Apache Wicket

Cristiano Kliemann, 2009

Introdução

- Este é um curso apenas introdutório ao Wicket.
  - Não é possível, em tão pouco tempo, falar sobre tudo.
  - É importante continuar com estudos e experimentos.
  - Ao final do treinamento, você terá muito mais condições de buscar mais informações por conta própria.
  - Peça algum tempo ou um projeto não crítico ao seu chefe! :-)

Introdução

- Em diversos momentos irei me repetir. A idéia é apresentar um pouco de cada coisa e ir aprofundando em ciclos.
- Me interrompam imediatamente:
  - Quando tiverem dúvidas
  - Se estiver indo muito rápido ou muito devagar
  - Não tenham medo de perguntar: qualquer informação pode ser importante – lembrem-se que isso é muito diferente de Struts

O que é Wicket?

- Framework web baseado em componentes.
- Patrocinado pela Apache: http://wicket.apache.org
- “Concorrente” de JSF
- Embora se pareça com JSF, parte de uma visão bem diferente.
  - **Na prática**, as coisas são bem diferentes.
- Vários pontos fortes, com destaque a gerência de estado
  - Páginas são *stateful* com suporte ao botão voltar do browser.

Breve História

- Jun/2005: Versão 1.0 no SourceForge. As versões até 1.2.x podem ainda ser pegas lá (http://sourceforge.net/projects/wicket/).
- Out/2006: aceito na Apache para incubação
- Jun/2007: graduou-se top level.
- Jan/2008: lançada a versão 1.3.0, a primeira sob o nome Apache Wicket.
- Versão atual: 1.3.x
- Em desenvolvimento: versão 1.4.0, que oferece suporte a Generics de Java 5.

Problemas na web

- MVC desacopla a ação, o fluxo da aplicação e a apresentação.
  - Resolve bem pouca coisa a mais.
- Na prática, o resto fica muito parecido com o fluxo requisição-resposta de um servidor HTTP.
- HTTP foi criado para servir páginas estáticas.

Problemas na web

- Sistemas web não são páginas estáticas.
  - São aplicativos (parecidos com aplicativos desktop) que
    - rodam em servidor;
    - usam o browser como plataforma de interação com usuário.
  - Telas normalmente podem ser organizadas em painéis, formulários, campos, lembrando muito pouco os documentos estáticos da web.
  - A maioria dos programadores perceberá que é natural organizar a tela em componentes. Programadores GWT, Swing e SWT sabem disso.
  - Por que a programação web precisa ser diferente?

Problemas na web

- Poucos frameworks lidam com estado – parece um problema complexo.

Wicket – Primeiras informações

- Orientado a **páginas e componentes**, a não a ações
  - O principal ponto de extensão é a página – é por onde "a programação começa" quando se quer implementar uma funcionalidade (com Struts, é a *action*).

- Programar para Wicket tem semelhanças com programação de GUIs como Swing ou Delphi.

Wicket – Primeiras informações

- Wicket é um framework *stateful* – guarda instâncias de páginas no servidor.
  - Mesmo assim, gerencia muito bem a memória – serializa instâncias antigas.

- Nos exercícios, esqueçam Struts!
  - Não pensem em campos *hidden*, em armazenar ids em combo-boxes, essas coisas. Inicialmente pensem em classes e objetos apenas.

Wicket – Java é Java

- O Wicket não tenta reduzir o uso de Java – ao contrário: usa Java sempre que possível.
- Componentes e páginas Wicket são classes Java que você manipula apenas com Java.
- Você constrói os componentes e manipula a hierarquia deles na página com construções Java comuns.
  - Ex: para instanciar um componente, você o constrói com new: Link link = new Link(...).

Wicket – Java é Java

- Isso possibilita que você:
  - use todos os pontos fortes da linguagem – herança, tipagem estática, etc;
  - aproveite as melhores funcionalidades das IDEs – refactoring, ajuda para completar código, navegação pelo código;
  - aprenda com mais facilidade o funcionamento do Wicket e dos sistemas.

Wicket – Java – Exemplo

```
public class HomePage extends WebPage {
    public HomePage() {
        Label label = new Label("mensagem", "Olá, Mundo!");
        add(label);
    }
}
```

Wicket – HTML

- Java é muito bom para manipular os componentes, responder a eventos, etc. Mas não é bom para definir layout.
  - HTML geralmente é melhor.

- A parte de apresentação com Wicket é feita com templates HTML.

Wicket – HTML

- Diferença com outros frameworks: força o uso de “templates limpos”.
  - Não há *scriptlets*, *bindings*, *expression language*, *taglibs*, etc.
  - Há apenas marcadores que ligam um elemento HTML ao o componente Java instanciado pelo desenvolvedor.
  - Todo o resto se faz em Java.

HTML - Exemplos

- JSP:

```
<ul>
<c:forEach var="item" items="${requestScope.items}">
  <li><c:out value="item.text"/></li>
</c:forEach>
</ul>
```

- Wicket:

```
<ul wicket:id="lista">
  <li wicket:id="item">Aqui vai o item da lista</li>
</ul>
```

- Não há ligação com os dados a serem mostrados nem lógica relacionada ao loop no template Wicket.
- Isso é feito exclusivamente na parte Java.

Mais informações sobre templates

- Por que o Wicket não permite lógica nos templates?

