Extracted from:

Grails 2: A Quick-Start Guide

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Grails 2

A Quick-Start Guide

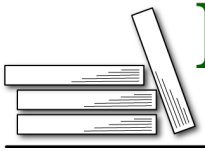
Dave Klein
Ben Klein

GRAILS

Grails 2: A Quick-Start Guide

Dave Klein
Ben Klein

The Pragmatic Bookshelf
Dallas, Texas • Raleigh, North Carolina



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Susannah Davidson Pfalzer (project manager)
Potomac Indexing, LLC (indexer)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-937785-77-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—December 2013

CHAPTER 6

Getting Things Done

In this iteration, we're going to take advantage of the generated scaffolding code to make our views more pleasing to our customer. We'll also work on implementing the task list features of TekDays so that TekDays users can get things done. Along the way, we'll learn just how easy it is to modify and extend Grails views.

Changing All Our Views at Once

We saw in the previous chapter how Grails uses SiteMesh to provide a consistent look throughout an application. That's what's been giving us that cool Grails logo on all of our views. But that's not quite what our customer wants for TekDays. Let's see what we can do about that. Open TekDays/grails-app/views/layout/main.gsp, and in the <body> section, modify the first :

things.2/TekDays/grails-app/views/layouts/main.gsp

```
<div id="logo" role="banner"><a href="${createLink(uri: '/')}">
  <img src="${resource(dir: 'images', file: 'td_logo.png')}"
    alt="TekDays"/></a></div>
```

We renamed the , changed the link to point to the TekDays home page, and replaced the logo. Of course, you can substitute your own logo design, or you can download td_logo from the book's website. Talk about low-hanging fruit! Our new logo will now show up on every page of our application. In Figure 25, *TekDays home page with new logo*, on page 6, we get a peek at what our pages will look like.

That's not all we can do in this file, but it's all we need to do for now. We could go on and add sidebars, a footer, a standard menu, and so on. But we don't want to get ahead of ourselves.

Let's look at another file that is shared across all the views in our application. Grails puts the CSS code for the scaffolded views in web-app/css/main.css. We can

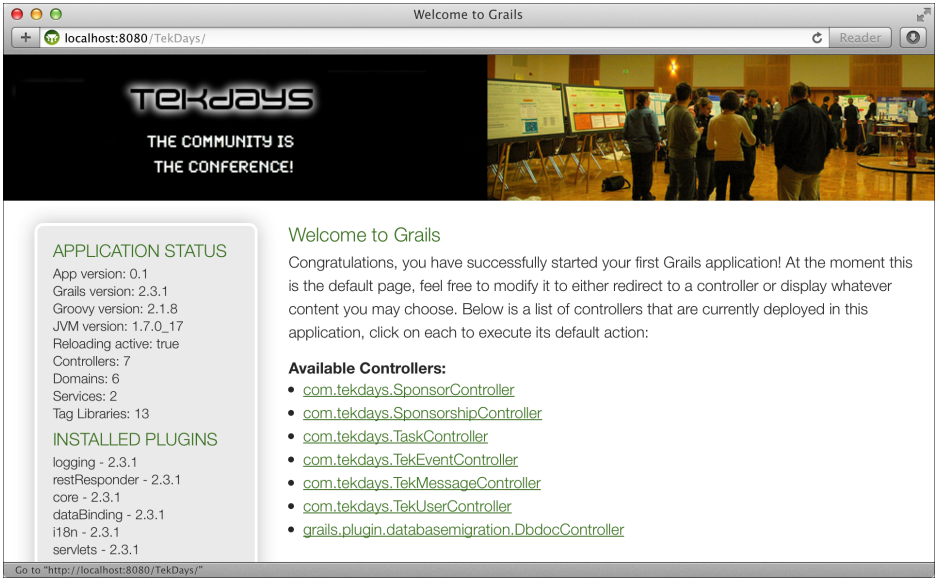


Figure 25—TekDays home page with new logo

change many aspects of our views by modifying this file. In an effort to keep style code out of our pages, we will be adding to this file for the small amount of additional CSS that we will be using in TekDays. The additional style rules that we are using can be found in Appendix 1, *Additional CSS Rules*, on page ?. Now let's turn our attention to our scaffolded views.

