**Aula XXVI - Desenvolvimento Web III**

**Introdução ao Spring - MVC**

**Arquitetura**

**Servlet**

**Arquivos Padrão do Spring**

**Arquivos de Usuário**

**JavaBeans - entendendo os “beans”**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Fontes:***

[*https://www.caelum.com.br/apostila-java-web/spring-mvc/#porque-precisamos-de-frameworks-mvc*](https://www.caelum.com.br/apostila-java-web/spring-mvc/#porque-precisamos-de-frameworks-mvc)[*https://spring.io/guides/gs/spring-boot/*](https://spring.io/guides/gs/spring-boot/)

[*http://www.ciceroednilson.com.br/configurando-um-projeto-java-com-spring-mvc-e-maven/*](http://www.ciceroednilson.com.br/configurando-um-projeto-java-com-spring-mvc-e-maven/)

[*https://www.codejava.net/frameworks/spring/understanding-spring-mvc*](https://www.codejava.net/frameworks/spring/understanding-spring-mvc)

[*https://www.devmedia.com.br/java-spring-mvc-criando-aplicacoes-web-em-java/31521*](https://www.devmedia.com.br/java-spring-mvc-criando-aplicacoes-web-em-java/31521)

[*https://www.codejava.net/frameworks/spring/creating-a-spring-mvc-project-using-maven-and-eclipse-in-one-minute*](https://www.codejava.net/frameworks/spring/creating-a-spring-mvc-project-using-maven-and-eclipse-in-one-minute)

[*https://spring.io/guides*](https://spring.io/guides)

[*https://www.devmedia.com.br/introducao-a-servlets-em-java/25285*](https://www.devmedia.com.br/introducao-a-servlets-em-java/25285)

[*https://www.devmedia.com.br/introducao-aos-javabeans/8621*](https://www.devmedia.com.br/introducao-aos-javabeans/8621)

[*https://www.devmedia.com.br/primeiros-passos-com-o-spring-boot/33654*](https://www.devmedia.com.br/primeiros-passos-com-o-spring-boot/33654)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Compreendendo o Spring MVC**

O framework Spring facilita o desenvolvimento de aplicativos da web, fornecendo o módulo Spring MVC . O módulo Spring MVC é baseado nos dois padrões de design mais populares - Front controller e MVC .

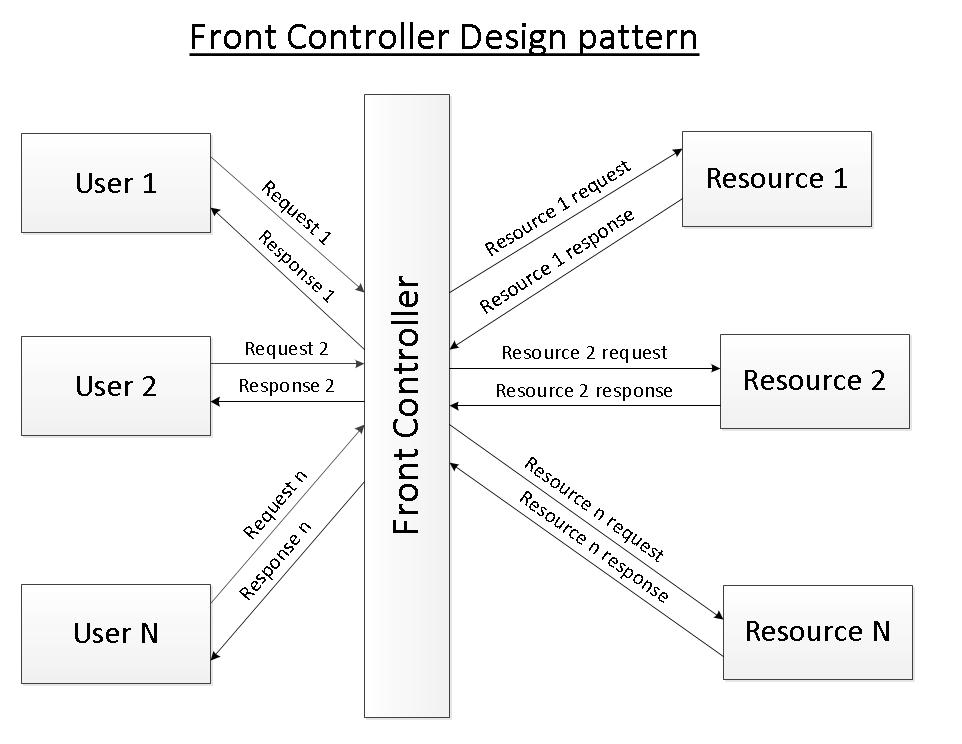
Primeiramente vamos entender sobre o controlador Front e o padrão de design do MVC e, em seguida, vamos ver os detalhes do módulo Spring MVC, sua arquitetura e vários componentes e, depois, começaremos a encaminhar a criação de uma API Web usando o NetBeans ou Eclipse IDE.