Mais informações sobre templates

- Problemas de lógica no template:
  - *Spaghetti code*
  - O compilador pode te ajudar muito pouco – muitas verificações são feitas apenas em execução.
  - As IDEs também ajudam pouco (*refactoring*, por exemplo). Mesmo que a IDE ofereça bom suporte a JSPs ou templates Facelets, por exemplo, dificilmente será tão bom quanto para Java.
  - É complicado trabalhar com os designers. Muitas vezes joga-se fora o HTML que elem mandam.
  - Muito difícil criar abstrações reusáveis.

Referências – **Usem sempre!**

- Site: http://wicket.apache.org
- Livro: “Wicket in Action”: http://www.manning.com/dashorst/
  - O mais recente e provavelmente melhor livro sobre Wicket. Muito bom.
- Boa introdução: http://wicket.apache.org/introduction.html
- Wiki, com bastante conteúdo:
  - http://cwiki.apache.org/WICKET/
  - Muito importante: http://cwiki.apache.org/WICKET/reference-library.html.

Referências

- Exemplos:
  - http://www.wicketstuff.org/wicket13/ - o melhor conjunto de exemplos. Rodando e com fontes. Vejam o link “Source code” na parte superior direita das telas.
  - http://wicket.apache.org/examples.html

Exemplo – Olá, Mundo!

☺ Exemplo: hello

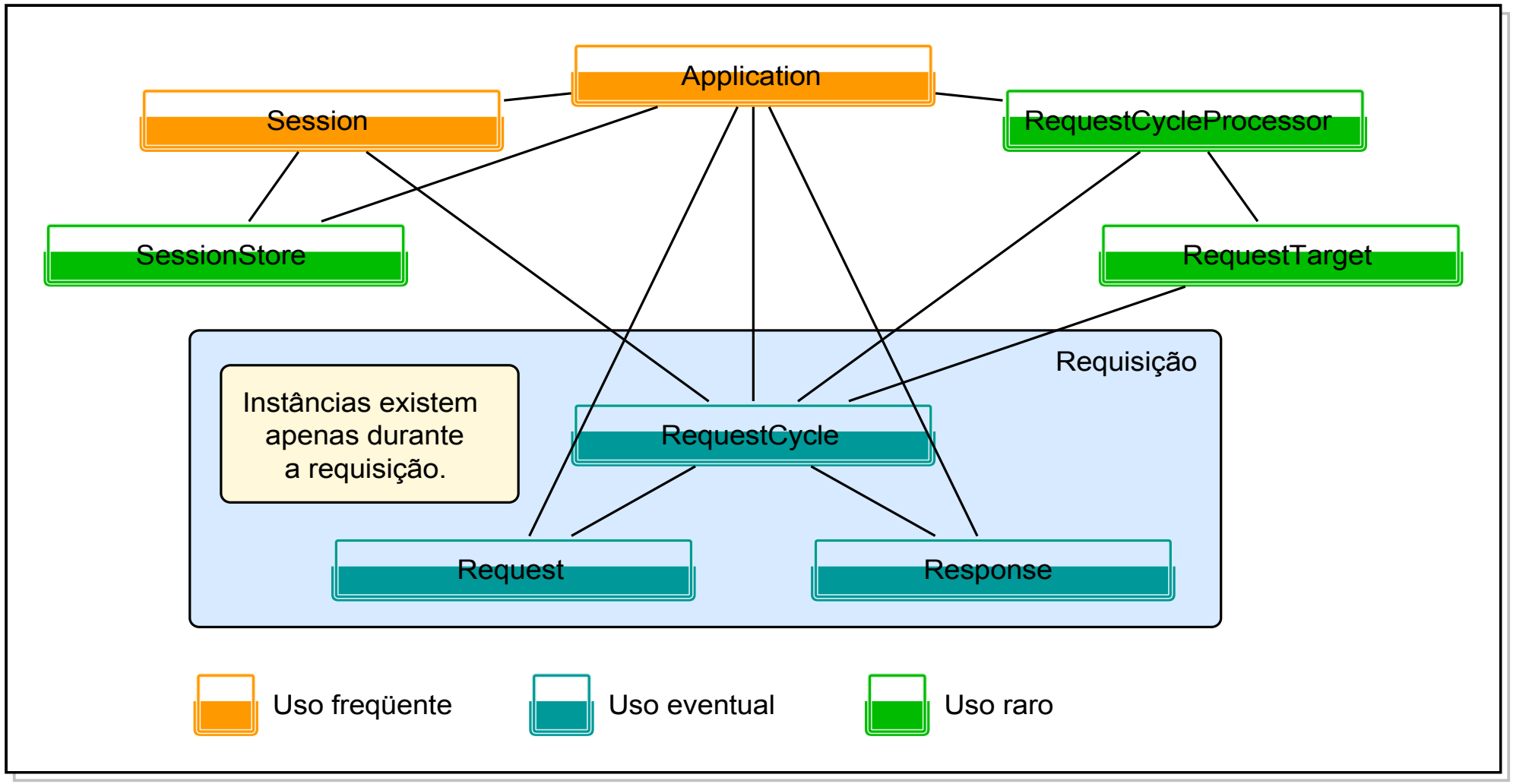
- Ambiente:
  - JDK 5, Maven 2.1.0, Eclipse 3.4
- Em todos os exemplos:
  - Execute “mvn eclipse:eclipse” no diretório do exemplo
  - Importe o projeto no Eclipse
  - Execute a classe “Inicio” como Java Application
  - No browser, acesse “http://localhost:8080”

Exemplo – Um link

- O segundo exemplo.

