

# Guia para Iniciantes do WebWork

**Marcelo Martins**

*Este guia descreve os primeiros passos na utilização do WebWork*

## Introdução

Desenvolver sistemas para Web requer mais do que simplesmente um amontoado de páginas JSP's. É importante ter organização, padronização e facilidade de implementação. E isso permite o desenvolvimento de sistemas modulares e de fácil manutenção.

Atualmente existe uma necessidade muito grande em se utilizar programação em três camadas no desenvolvimento de sistemas para web. O desenvolvimento com MVC (Model-View-Controller), separando o modelo de negócios (M) da camada de visualização (V) e do controlador (C) faz com que o desenvolvimento e a manutenção do sistema se tornem muito mais fáceis.

Ferramentas como o WebWork e o XWork desempenham o papel de controlador (C), deixando para o programador as tarefas relativas às regras de negócio e a camada de visualização.

### Apresentação

Antes de entender o WebWork é necessário entender o XWork. O XWork implementa diversos padrões de desenvolvimento. Ele provê poderosas funcionalidades para o processamento de comandos e faz com que a camada de negócios seja independente de bibliotecas específicas para a Web.

O WebWork funciona em cima do XWork dispondo recursos para criação de sistema no ambiente Web permitindo separar a camada de visualização das regras de negócio. Neste guia, muitos conceitos podem ser apresentados como sendo do WebWork mas que na verdade são implementados pelo XWork, no entanto na prática isto não faz diferença pois os 2 trabalham juntos.

### A quem se destina este guia

Este guia é destinado a desenvolvedores que desejam aprimorar suas técnicas de programação e conhecer o WebWork.

Os pré-requisitos para a leitura deste guia são:

#### *Arquitetura de aplicações Web*

É fundamental ter conhecimento de como funciona a arquitetura de uma aplicação web. Conhecer qual a função do servidor e como o navegador mostra as informações. Conhecimento de formulários em HTML será necessário.

#### *Java*

O WebWork é escrito em Java e é utilizado em sistemas desenvolvidos em Java. Então, é necessário o conhecimento e o entendimento da plataforma Java.

#### *JSP e Servlet*

Conhecer a arquitetura J2EE do Java também é um pré-requisito para este guia. É necessário o conhecimento de como os JSP's e os Servlet's funcionam, e como os servidores de aplicações trabalham. Neste guia será utilizado o Apache Tomcat do projeto Jakarta como servidor de JSP e Servlet.

### Escopo deste guia

Este é um guia básico. Sendo assim, não será apresentada detalhadamente todas as características do WebWork. No capítulo de referência pode ser encontrado endereços na internet com mais informações sobre o WebWork e o XWork

## O que é MVC?

Ter códigos de acesso a dados, regras de negócio e códigos de apresentação misturados, pode gerar muitos problemas e dificultar muito a manutenção. Por causa da interdependência entre as partes do código, alterar um trecho pode refletir efeito e qualquer outra parte e tornar o sistema não funcional. Com códigos misturados a reutilização de código é muito difícil, ou impossível. Quando se deseja alterar a visualização do sistema, ou quando se deseja mudar de banco de dados, o sistema precisa ser quase inteiramente re-escrito, e isso pode gerar muitos outros problemas.

Para resolver estes problemas, existe o padrão de desenvolvimento chamado MVC Model 2 (neste guia chamado simplesmente de MVC) que visa separar a camada de visualização das regras de negócios através de um controlador.

Quando um sistema é desenvolvido utilizando a arquitetura MVC, os códigos referentes às diferentes camadas ficam separados do resto do sistema, e sua manutenção se torna mais simples. A troca completa de uma camada pode ser feita sem gerar efeitos nas outras. Outra vantagem é que múltiplas camadas de visualização, por exemplo, podem compartilhar da mesma camada de negócios.

Definindo melhor:

*Model:* é onde estão as regras de negócio e o acesso aos dados no sistema.

*View:* é onde os dados serão apresentados ou requisitados ao usuário. Os dados são acessados na base pela camada Model e repassados para a camada View para serem apresentados ou são requisitados pela camada View e repassados para a camada model para serem persistidos.

*Controller:* o Controller é o responsável por fazer a ligação do View com o Model. É o controller que vai saber qual ação do Model deve ser chamada em cada requisição do View.