Modifying the Scaffolded Views

We're going to go through the scaffolded view code of the TekEvent class and make some simple modifications. (The things we change here can just as easily be done for the other classes' views.) As we go through these changes, we can leave the application running and immediately see the changes by simply refreshing the browser—another example of how Grails keeps that rapid feedback loop going. This also takes the pain out of the tweaking process that we so often have to go through to get a page "just right."

The List View

We'll start by removing the organizer from the list view. When we generated the scaffolded views, Grails simply used the first six properties of our TekEvent for the table in the list view, and this included organizer. But the organizer is not really something users will be concerned with as they look through a list of events.

Let's see what this looks like. Open TekDays/grails-app/views/tekEvent/index.gsp, and go to the <thead> block. Remove the Organizer column. You should be left with this:

things.2/TekDays/grails-app/views/tekEvent/index.gsp

```
<thead>
    <tr>
      <g:sortableColumn property="name"
        title="${message(code: 'tekEvent.name.label',
        default: 'Name')}" />
      <g:sortableColumn property="city"
        title="${message(code: 'tekEvent.city.label',
        default: 'City')}" />
      <g:sortableColumn property="description"
        title="${message(code: 'tekEvent.description.label',
        default: 'Description')}" />
      <g:sortableColumn property="venue"
        title="${message(code: 'tekEvent.venue.label',
        default: 'Venue')}" />
      <g:sortableColumn property="startDate"
        title="${message(code: 'tekEvent.startDate.label',
        default: 'Start Date')}" />
    </tr>
</thead>
```

Next, remove the associated <td> from the <tbody>. It should look like this:

things.2/TekDays/grails-app/views/tekEvent/index.gsp

```
<tbody>
<g:each in="${tekEventInstanceList}" status="i"
  var="tekEventInstance">
  <tr class="${(i % 2) == 0 ? 'even' : 'odd'}">
    <td><g:link action="show"
      id="${tekEventInstance.id}">
      ${fieldValue(bean: tekEventInstance,
        field: "name")}</g:link>
    </td>
    <td>${fieldValue(bean: tekEventInstance,
      field: "city")}</td>
    <td>${fieldValue(bean: tekEventInstance,
      field: "description")}
    </td>
    <td>${fieldValue(bean: tekEventInstance,
      field: "venue")}</td>
    <td><g:formatDate
      date="${tekEventInstance.startDate}" />
    </td>
  </tr>
</g:each>
</tbody>
```

Now when we refresh the page, it will look like the following figure.