- Exemplo: hello-com-link

Arquitetura Wicket – Visão Geral



Arquitetura Wicket – Visão Geral

- Para alguns elementos, há uma subclasse com o prefixo “Web” (ex: WebApplication), que é a implementação normalmente usada

- Não se assustem – veremos apenas
  - Application
  - RequestCycle
  - Session

Application

- Ponto inicial de implementação de um aplicativo.
- Usa-se estendendo WebApplication e implementando alguns métodos
  - **Isso é obrigatório.**
- Há apenas uma instância por aplicação.

Application

- Local onde as configurações são feitas
  - por código, no método init().
- Factory para alguns outros elementos Wicket
  - Session, RequestCycle
- É o primeiro ponto de extensão do Wicket.
  - Ex: para implementar um Request próprio, é preciso reimplementar o RequestCycle padrão e depois ainda reimplementar o método newRequestCycle de Application.

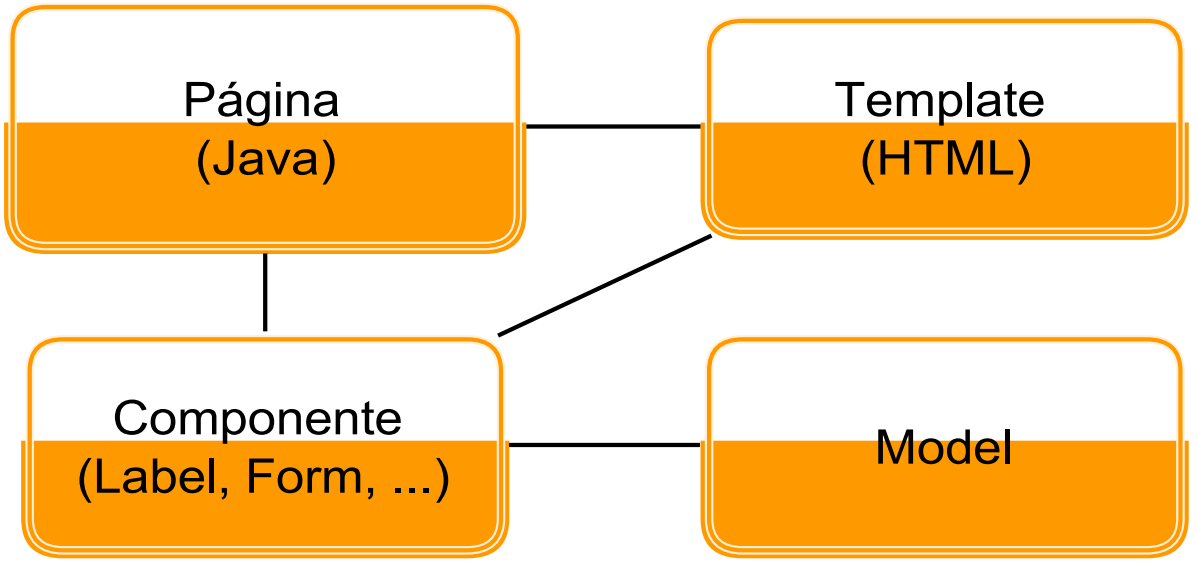
RequestCycle

- Responsável por processar uma requisição, geralmente delegando ao RequestCycleProcessor.
- Guarda Request, Response e o RequestTarget.
- Uma instância por requisição. Depois de terminada, a instância é descartada.
- Bastante utilizado pelos programadores.

```
RequestCycle rc = RequestCycle.get();
// Mais comum:
WebRequestCycle rc = (WebRequestCycle) RequestCycle.get();
```

O quarteto Página, Componente, Template, Model

- Elementos que farão obrigatoriamente parte do seu dia-a-dia com Wicket



Páginas

- São classes Java estendidas pelo programador acompanhadas por um template HTML.
- Representam.... ora, páginas...
  - O equivalente a uma janela em uma interface gráfica desktop.
- A primeira página exibida ao usuário:
  - Definida pelo método Application.getHomePage()
  - Ele retorna a classe da página, que será instanciada automaticamente pelo Wicket.

Páginas

- Uma página é instanciada pelo Wicket ou pelo programador.
  - Após ser mostrada, normalmente fica em memória “para sempre” (é possível configurar) e seus métodos podem redirecionar o cliente para outra classe ou instância de página.
  - Isso permite o funcionamento o botão voltar do browser.
  - Todo o histórico do usuário fica na sessão.

Páginas

- Páginas contém instâncias de **componentes**.
- Na realidade, páginas são também componentes (mas isso é só um detalhe).

Componentes - introdução

- São elementos reusáveis (classes) que representam partes dinâmicas de uma página
  - mensagens, formulários, campos, botões, etc
- São classes Java instanciadas e manipuladas pelo programador.
- São extensíveis: para modificar um componente, basta estendê-lo.
- Conceito muito comum em GUIs: Swing, SWT, Delphi.

Componentes - introdução

- Há muitos componentes prontos no Wicket.
- Cada componente em uma página deve ter um id String e um elemento corresponde no template, com o atributo “wicket:id” igual.
- Exemplo no próximo slide.

Componentes - template

- Uma página:

```
public class HomePage extends WebPage {
    public HomePage() {
        Label label = new Label("mensagem", "Olá, Mundo!");
        add(label);
    }
}
```

- Trecho do template:

```
<body>
<span wicket:id="mensagem">O "Olá, Mundo" virá aqui.</span>
</body>
```

- Já vimos isso no projeto “hello”. Vamos rever?