## O WebWork

É um projeto comunitário de código aberto criado para prover ferramentas para criação de sites web. Sites devem ser criados em um curto período de tempo sendo fácil de entender e fácil de manter. A motivação do WebWork é para manter o desenvolvimento o mais simples possível, sem esquecer da flexibilidade.

Os seguintes recursos estão disponíveis no WebWork:

### *Ações*

O gerenciamento de ações ou de comandos é a característica mais básica do WebWork. Descrevendo uma ação no arquivo de configuração e criando uma classe Java para esta ação é o suficiente para ver o WebWork em funcionamento.

### *Redirecionamento*

Como resultado de uma ação temos um redirecionamento. Ao ser chamada uma ação, esta retorna para onde deseja redirecionar a aplicação na camada de visualização.

### *Validação de formulários*

O WebWork pode automatizar a validação de formulários com arquivos simples em XML, separando e facilitando sem a necessidade de programação em Java.

### *Componentes*

Com a característica de Inversão de Controle do XWork, é possível ter objetos que ficam disponíveis para as ações em um determinado escopo.

### *Interceptadores (Interceptors)*

Com os interceptors, uma ação pode ser interceptada antes e depois da sua execução podendo ter seu fluxo desviado. Eles são modulares e múltiplos interceptors podem ser utilizados para uma ação.

### *Internacionalização*

O WebWork possui arquivos de recursos separados por ações e por línguas. Isso permite a fácil criação de recursos internacionalizados nos aplicativos web.

## Comparando com o Struts

É inevitável a comparação do WebWork com o Struts. O propósito dos dois é o mesmo, ser um controlador na arquitetura MVC, mas eles trabalham de forma um pouco diferente.

Algumas diferenças são listadas a seguir:

### *Abstração da API do Servlet*

Uma ação do Struts é totalmente dependente dos servlets porque a ação que deve ser estendida é uma extensão de `HTTPServlet`. Na assinatura do seu método de entrada já é necessário ter os atributos `ServletRequest` e `ServletResponse` como parâmetros. No modelo de ações do XWork, as ações não dependem da API do Servlet e podem funcionar sem estar na web. É possível ter acesso ao contexto do servlet através da classe `ActionContext` mas isto não é necessário.

### *Facilidade de implementação*

O WebWork é bem mais simples de se trabalhar do que o Struts. Criar uma ação no WebWork é mais produtivo, não é necessário criar os `FormBeans` do Struts, e o conceito `Model Driven` do WebWork faz com que os dados do formulário sejam colocados automaticamente em uma classe de negócio.

### *Testabilidade*

Testar uma ação do Struts é um processo trabalhoso, tendo em vista que os objetos de `Request` e `Response` são necessários. Já uma ação do WebWork pode ser facilmente testada por ser independente da API do Servlet.

### *Interceptadores modulares*

Os interceptadores do WebWork são modulares, e podem ser implementados sem ligação direta com uma determinada ação. No Struts os processamentos feitos antes e depois das ações estão ligados a hierarquia das classes de ação, o que dificulta a modularização e a implementação de múltiplos interceptadores.

Mais detalhes podem ser encontrados na página do WebWork em <http://wiki.opensymphony.com/display/WW/Comparison+to+Struts>.

## Instalação e Configuração

Para utilizar o WebWork é necessário ter o J2SDK instalado no computador bem como um servlet container. É recomendável que seja utilizado a versão do J2SDK™ 1.4.x que pode ser encontrada em <http://java.sun.com>. O servlet container utilizado nos exemplos é o Apache Tomcat 5.0, mas a versão 4.1 pode ser utilizada perfeitamente. O download do Apache Tomcat pode ser feito em <http://jakarta.apache.org/tomcat>.

O download do WebWork pode ser feito na página <https://webwork.dev.java.net/servlets/ProjectDocumentList>. O arquivo que deve ser feito o download é o `webwork-X.X.zip`. Onde X.X é a versão do WebWork. Prefira sempre a mais nova.

O primeiro passo para instalar o WebWork é copiar o arquivo `webwork-X.X.jar` e todos os arquivos da diretório `lib/core` para o diretório `WEB-INF/lib` da aplicação web.

Depois, é necessário configurar o arquivo `web.xml` que serve para configurar a aplicação web. Será adicionado ao arquivo o trecho.

```
<servlet>
  <servlet-name>webwork</servlet-name>
  <servlet-class>
    com.opensymphony.webwork.dispatcher.ServletDispatcher
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>webwork</servlet-name>
  <url-pattern>*.action</url-pattern>
</servlet-mapping>
```

Esta configuração mapeia o servlet ServletDispatcher do WebWork e faz com que todas as ações que terão a extensão \*.action para este servlet que se encarregará de chamar a ação correspondente.