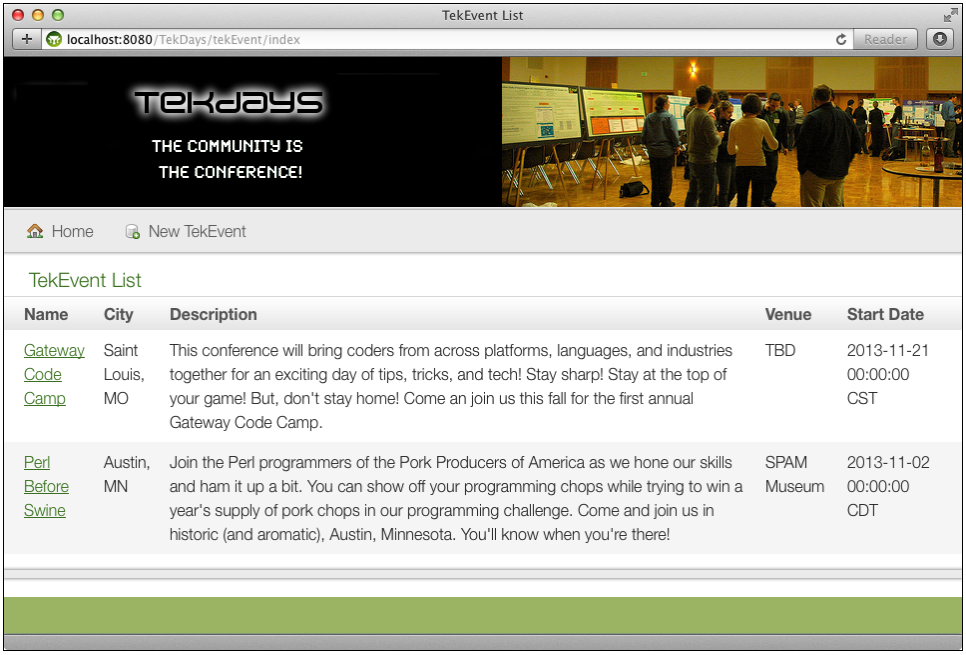


Figure 26—Modified TekEvent list view

The Show View

The show view presents several opportunities for improvement. We'll go through from top to bottom and fix things up. You can save the file and refresh after each step, and we'll show the new view when we're done. Open TekDays/grails-app/views/tekEvent/show.gsp, and let's get started.

Near the top of this file you'll see an <h1> containing a <g:message> tag that will render the text "Show TekEvent." Let's replace that tag with the name of the event:

things.2/TekDays/grails-app/views/tekEvent/show.gsp

```
<h1>${tekEventInstance?.name}</h1>
```

This is an example of how we can use Groovy expressions anywhere on a page.

Next, let's remove the name property from the main part of the page, because we already have it displayed in the heading. Notice that each property is displayed within a <li> tag surrounded by <g:if> tags; just remove the opening

<g:if>, the closing </g:if>, and everything in between. You can repeat this process for any other properties you want to remove.

We'll move description to right above city. We'll move the organizer down directly before the volunteers. Then we'll do something a little more clever for the city property:

things.2/TekDays/grails-app/views/tekEvent/show.gsp

```
<g:if test="${tekEventInstance?.city}">
<li class="fieldcontain">
  <span id="city-label" class="property-label">
    Location
  </span>

  <span class="property-value" aria-labelledby="city-label">
    <g:fieldValue bean="${tekEventInstance}" field="venue"/>,
    <g:fieldValue bean="${tekEventInstance}" field="city"/>
  </span>

</li>
</g:if>
```

We changed the label, which is the value in the first , to "Location," and we included the venue in the same line.

Next we'll tackle the date properties. The way they're currently being displayed is not going to cut it. Sure, people will want their events to run on schedule, but they're probably not going to worry about the exact hour, minute, and second that it starts.

things.2/TekDays/grails-app/views/tekEvent/show.gsp

```
<g:if test="${tekEventInstance?.startDate}">
<li class="fieldcontain">
  <span id="startDate-label" class="property-label"><g:message
    code="tekEvent.startDate.label" default="Start Date" /></span>

  <span class="property-value" aria-labelledby="startDate-label">
    <g:formatDate format="MMMM dd, yyyy"
      date="${tekEventInstance?.startDate}" /></span>

</li>
</g:if>
```

Here we added the format attribute[1] to the <g:formatDate> tag used for startDate. Do the same with the endDate property.