**Arquitetura Spring MVC**

* Padrão de design do controlador frontal
* Padrão de design MVC
* Arquitetura MVC do Spring
* Antes de entrar em detalhes da arquitetura Spring MVC, vejamos primeiro os dois padrões de design populares usados ​​para o desenvolvimento da web.

**Padrão de design do controlador frontal**

Esse padrão de design **impõe um único ponto de entrada para todas as solicitações recebidas.** Todas as solicitações são tratadas por um único “pedaço” de código que pode delegar ainda mais a responsabilidade de processar a solicitação em outros objetos de aplicativo.



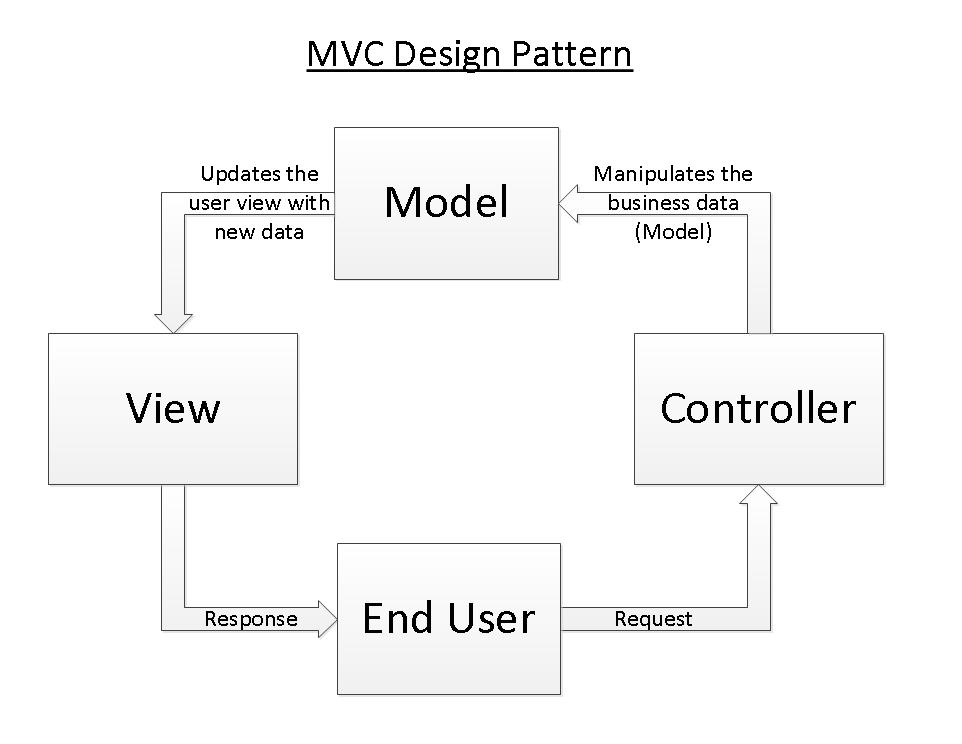
**Padrão de design MVC**

Esse padrão de design nos ajuda a desenvolve**r aplicativos fracamente acoplados, segregando várias preocupações em diferentes camadas. O padrão de design do MVC impõe que o aplicativo seja dividido em três camadas, Model , View e Controlle**r .