Componentes – Hierarquia

- Componentes podem ter outros componentes internos.
  - Um exemplo comum é o Form.
- Assim, a página é formada por uma hierarquia de componentes.

Componentes – Hierarquia

- Classe (construtor):

```
public HomePage() {
    Form form = new Form("form");
    add(form);

    form.add(new TextField("nome"));
}
```

- Trecho do template:

```
<form wicket:id="form">
Nome: <input type="text" wicket:id="nome">
</form>
```

- A hierarquia de componentes na página deve ser exatamente a mesma do HTML.

Componentes – Hierarquia

- Não é necessário criar componentes para todo o HTML
  - apenas para o que for dinâmico
- A classe anterior funcionaria com o template abaixo:

```
<form wicket:id="form">
<div class="campo">
Nome: <input type="text" wicket:id="nome">
</div>
</form>
```

- Veja que não foi preciso incluir um componente para o elemento .
- Exemplo: hierarquia-componentes

Mais sobre componentes e templates

- Qual elemento HTML é o correto para um componente?
  - Lembre-se: nos exemplos, usamos o componente Label com o elemento .
  - A resposta é:
    - Depende do componente.
    - Olhe o Javadoc.
- O Wicket normalmente não substitui o tipo de elemento:
  - Apenas modifica seus atributos e conteúdo interno.
  - Mas há de tudo... alguns modificam o elemento.
  - Tudo está nos Javadocs.
- Experimente usar um Label em um ou em um <p>.

Tipos de requisições

- No Wicket, há dois tipos de URL:
  - “Bookmarkable”
    - O usuário pode colocar no browser.
      - http://servidor/app
      - http://servidor/app/pessoa/incluir
      - http://servidor/app/pessoa/detalhar/id/21
      - http://servidor/app/pessoa/detalhar?id=21
- Não bookmarkable.
  - Geradas pelo Wicket
    - http://servidor/app/?wicket:interface=:2:link::ILinkListener::

URLs Bookmarkable

- URLs do tipo bookmarkable são previamente configuradas pelo programador.
  - “Montadas” no Application.init() – cada uma
- Mas há uma sempre configurada: a home page.
  - Application.getHomePage().
- São os “pontos de entrada” do usuário na aplicação.
- Quando recebe uma URL assim:
  1. Instancia a página
  2. Executa seus métodos de ciclo de vida
  3. Devolve ao usuário
  4. E mantém a página em memória

URLs Bookmarkable

- Para serem bookmarkable, páginas devem ter um ou ambos os construtores:
  - public MinhaPagina()
    - Construtor sem parâmetros
  - public MinhaPagina(PageParameters)
    - Forma de receber parâmetros da URL
- Se ambos forem implementados, o Wicket executa o com PageParameters

URLs não Bookmarkable

- São geradas pelos componentes do Wicket
  - Ex: Link
- Usuário não digita no browser diretamente
- Quando recebe uma requisição assim:
  1. Decodifica a URL e a partir disso:
  2. Encontra na memória a página que contém o componente que irá respondê-la.
  3. Encontra na página o componente que irá respondê-la.
  4. Executa os eventos do componente (Link.onClick).
  5. Renderiza a página.
  6. Mantém a página em memória.

O último participante do quarteto – Model!

- Observações:
  - Talvez o conceito mais complicado do Wicket.
    - Apenas mais complicado que os anteriores.
  - É muito importante compreendê-lo bem.

- Cada componente precisa de uma “fonte de dados”:
  - Um Label precisa de um texto.
  - Um TextField (<input type=”text”>) precisa de um local para ler o seu valor e atualizá-lo quando o usuário submeter o formulário.

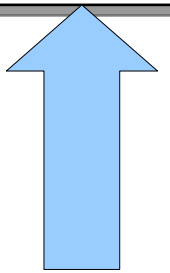
Model – cont.

- Model é o elemento que faz a ligação de um componente com a sua fonte de dados.
- Model é uma classe Java.
- Cada instância de componente tem associado uma instância de um Model.
- É representado pela interface IModel. Vamos vê-la no Eclipse?

Model – cont.

- Ao criar nosso Label no exemplo anterior, usamos um atalho.

```
Label label = new Label("mensagem", "Olá, Mundo!");
// é um atalho para
Label label = new Label("mensagem", new Model("Olá, Mundo!"));
```



Model – cont.

- A classe Model é uma implementação simples de IModel. Pouco mais que:

```
public class Model implements IModel {

  private Serializable object;

  public Model(final Serializable object) {
      setObject(object);
  }

  public Object getObject() { return object; }

  public void setObject(final Serializable object) {
      this.object = object;
  }

  public void detach() { }
}
```

Models – cont.

- Models dão grande flexibilidade aos programadores.

- Exemplos:
  - models-exemplo1
  - models-exemplo2

Models estáticos e dinâmicos

- Reforçando:
  - A classe Model é um modelo estático, pois o valor está dentro dele – muda apenas pelo programador ou pelo Wicket.
  - PropertyModel é um modelo dinâmico.

- Há várias implementações de IModel. Algumas:
  - Model e PropertyModel
  - CompoundPropertyModel
  - LoadableDetachableModel
  - StringResourceModel e ResourceModel
  - crie a sua...