A seguir segue o arquivo web.xml completo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <display-name>Guia para Iniciantes do WebWork</display-name>

  <servlet>
    <servlet-name>webwork</servlet-name>
    <servlet-class>
      com.opensymphony.webwork.dispatcher.ServletDispatcher
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>webwork</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>

</web-app>
```

Depois de configurar o container com o arquivo web.xml é necessário configurar o XWork. A configuração é feita no arquivo xwork.xml que deve ficar em WEB-INF/classes. O arquivo a seguir não faz nada, é a configuração mais básica do Xwork.

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>

  <include file="webwork-default.xml" />
  <package name="default" extends="webwork-default">

  </package>

</xwork>
```

## Ações e Resultados

As ações são as classes que recebem a requisição do WebWork. Estas classes devem estar configuradas no arquivo xwork.xml e deve estar prontas para receber a requisição do WebWork.

Utilizando o WebWork, os JSP's terão apenas a tarefa de controlar a visualização do sistema, a camada View, e eles é quem irão fazer a requisição das ações ao WebWork.

Os resultados são o destino do WebWork após executar uma ação. Ao ser configurada uma ação, devem ser especificados todos os possível retornos, definindo ainda o tipo de retorno. Abaixo segue alguns retornos do WebWork:

### *dispatcher*

Encaminha o resultado para um determinado local, este é o tipo de retorno mais utilizado no WebWork.

### *redirect*

Encaminha o resultado para um determinado local. Diferente do dispatcher, o redirect não envia as propriedades da ação para a página de resultado.

### *velocity*

Encaminha o resultado para um arquivo de modelo do velocity. O WebWork trabalha nativamente com o Velocity e sua configuração é bem simples.

### *chain*

Encaminha o resultado de uma ação para outra ação.

Os seguintes passos serão utilizados para criar uma ação no WebWork.

- 1) Configurar a ação no arquivo xwork.xml;
- 2) Criar uma página JSP que irá chamar a ação;
- 3) Criar a classe da ação;
- 4) Criar uma página JSP que irá receber o retorno da ação.

Voltando ao arquivo xwork.xml para configurar a ação. Na tag <package> é aonde vai a configuração das actions e interceptors.

Uma melhor definição de interceptors está fora do escopo deste guia, mas é possível entender eles como sendo os responsáveis em preparar o ambiente de execução da ação e do retorno da ação.

Será utilizado o interceptor defaultStack que é o padrão para utilização básica do WebWork. Para incluir ele é utilizada a tag <default-interceptor-ref name="defaultStack"/>.

Para configurar uma ação é utilizado a tag <action> e dentro da tag action é possível ter diversas tags <result>, indicando os resultados da ação. É possível ter diversas tags <interceptor-ref> para interceptadores específicos da ação.

Será criado um exemplo de ação, que será uma página que perguntará o nome e a idade e de uma pessoa e se esta pessoa já fez aniversário este ano. Como resposta terá o ano de nascimento dessa pessoa. Abaixo a configuração completa da ação no arquivo xwork.xml.

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>

  <include file="webwork-default.xml" />

  <package name="default" extends="webwork-default">

    <default-interceptor-ref name="defaultStack" />

    <action name="AnoNascimento" class="guia.CalculaAnoNascimento">

      <result name="error" type="dispatcher">
        leIdade.jsp
      </result>

      <result name="success" type="dispatcher">
        mostraAno.jsp
      </result>

    </action>

  </package>

</xwork>
```

A estrutura de diretórios utilizada no exemplo é a seguinte:

```
/guia
  /WEB-INF
    /classes
    /lib
    /src
```

O diretório /guia, deve ficar dentro do diretório webapps do Apache Tomcat. As fontes das classes de ações serão colocadas dentro do diretório /src, mas podem ficar em qualquer lugar, inclusive fora da aplicação. O importante é que as classes compiladas fiquem no diretório /classes.

Agora será criada a página JSP inicial leIdade.jsp que requisita os dados do usuário.

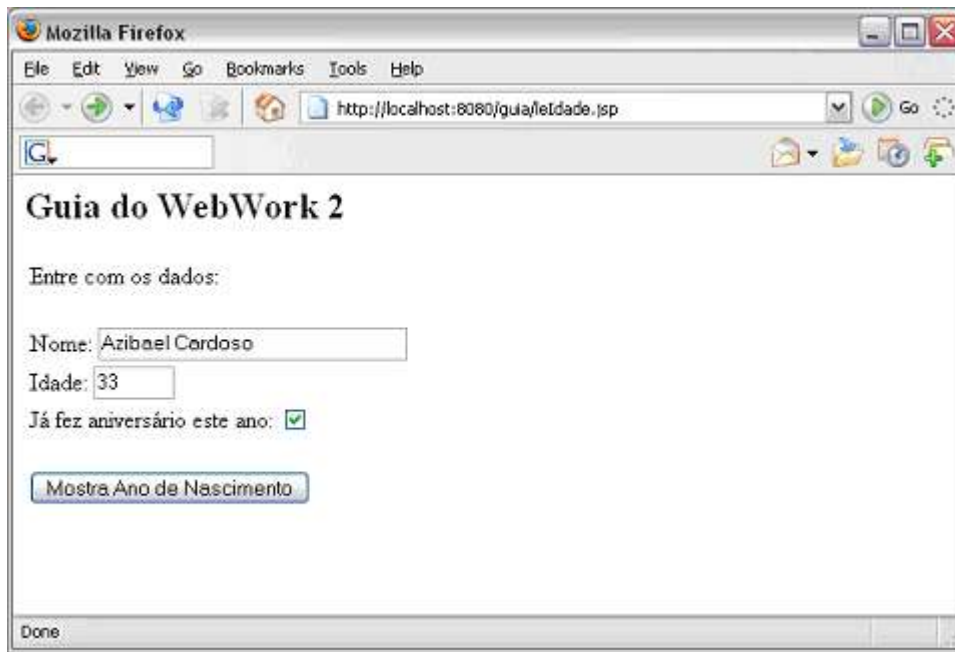
```
<html>
<body>
<h2>Guia do WebWork</h2>
<form method="POST" action="AnoNascimento.action">
Entre com os dados:<BR><BR>
Nome: <input type="text" name="nome" size="30"><BR>
Idade: <input type="text" name="idade" size="5"><BR>
```

```
Já fez aniversário este ano: <input type="checkbox" name="aniversario" value="true"><BR><BR>
<input type="submit" value="Mostra Ano de Nascimento">
</form>
</body>
</html>
```

Este é um HTML simples, com um formulário e três campos. O importante do código é a ação que será chamada pelo formulário. Como é possível ver, ao enviar o formulário será chamado AnoNascimento.action.

Este não é um arquivo, mas sim um direcionamento. No arquivo web.xml, foi configurado para que tudo que termine com \*.action seja direcionado ao WebWork. E no xwork.xml foi configurado uma ação AnoNascimento. É desta forma que as ações do WebWork são chamadas, o nome da ação definido no xwork.xml mais um ponto e a palavra action.

A figura a seguir ilustra como ela deve aparecer no navegador.



Agora será criada a classe de ação CalculaAnoNascimento.java, segue abaixo o código.

```
package guia;

import com.opensymphony.xwork.ActionSupport;
import java.util.Calendar;

public class CalculaAnoNascimento extends ActionSupport {

    private String nome;
    private Integer idade;
    private Boolean aniversario;
    private Integer ano;

    public String execute() throws Exception {

        if (nome == null || nome.equals("") || idade == null || idade.equals("")) {
            return ERROR;
        }

        if (aniversario == null) {
            aniversario = new Boolean(false);
        }

        if (!aniversario.booleanValue()) {
            idade = new Integer(idade.intValue()+1);
        }
    }
}
```

```
Calendar data = Calendar.getInstance();
data.add(Calendar.YEAR, (idade.intValue()*-1));
ano = new Integer(data.get(Calendar.YEAR));

return SUCCESS;
}

public String getNome() {
return nome;
}

public void setNome(String nome) {
this.nome = nome;
}

public Integer getIdade() {
return idade;
}

public void setIdade(Integer idade) {
this.idade = idade;
}

public Boolean getAniversario() {
return aniversario;
}

public void setAniversario(Boolean aniversario) {
this.aniversario = aniversario;
}