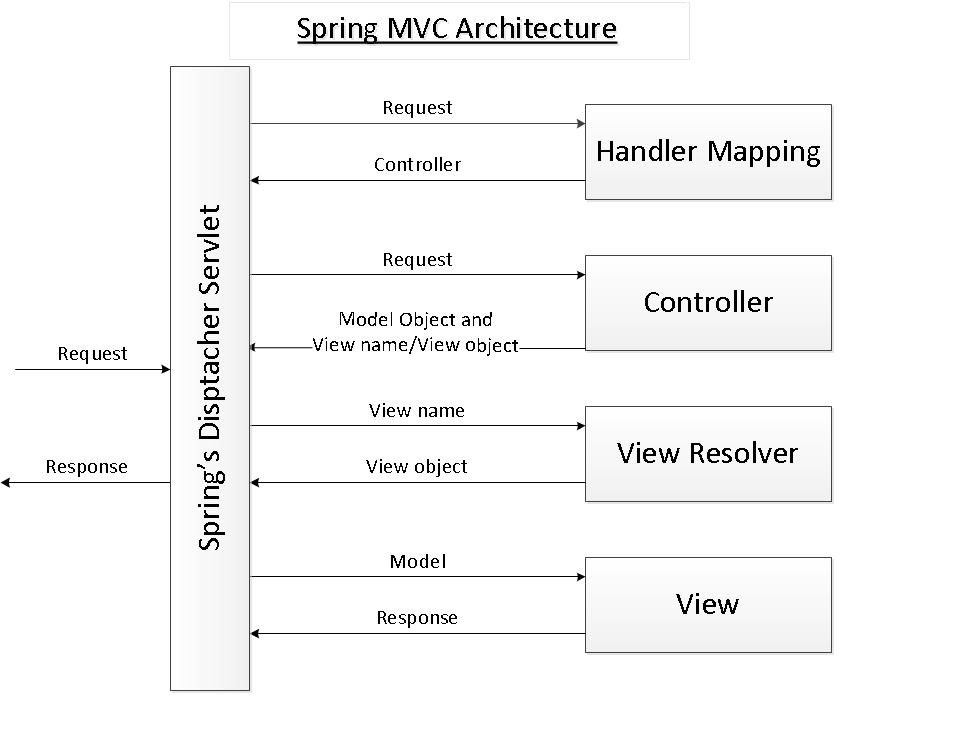
|  |
| --- |
| **Modelo: representa os dados do aplicativo.**  **Exibir: representa a interface do usuário do aplicativo. O View pega o modelo como entrada e o renderiza adequadamente para o usuário final.**  **Controlador: O controlador é responsável por manipular a solicitação e gerar o modelo e selecionar a visualização apropriada para a solicitação** |

**Módulo MVC da Spring**

O módulo MVC da Spring é baseado no padrão de design do **controlador frontal seguido pelo padrão de design do MVC.** Todas as **solicitações recebidas são tratadas pelo servlet único chamado DispatcherServlet, que trabalha como o controlador frontal no módulo MVC do Spring.**

* O **DispatcherServlet** se refere então ao **HandlerMapping**  para encontrar um objeto de controlador que possa manipular a solicitação. DispatcherServlet então despacha a solicitação para o objeto do controlador para que ele possa realmente executar a lógica de negócios para atender à solicitação do usuário. (O controlador pode delegar a responsabilidade de outros objetos de aplicativos conhecidos como objetos de serviço). O controlador retorna um objeto encapsulado que contém o objeto de modelo e o objeto de visualização (ou um nome lógico da visualização). **No MVC do Spring, esse objeto encapsulado é representado pela classe ModelAndView . Caso ModelAndView contenha o nome lógico da visualização, o parâmetro DispatcherServlet refere o ViewResolver para localizar o objeto View real com base no nome lógico.** **DispatcherServlet passa o objeto de modelo para o objeto de exibição, que é renderizado para o usuário final.**

|  |
| --- |
| Afinal, o que é HandlerMapping??  É um manipulado de interfaces, definindo uma interface pública que possa mapear (setar) todos os objetos que compõe o projeto. É uma Interface a ser implementada por objetos que definem um mapeamento entre solicitações e objetos manipuladores, ou seja, podemos fazer a analogia que o HandlerMapping contém os “endereços” de todos os arquivos, classes, objetos que formam o projeto. |