Formulários

- Formulários são simples composições de componentes Wicket
  - Componente principal: Form
  - Componentes para os campos. Ex:
    - TextField, RequiredTextField: <input type=”text”>
    - DropDownChoice: <select>
    - ListChoice, ListMultilpeChoice: <select size=”x”>
    - RadioChoice, RadioGroup, Radio: <input type=”radio”>
    - CheckBox, CheckGroup, Check: <input type=”checkbox”>
    - Button, ImageButton
    - outros...

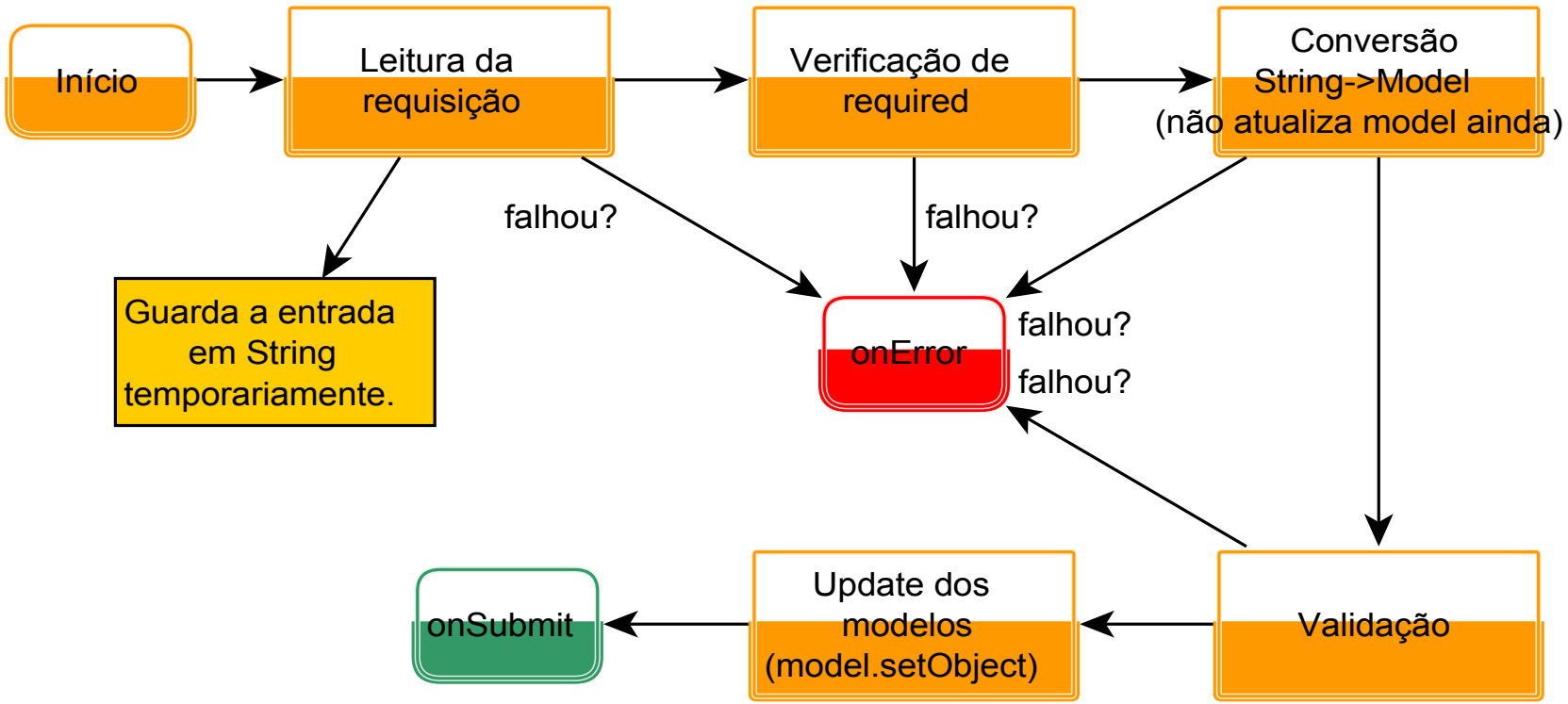
Formulários

- Melhores referências:
  - Javadocs
  - Muito bom: http://www.wicket-library.com/wicket-examples/forminput/
- Exemplos:
  - formularios-1
  - formularios-2

CompoundPropertyModel

- Repetir PropertyModel para cada campo do formulário é trabalhoso e prejudica a legibilidade/manutenibilidade.
- CompoundPropertyModel simplifica isso.
- Exemplo: formularios-3

Processamento de formulários



Conversores

- Problema:
  - HTTP e HTML são baseados em String, mas...
  - A aplicação usa outros tipos: Date, int, BigDecimal...
- A solução do Wicket: Converters
- Classes que convertem String de e para qualquer tipo.
  - Implementações de IConverter.

Converters

- Há uma implementação por tipo:
  - DateConverter, BigDecimalConverter, BooleanConverter...
- É normalmente automático.
  - O Wicket escolhe o converter a partir do que o Model retorna.
- Conversores usam Locale do browser (fica na sessão)
  - Se não existir, usa do servidor.
- Exemplo: converters-1

Converters – escolhendo um

- E quando os conversores do Wicket não são suficientes?
- É possível trocar e incluir conversores
- Duas formas:
  - Geral: troca ou inclui um converter associado a um tipo – para todo o sistema.
  - Por componente

Converters – escolhendo um

- Geral:
  - Sobrescrever Application.newConverterLocator()
  - Estender ConverterLocator ou apenas instanciá-lo e configurá-lo
- Por componente
  - Estender método getConverter(Class) do componente.
- Exemplo: converters-2

Resource Bundles e Wicket – parte 1

- O Wicket oferece boa flexibilidade para internacionalização.
- Uma das formas é por arquivos de propriedades.
- Há vários escopos de arquivos de propriedades.