public Integer getAno() {
return ano;
}

public void setAno(Integer ano) {
this.ano = ano;
}
}
```

No WebWork, as classes de ação devem implementar a interface Action e devem ter um método público execute(). É possível também estender a classe ActionSupport que já implementa a interface Action. No exemplo, a classes de ação esta estendendo a classe ActionSupport, e tem o método execute () retornando uma String. Cada vez que a ação for chamada uma nova instancia da classe é criada e o método execute() é invocado. O retorno da classe irá dizer ao WebWork para qual dos retornos configurados no arquivo xwork.xml será direcionada.

Existem alguns retornos definidos como padrão do WebWork, eles são: success, error, input, login. Para cada um desses retornos existe uma constante para retornar do método execute. É importante entender que não é obrigatório o uso destas strings. É perfeitamente possível utilizar o retorno sucesso, por exemplo, na configuração do xwork.xml e no retorno do método execute().

Esta classe recebe os parâmetros do formulário, calcula e retorna o ano de nascimento. Muitas coisas aparecem neste código.

Primeiro, onde estão os dados do formulário HTML? O WebWork utiliza o conceito de Inversão de Controle e Injeção de Dependência, isto faz com que ele seja muito poderoso, e criar ações seja muito simples. Os dados do formulário HTML são injetados dentro da classe em seus respectivos atributos. Por exemplo, o campo do formulário que possui name="nome" será repassado para o atributo nome, através do método setNome(String nome). E assim por diante, o atributo do formulário idade será repassado para o atributo idade e o campo aniversario será repassado para o atributo aniversario.

Então, podemos ter certeza de que, quando iniciar a execução do método execute() as propriedades já estarão setadas, caso não tenham sido deixadas em branco.

No código da classe da ação, primeiro é feito a verificação da integridade dos dados. Valores em branco chegam como null. Caso algum campo seja null, a classe retorna para a página de entrada de dados. Esse retorno é feito através do return ERROR.

O WebWork vai no arquivo xwork.xml e verifica para onde deve ir quando o retorno é ERROR, no caso, está configurado para retornar para a página de entrada de dados.

Caso o código seja executado corretamente, o retorno é SUCCESS. Esse retorno, no arquivo xwork.xml diz ao WebWork para direcionar a aplicação para o arquivo mostraAno.jsp que segue abaixo.

```
<%@ page import="com.opensymphony.xwork.util.OgnlValueStack"%>

<html>
<body>
<h2>Guia para Iniciantes do WebWork</h2>

<%
OgnlValueStack stack = (OgnlValueStack) request.getAttribute("webwork.valueStack");

String nome = (String)stack.findValue("nome");
Integer ano = (Integer)stack.findValue("ano");
%>