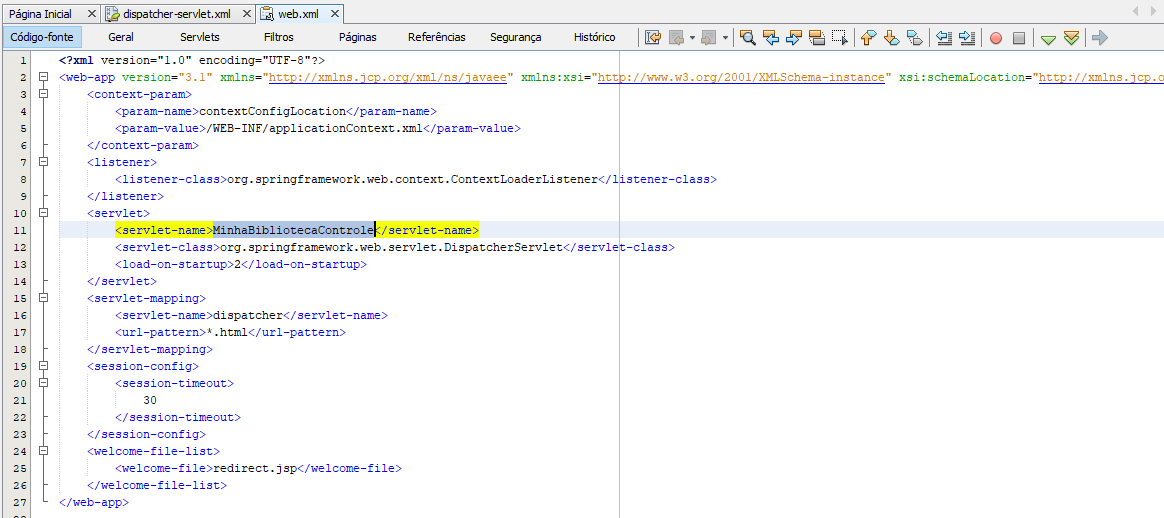


**Servlet do Dispatcher**

Antes vamos relembrar o que é um Servlet:

|  |
| --- |
| O nome “servlet” vem do inglês e dá uma ideia de servidor pequeno cujo objetivo basicamente é receber requisições HTTP, processá-las e responder ao cliente, essa resposta pode ser um HTML, uma imagem etc.  Em Java, procuramos sempre trabalhar orientado a objeto, portanto essa nosso servlet será também um objeto de uma classe Java.  E o funcionamento se dá da seguinte forma:   1. Cliente (navegador) faz uma requisição HTTP ao servidor. 2. O servlet responsável trata a requisição e responde ao cliente de acordo. 3. O cliente recebe os dados e exibe. |

DispatcherServlet como já mencionei, trabalha como o controlador frontal no módulo MVC do Spring. Todas as solicitações do usuário são tratadas por este servlet. Como isso é como qualquer outro servlet, ele deve ser configurado no arquivo do descritor de implementação da web do aplicativo, ou seja, **web.xml**.



Arquivo web.xml

Podemos nomear este servlet de acordo com o nosso projeto, neste caso, ali, na linha 10 está como **dispatcher, o nome padrão**. Podemos alterar o nome deste servlet, no caso da imagem acima, o nome é: **MinhaBibliotecaControle**. O padrão URI na seção de mapeamento de servlet é **" \* .htm " ou .html”**, podemos fazer esta alteração quando construímos o projeto. Portanto, todas as solicitações correspondentes ao padrão URI serão tratadas por este arquivo.

**Contexto do Aplicativo Spring - o que temos e o que podemos fazer**

* Arquivo de contexto de aplicativo padrão
* Arquivo de contexto do aplicativo definido pelo usuário
* Vários arquivos de contexto do aplicativo

