
        @RequestMapping("/bemvindo")
        public ModelAndView bemVindo(Model model){
                model.addAttribute("bemvindo" , "Olha só que facil dizer bem vindo");
                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.InternalResourceViewResolver">
                        <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.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Dev Media Hello Spring</title>
</head>
<body>
```