Resource Bundles e Wicket – parte 1

- Quando busca uma chave, faz na seguinte ordem:
  1. <pagina>.properties
  2. <superpaginas>.properties ...
  3. <aplicacao>.properties
  4. <superclasses da aplicacao>.properties

- Exemplo:
  1. HomePage.properties
  2. BasePage.properties
  3. HelloApplication.properties
  4. BaseApplication.properties

Resource Bundles e Wicket – parte 1

- Arquivos de propriedades devem estar juntos das classes a que correspondem.
- Há muito mais sobre internacionalização – veremos mais ao final do curso.

Erros de conversão – mensagens

- Como alterar as mensagens de erros de conversão?

- Arquivos de propriedades

Erros de conversão – mensagens

- Chaves – três formas:

	Forma	Exemplo
1	`IConverter.<tipo>`	`IConverter.Date`
2	`<id>.IConverter.<tipo>`	`data.IConverter.Date`
3	`<id composto>.IConverter.<tipo>`	`form.data.IConverter.Date`

- Pode ser na página ou na aplicação.
  - Forma 1 mais comum na aplicação
  - Formas 2 e 3 mais comuns na página
- Exemplo: converters-properties-1

Mensagens – e o nome do campo?

- Mensagens podem ter nome do campo através de substituição de expressões.
  - IConverter.Date=Campo ${label} deve ser uma data válida.
- Wicket não sabe automaticamente.
- Duas formas:
  - Arquivo de propriedades
  - Método setLabel() do componente.

Mensagens – e o nome do campo?

- Arquivo de propriedades.
  - Chave:
    - <id completo>=Nome do campo
    - form.campoData=Data
  - Use no arquivo de propriedades da página apenas
  - Exemplo: labels-1

- Método setLabel
  - componente.setLabel(new Model("Nome do campo"));
  - Exemplo: labels-2

Mensagens – e o nome do campo?

- Como escolher a estratégia?

- Tem internacionalização?
  - Se positivo: properties
  - Se negativo: preferência setLabel.

Validação

- O Wicket tem um bom suporte a validação.
- Basta incluir em código:
  - componente.add(IValidator)
- Possível adicionar vários validadores a um componente.
- Vários validadores prontos.
  - Mas... é comum implementar outros.

Validação

- Validadores e *factories* prontos:
  - StringValidator.exactLength
  - StringValidator.maximumLength
  - ...
  - PatternValidator.xxxxx
  - NumberValidator.xxxxx
  - EmailAddressValidator
  - DateValidator
  - UrlValidator

Validação

- Uma validação especial: Required
  - Habilitada por componente.setRequired(boolean)
  - Para TextField, há o RequiredTextField
- Exemplo: validacao-1

Criando um validador

- Basta implementar IValidator.
- Tem apenas um método: void validate(IValidatable validatable)
- O melhor é sobrescrever a classe AbstractValidator.
  - Método onValidate.

Validação - mensagens

- Arquivos de propriedades

- Chaves:
  - Dependem o método resourceKey() do validador.
  - Para os built-in, veja a tabela:

Validação - mensagens

Validador	Forma para instanciar	Chave
NumberValidator (*enclosing* class)		
RangeValidator	NumberValidator.range()	NumberValidator.range
MinimumValidator	NumberValidator.minimum()	NumberValidator.minimum
...		
StringValidator (*enclosing* class)		
ExactLengthValidator	StringValidator.exactLength()	StringValidator.exactLength
...		
Outros (sem *enclosing* class)		
PatternValidator	new PatternValidator()	PatternValidator
...		
Especiais		
Required	componente.setRequired RequiredTextField	Required

Validação - mensagens

- Para os validadores personalizados, a chave default é o nome da classe.
- A chave no arquivo de propriedades pode ser:
  - Direta. Ex:
    - NumberValidator.range=O valor deve estar entre ${minimum} e ${maximum}
    - PatternValidator
  - Por componente. Ex:
    - form.nome.NumberValidator.range=O valor de X deve...

Validação - mensagens

- A maior parte tem os seguintes parâmetros:
  - ${label}: label do componente
  - ${name}: nome do componente
  - ${input}: valor
- Alguns tem:
  - ${maximum}, ${minimum}, etc
- Veja os outros parâmetros no Javadoc dos validadores ou no método variablesMap() do validador.
- Exemplo: validacao-2

Cancelando processamento padrão

- Algumas vezes é necessário fazer submit sem converter ou validar tudo. Ex:
  - Uma combo que é usada para preencher as opções de outra.
  - Telas com abas
- Método setDefaultFormProcessing(boolean)
- Nos componentes:
  - Button
  - SubmitLink
- Sem exemplos: fica como referência

“Mensagens rápidas”

- É possível incluir mensagens rápidas diretamente na página.
- Métodos de Component:
  - fatal, error, info, warn, debug
- Vão para o FeedbackPanel como as outras mensagens.
- Mais usados:
  - info: Para mensagens de sucesso.
  - error: Erros.
- São eliminadas na próxima renderização da página.
- Exemplo: flash-messages