Olá <%=nome%>.<BR><BR>
Você nasceu em <%=ano.toString()%>
</body>
</html>
```

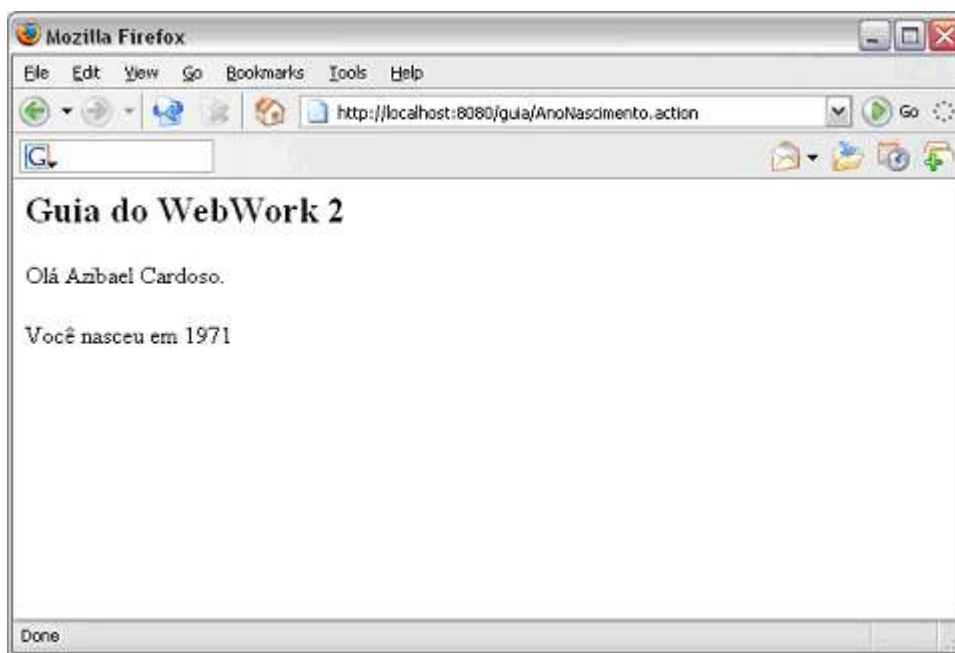
Este é um arquivo JSP simples, com alguns scripts JSP. Primeiro é preciso entender o ValueStack do WebWork. Quando utilizamos o tipo de retorno dispatch o WebWork deixa disponível para o JSP redirecionado todos os atributos da ação chamada, desde que tenham o método get apropriado. Esta disponibilidade é feita através da classe OgnlValueStack que pode ser obtida da sessão do request com o nome de webwork.valueStack.

No exemplo, a primeira linha importa a classe OgnlValueStack. Depois é feita a carga do objeto através da sessão com o nome webwork.valueStack.

Neste ponto, é possível acessar os atributos da classe de ação, CalculaAnoNascimento, simplesmente chamando o método findValue() e passando o nome do atributo como parâmetro. O valor retornado será o mesmo do atributo original.

Aqui já é possível ter uma boa visão de como o WebWork trabalha, como é a criação de ações e ainda como é feito a passagem de parâmetros da camada view para dentro da ação e vice-versa.

A tela de retorno da ação é demonstrada a seguir.



## Tags Personalizadas do WebWork

Existe um recurso do WebWork que são as tags personalizadas do WebWork. Este recurso, apesar de não ter sido utilizado nos exemplos, é muito recomendado. Estas tags dão um poder maior ao WebWork e facilitam o desenvolvimento de camadas de visualização em JSP.

No exemplo anterior, a página JSP de retorno teve que buscar os dados em um objeto na sessão, mas poderia ser feito de maneira diferente. Para utilizar as tags do WebWork é necessário incluí-las no arquivo JSP com a diretiva:

```
<%@ taglib uri="webwork" prefix="ww" %>
```

A partir daí, já é possível utilizar as tags do WebWork, e no exemplo do arquivo de retorno, poderia ser utilizada a tag `<ww:property value="nome"/>` para recuperar o nome e `<ww:property value="ano"/>` para recuperar o ano dos atributos da ação. Note que o código ficará mais limpo e simples. O processo do WebWork é o mesmo, mas as tags fazem o trabalho implicitamente, e deixa o código JSP bem mais limpo.

A seguir o arquivo de retorno da ação `mostraAno.jsp` com as tags do WebWork.

```
<%@ taglib uri="webwork" prefix="ww" %>

<html>
<body>

<h2>Guia para Iniciantes do WebWork</h2>
Olá <ww:property value="nome"/>.<br><br>
Você nasceu em <ww:property value="ano"/>

</body>
</html>
```

### Isso é só o começo

Todos os conceitos aqui apresentados representam uma pequena parte do poder do WebWork e do XWork. É muito importante entender bem os conceitos e, a partir daí, buscar mais informações sobre o WebWork.

Conceitos como os componentes e os interceptadores são extremamente úteis na construção de sistemas baseados em web. O poder dos validadores e o sistema de internacionalização fazem com que sistemas personalizados sejam facilmente construídos.

Nos endereços de referência deste guia há mais exemplos e detalhes sobre todo o poder do WebWork.

## Referências

Design Patterns: Model-View-Controller  
<http://java.sun.com/blueprints/patterns/MVC.html>

Grupo de Usuários Java – Fórum e Artigos  
<http://www.guj.com.br/>

The J2EE™ 1.4 Tutorial  
<http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html>

WebWork Documentation  
<http://www.opensymphony.com/webwork/>

WebWork Documentation Wiki  
<http://wiki.opensymphony.com/display/WW/WebWork>

(ootips) Model-View-Controller  
<http://ootips.org/mvc-pattern.html>

## Conclusão

Neste pequeno guia é possível perceber que o WebWork é um framework muito poderoso e de fácil aprendizado. Vale a pena estudar mais a fundo e explorar seus recursos avançados.

**Marcelo Martins** ([marcelomartins2@yahoo.com.br](mailto:marcelomartins2@yahoo.com.br)) é desenvolvedor de sistemas a quase 10 anos trabalhando com Java a 2 anos com WebWork, Velocity e Hibernate.