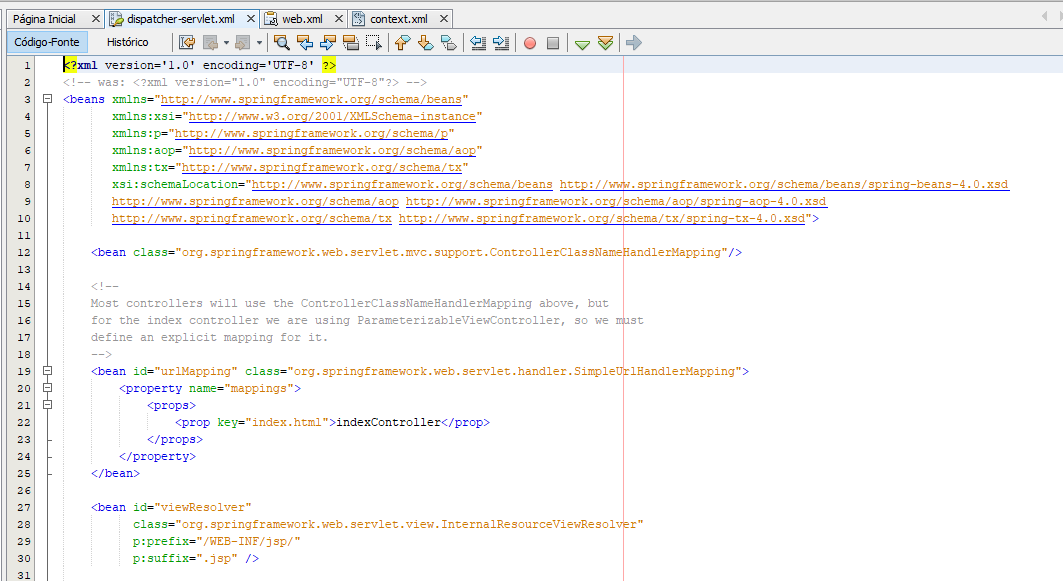
**Arquivo de contexto de aplicativo padrão**

Por padrão, o servlet do dispatcher carrega o contexto do aplicativo **Spring do arquivo XML com o nome [nome do servlet] -servlet.xml (dispatcher-servlet.xml)**. Assim, quando nosso servlet **MinhaBibliotecaControle** for carregado pelo contêiner, ele carregará o contexto do aplicativo **Spring do arquivo XML “ /WEB-INF/MinhaBibliotecaControle-servlet.xml ”.**

**Obs: também podemos deixar o nome padrão.**

|  |
| --- |
| O que são os arquivos XML?   * XML significa eXtensible Markup Language * XML é uma linguagem de marcação semelhante ao HTML * XML foi projetado para armazenar e transportar dados * XML foi projetado para ser auto-descritivo * XML é uma recomendação do W3C   Para estudar mais, clique neste [link](https://www.w3schools.com/xml/xml_whatis.asp) |

Arquivo **dispathcwer-servlet.xml**



**Arquivo de contexto do aplicativo definido pelo usuário**

Podemos substituir o nome e o local do arquivo XML padrão, fornecendo os parâmetros de inicialização para o servlet do distribuidor. **O nome do parâmetro de inicialização é contextConfigLocation.** O valor do parâmetro especifica o nome e o local do contexto do aplicativo que precisa ser carregado pelo contêiner.

**Mapeamentos do Spring Handler**

* BeanNameUrlHandlerMapping
* SimpleUrlHandlerMapping

**Como o nome especifica, o mapeamento do manipulador mapeia a solicitação com o manipulador de solicitação correspondente (na verdade, a cadeia de execução do manipulador).**

|  |
| --- |
| **Como funciona?**   1. Quando uma solicitação chega ao servlet de emissor do Spring, ela entrega a solicitação ao mapeamento do manipulador. 2. O mapeamento do manipulador inspeciona a solicitação e identifica a cadeia de execução do manipulador apropriada e a entrega ao servlet do expedidor. 3. A cadeia de execução do manipulador contém um manipulador que corresponde à solicitação recebida e, opcionalmente, contém a lista de interceptores que são aplicados à solicitação. 4. O servlet do expedidor executa os manipuladores e qualquer interceptador de manipulador associado. |

Há várias implementações de mapeamento de manipulador fornecidas pelo módulo MVC do Spring. Alguns destes são descritos abaixo. **Todas as classes de mapeamentos de manipulador implementam a interface org.springframework.web.servlet.HandlerMapping .**

**Beans, o que são?**

De acordo com (DEVMEDIA): Do ponto de vista arquitetural, componentes são como “caixas pretas”, que oferecem funcionalidades específicas e possuem comportamento conhecido, tendo sido testados e usados em outros projetos. **No Java, os componentes são especificados através do padrão JavaBeans**, que define regras de implementação e empacotamento.