Repetidores

- Wicket oferece alguns componentes que repetem conteúdo do template.
- Os mais importantes:
  - ListView
  - DataView
  - RepeatingView
  - Loop
- ListView
  - Muito usado
  - Relativamente simples
  - Limitação: permite apenas List
  - Exemplo: listview-1

Repetidores

- Loop
  - Repete por um número de vezes predeterminado
  - Não há “fonte de dados”, apenas um número
  - Como se fosse um “for”
- RepeatingView:
  - Repete o número de vezes que houver filhos
  - Os filhos são adicionados com add simples
  - Ver Javadoc
- DataView: veremos mais adiante

LoadableDetachableModel – problema

- Wicket é stateful
- Guarda todas as páginas em memória e em disco
- Mas e as referências contidas nelas?
  - Guarda também

LoadableDetachableModel – problema

- Referências: use-as com cuidado.
  - Não guarde referências a singletons (DAOs e outros objetos que forneçam serviços)
  - A não ser que exista um mecanismo para evitar a serialização delas
  - Mas e as que são usadas pelo Wicket nos models?
    - Entidades de negócio
    - Resultados de pesquisa

LoadableDetachableModel – ciclo de requisição

- O Wicket tem um mecanismo para “desatracar” referências
- Ciclo de requisição:
  1. Recebe requisição
  2. Decodifica a requisição
  3. Encontra a página e o componente que irão responder
  4. Executa os eventos
  5. Renderiza a página
  6. Excuta o método detach() de:
     1. Todos os componentes das páginas envolvidas na requisição
     2. Todos os Models das páginas
     3. Sessão
  7. Mantém as páginas em memória e/ou em disco

LoadableDetachableModel – ciclo de requisição

- O “detach” é um método normal.
  - Você pode sobrescrevê-lo.

- Serve para se desfazer de tudo o que não é mais necessário ou que pode ser obtido novamente depois:
  - Principalmente referências

LoadableDetachableModel – um caso comum

- Um caso comum:
  - Há uma página de consulta, que mostra uma lista de acordo com alguns critérios.
  - Essa lista é renderizada por um ListView
  - A lista então acaba ficando dentro do ListView
  - Mas... não precisaria, pois a consulta deve ser refeita no caso de refresh ou novo submit
- Veja o exemplo listview-2
  - Um caso muito semelhante

LoadableDetachableModel – solução

- E se o método getObject do model buscasse no banco de dados a lista e depois, quando o Wicket não precisasse mais, a descartasse?

- É isso que LoadableDetachableModel faz
  - Internamente, descarta a referência no método detach

- Exemplos: loadabledetachablemodel-1 e 2

- Outro motivo:
  - Garantir a sincronia com o banco de dados

LoadableDetachableModel e form

- Uma tela simples de consulta:
  - Exemplo: loadabledetachablemodel-3

DataView

- Poderoso para tabelas de dados
- Permite ordenação e paginação
- Um pouco mais complicado...
- Exemplos:
  - dataview-1: simples, sem paginação e ordenação
  - dataview-2: com paginação
  - dataview-3: com paginação e ordenação
- Parece muito complicado no início
- Você vai se acostumar
- Crie extensões para te ajudar

Herança de páginas

- O Wicket permite herança de páginas.
- Uso de tags especiais nos templates:
  - <wicket:extend>
  - <wicket:child>
- Exemplo: heranca-1

Painéis

- Até agora estendemos componentes simples
  - Link, Form, ...
- Como criar componentes reusáveis complexos, com vários componentes internos?
- Painéis:
  - Componentes com template que podem ser incluídos em qualquer página.
  - Forma de construção muito parecida com a de páginas.
  - Diferença: uso de <wicket:panel>
- Exemplo: panel-1

Mais tags Wicket

- Já vimos que o Wicket tem algumas tags especiais para templates:
  - <wicket:panel>, <wicket:child> e <wicket:extend>
- Há mais: http://cwiki.apache.org/WICKET/wickets-xhtml-tags.html
- Veremos algumas delas
- <wicket:remove>
  - O Wicket irá remover o conteúdo que estiver entre as tags.
  - Usado para facilitar integração com web designer OU
  - Permitir renderização do template direto no browser.

Mais tags Wicket

- <wicket:head>
  - Faz o trecho interno ser incluído no <head> da página
  - Forma de uma subclasse de página ou um painel contribuir para o <head> da página final
  - Muito útil para importar Javascripts e CSSs
- <wicket:enclosure>
  - Esconde um trecho de HTML quando um componente interno estiver invisível
  - Poupa um WebMarkupContainer
- Exemplo: tags-1

Escopos de aplicação e sessão

- Como guardar informações com escopo diferente de página?
- Escopo de aplicação:
  - Simples: no seu Application
- Escopo de sessão:
  - O Wicket cria um objeto derivado de WebSession por sessão de usuário
  - Criado por WebApplication.newSession.
  - Basta estendê-lo e retornar sua própria subclasse de WebSession.
- Vantagens sobre Sessão servlet:
  - Propriedades bem definidas
  - Tipos!

Escopos de aplicação e sessão

- Sessions podem ser temporárias
  - Se o Wicket perceber que não há páginas stateful, pode descartar a sessão recém criada.
  - Para não fechar, use o método Session.bind().