1. See http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html.

things.2/TekDays/grails-app/views/tekEvent/show.gsp

```
<g:if test="${tekEventInstance?.endDate}">
<li class="fieldcontain">
  <span id="endDate-label" class="property-label"><g:message
    code="tekEvent.endDate.label" default="End Date" /></span>

  <span class="property-value" aria-labelledby="endDate-label">
    <g:formatDate format="MMMM dd, yyyy"
      date="${tekEventInstance?.endDate}" /></span>

</li>
</g:if>
```

Finally, let's clean up the way the Sponsorship collection is displayed. Recall from the discussion in *Many-to-Many Relationships*, on page ?, that we did not declare a toString() method because the way we display a Sponsorship will depend on the context. That's why it currently shows up as com.tekdays.Sponsorship : 1. Since we're working on the TekEvent views, we'll modify the display with that context in mind.

things.2/TekDays/grails-app/views/tekEvent/show.gsp

```
<g:if test="${tekEventInstance?.sponsorships}">
<li class="fieldcontain">
  <span id="sponsorships-label" class="property-label"><g:message
    code="tekEvent.sponsorships.label" default="Sponsorships" /></span>

    <g:each in="${tekEventInstance.sponsorships}" var="s">
    <span class="property-value" aria-labelledby="sponsorships-label">
      <g:link controller="sponsorship" action="show"
        id="${s.id}">${s?.sponsor.encodeAsHTML()}</g:link></span>
    </g:each>

</li>
</g:if>
```

All we had to do was change the s?.encodeAsHTML() to s?.sponsor.encodeAsHTML(). If this was the Sponsor show view, we would change it to s?.event.encodeAsHTML(). Take a look at Figure 27, *Modified TekEvent show view*, on page 11 to see the results of our changes.

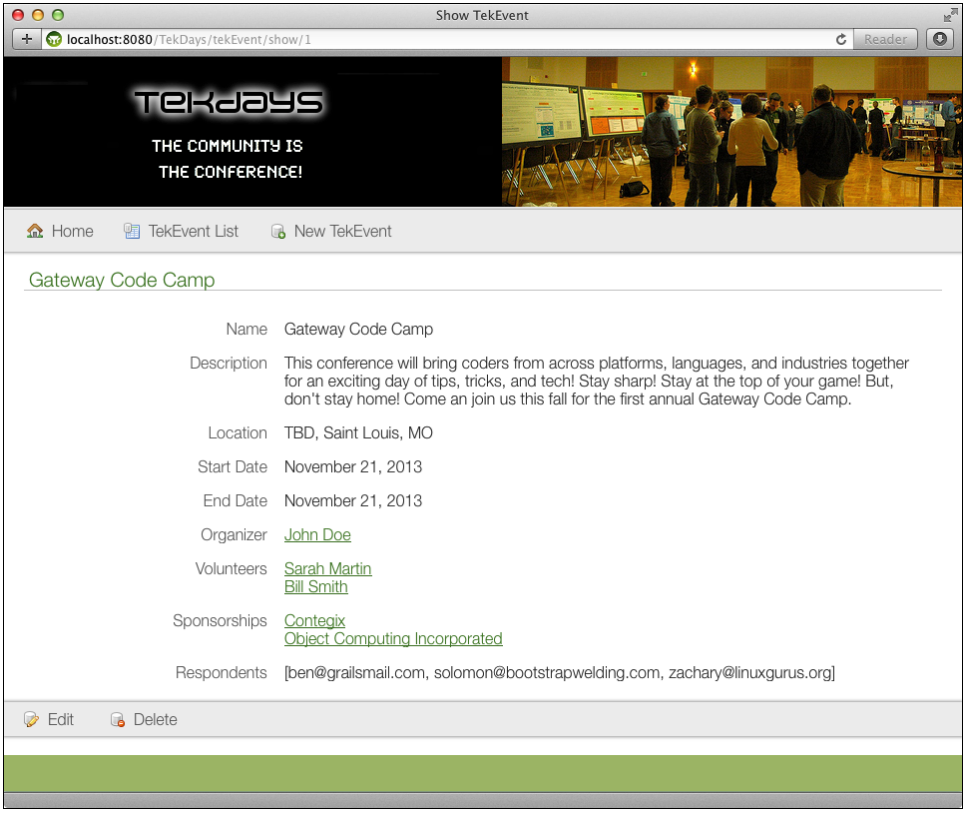


Figure 27—Modified TekEvent show view