**BeanNameUrlHandlerMapping**

**Essa implementação do mapeamento do manipulador corresponde à URL da solicitação recebida com o nome dos beans do controlador.** O bean correspondente é então usado como o controlador da solicitação. **Esse é o mapeamento de manipulador padrão usado pelo módulo MVC do Spring, ou seja, caso o servlet do expedidor não encontre nenhum bean de mapeamento do manipulador definido no contexto de aplicativo do Spring, o servlet do expedidor usa BeanNameUrlHandlerMapping.**

**Vamos supor que temos três páginas da web em nosso API. O URL das páginas é:**

(fonte: <https://www.codejava.net/frameworks/spring/understanding-spring-mvc>)

**http: // servername: portnumber / ApplicationContext / welcome.htm**

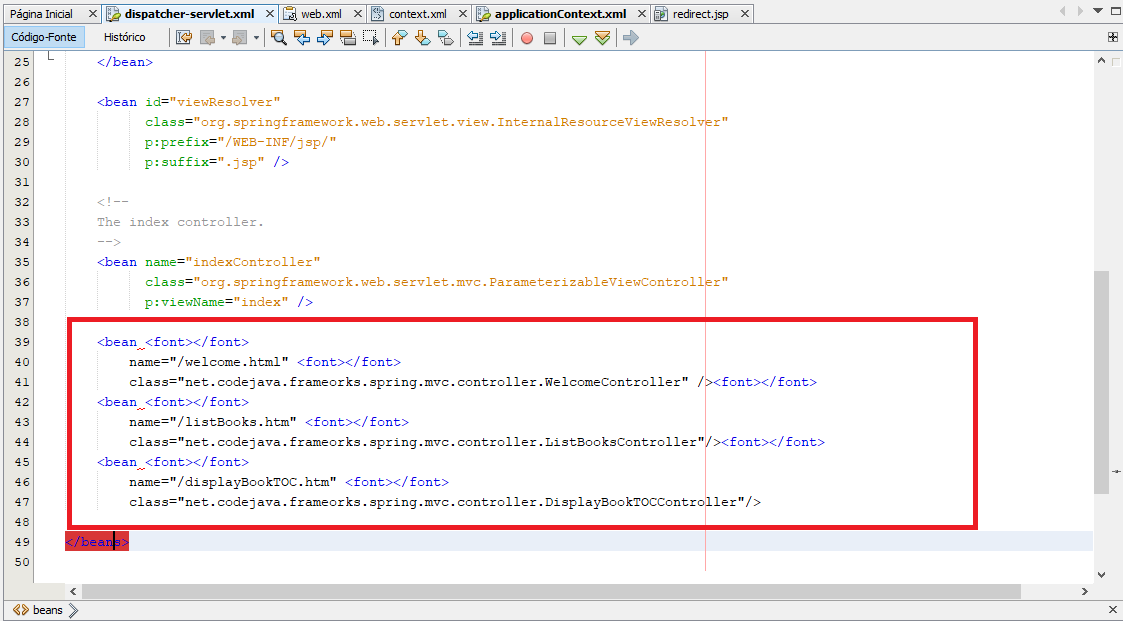
**http: // servername: portnumber / ApplicationContext / listBooks.htm**

**http: // servername: portnumber / ApplicationContext / displayBookContent.htm**

**Os controladores que executarão a lógica comercial para atender à solicitação feita nas páginas acima são:**

|  |
| --- |
| net.codejava.frameorks.spring.mvc.controller.WelcomeController  net.codejava.frameorks.spring.mvc.controller.ListBooksController  net.codejava.frameorks.spring.mvc.controller.DisplayBookTOCController |

Portanto, precisamos definir os controladores no arquivo de contexto do aplicativo Spring, de modo que o nome do controlador corresponda à URL da solicitação. **Os beans do controlador no arquivo de configuração XML terão a aparência abaixo.**



**SimpleUrlHandlerMapping**

O BeanNameUrlHandlerMapping coloca uma restrição no nome dos beans do controlador que devem corresponder à URL da solicitação recebida. O SimpleUrlHandlerMapping remove essa restrição e mapeia os beans do controlador para solicitar a URL usando uma propriedade **"mappings “**