- Exemplo: session-1

Thread safety

- Páginas são thread-safe
  - Nunca há mais de uma requisição sendo tratada ao mesmo tempo por uma mesma instância
- Application e Session não são!
- Cuidados:
  - Use métodos synchronized para getters e setters de aplicação e sessão quando possam ser acessados dentro de uma requisição.
    - Cuidado especial com coleções.
  - Isso não é exclusivo do Wicket: com servets também há o problema! Você toma esse cuidado hoje?
  - Evite manter referências a componentes em outros locais que não as páginas onde estão
  - Não compartilhe componentes entre páginas

Passando parâmetros para uma página

- Como passar parâmetros para uma página?
  - Ex: uma página de detalhamento de entidade precisa da instância ou identificação da entidade

- Duas formas:
  - Não bookmarkable
  - Bookmarkable

Passando parâmetros para uma página

- Não bookmarkable:
  - Se:
    - Páginas em Wicket são classes Java
    - Páginas podem ser instanciadas diretamente pelo seu construtor
    - Construtores podem ter parâmetros
    - Classes podem ainda ter outros métodos que alteram seu estado
  - Então...:
    - PaginaDetalhes detalhes = new PaginaDetalhes(entidade);
    - setResponsePage(detalhes);

Passando parâmetros para uma página

- Bookmarkable:
  - Relembrando:
    - Páginas bookmarkable podem ser acessadas diretamente pela URL
    - Devem ser previamente “montadas” no init do Application
  - Páginas bookmarkable aceitam parâmetros
  - Basta implementar um construtor com a assinatura:
    - public MinhaPagina(PageParameters)

Passando parâmetros para uma página

- Páginas podem ser bookmarkable e não bookmarkable ao mesmo tempo.
  - Depende do construtor usado.
- Exemplo: pagina-com-parametros

Behaviors

- Algumas vezes precisamos mudar atributos de um elemento HTML dinamicamente
  - Ex: “style” em uma tabela de dados – para implementar “zebra”
- Podemos estender o componente e seu método onComponentTag
- Ou podemos usar Behaviors!

Behaviors

- São associados aos componentes através do método Component.add(IBehavior).
- Componentes encadeiam alguns dos seus métodos de ciclo de vida aos behaviors associados a eles.
- Como o nome diz:
  - Acrescentam um comportamento extra ao componente
- Behavior interessante:
  - AtributeModifier

- Muito usado com Ajax no Wicket.

Wicket e Ajax

- Wicket tem suporte nativo a Ajax
- Várias formas de usar
  - Há componentes para uso específico com Ajax
  - Há behaviors que “plugam” Ajax em componentes regulares
- Assunto extenso...
  - Já estão capacitados a procurar (e a perguntar)

Resource Bundles e Wicket – parte 2

- Qualquer componente pode ter seu arquivo properties
  - Só precisa ter o mesmo nome mas com extensão properties
- Ordem de busca:
  1. Properties da página que contém o componente
  2. Properties das superclasses da página
  3. Properties do componente.
  4. Properties das superclasses do componente.
  5. Properties da aplicação.
  6. Properties das superclasses da aplicação.

Resource Bundles e Wicket – parte 2

- Formas de se obter uma mensagem:
  - Método getString
    - componente.getString(“chave”)
  - Localizer
    - componente.getLocalizer().getString(...)
    - Oferece mais opções que componente.getString().
    - O Localizer é único **por aplicativo** (mesmo que seja obtido pelo componente).
- Lembre-se:
  - Page também é um componente.
- Tudo isso vale para mensagens de validação e erros de conversão.

Modelos internacionais

- Veremos mais dois models:
  - ResourceModel
  - StringResourceModel
- Apenas leitura
- Pegam mensagens de resource bundles
- ResourceModel
  - simples, sem substituições, expressões
- StringResourceModel:
  - mais features
  - veja Javadoc.
- Exemplo: resource-models

Internacionalização direto no template

- Tag especial:
  - <wicket:message>
  - Substitui o trecho interno pelo conteúdo obtido a partir da chave

```
<span>
<wicket:message key="chave">Texto</wicket:message>
</span>
```

```
texto=Novo Texto
```

```
<span>
Novo Texto
</span>
```

Internacionalização direto no template

- Atributo wicket:message
  - Substitui atributos pelo conteúdo obtido através da chave

```
<input type="submit" wicket:message="value:pagina.busca"/>
```

```
pagina.busca=Buscar
```

```
<input type="submit" value="Buscar" />
```

- Use vírgula para incluir vários atributos:

```
<img src="img.gif" wicket:message=
  "alt:pagina.img.alt,title=pagina.img.titulo"/>
```

Templates internacionais

- O Wicket também pode escolher templates de acordo com o Locale do usuário.
- O Wicket busca os templates como busca arquivos de propriedades.
- Ex: se o Locale for pt_BR:
  1. HomePage_pt_BR.html
  2. HomePage_pt.html
  3. HomePage.html

Mais referências

- Visão (“filosofia”): http://wicket.apache.org/vision.html
- Livro: “Enjoying Web Development with Wicket”: http://www.agileskills2.org/EWDW/
  - Wicket 1.3
- Livro: “Pro Wicket”: http://www.apress.com/book/view/1590597222
  - Wicket 1.2 (pré-Apache)

Mais referências – alguns artigos

- A Year of Wicket
  - Não é novo, mas comenta sobre o uso do Wicket em um produto da IBM.
  - Veja alguns screenshots do produto aqui.
- Tapestry and Wicket compared
  - Artigo na IBM. Não deixe de ler a seção “Conclusion”.
- Seam / JSF vs Wicket: performance comparison