Aqui está a síntese definitiva para você dominar a montagem de páginas no Grails (Sitemesh). Vamos dividir em três categorias lógicas: **Estrutura Principal**, **Injeção Dinâmica** e **Reutilização**.

1. O Pilar Central: `<g:layoutBody/>`

É a tag mais importante do sistema de layouts. Ela define onde o conteúdo "bruto" da sua view vai aparecer.

- **Onde usar:** Apenas no **Layout** (`main.gsp`).
- **Quem preenche:** O conteúdo que está dentro do `<body>` da sua **View** (`index.gsp`).

Característica	Detalhes
Vantagem	**Automático.** Você não precisa configurar nomes ou IDs. É o comportamento padrão do Grails.
Desvantagem	**Único.** Você geralmente só tem um `layoutBody` por página. Ele não serve para colocar coisas em lugares espalhados (como na sidebar e no rodapé ao mesmo tempo).

Exportar para as Planilhas

2. A Dupla Dinâmica: `<content tag="...">` e `<g:pageProperty/>`

Essas duas **sempre trabalham juntas**. Elas servem para abrir "portais" ou "gavetas" específicas no seu layout para que a view possa preencher com conteúdos pontuais.

A. O Remetente: `<content tag="nome">`

- **Onde usar:** Na **View** (`index.gsp`).
- **Função:** Empacota um pedaço de HTML e dá um nome a ele (ex: "botoesTopo"), mas **não mostra nada na tela** naquele momento. Ele apenas guarda na memória.

B. O Destinatário: `<g:pageProperty name="page.nome"/>`

- **Onde usar:** No **Layout** (`main.gsp`).
- **Função:** Busca o pacote que foi guardado na memória e imprime na tela naquele local exato.

Característica	Detalhes
Vantagem	**Cirúrgico.** Permite que a View (`index.gsp`) controle partes distantes do Layout (como mudar o título da aba do navegador, adicionar um botão na sidebar ou um script JS no final do body).

Desvantagem	**Acoplamento.** Exige que você lembre o nome exato da tag (`name="page.xyz"`) nos dois arquivos. Se errar uma letra, o conteúdo não aparece e não gera erro.

3. O Carimbo: `<g:render template="...">`

É a ferramenta de modularização (DRY - Don't Repeat Yourself).

- **Onde usar:** Em **qualquer lugar** (Views, Layouts ou dentro de outros Templates).
- **Regra de Ouro:** O arquivo do template deve começar com underline `_` (ex: `_formulario.gsp`), mas na tag você chama sem o underline.

Característica	Detalhes
Vantagem	**Organização e Reúso.** Se você tem um menu, um formulário ou um card que se repete em 10 páginas, você cria um arquivo só. Alterou no template, altera no site todo.
Desvantagem	**Contexto.** O template não "vê" as variáveis da página principal automaticamente, a menos que você as passe explicitamente (usando `model="[var: valor]"`).

Quadro Comparativo Final

Cenário	Qual usar?	Por quê?
O "miolo" da página	`<g:layoutBody/>`	É a estrutura padrão do framework.
Um menu lateral fixo	`<g:render/>`	É um componente estático que se repete em tudo.
Botões que mudam por página	`<content>` + `<pageProperty>`	O local é fixo (layout), mas o conteúdo varia (view).
Um formulário de contato	`<g:render/>`	Você pode querer usar esse form na página "Contato" e no rodapé da "Home".
Scripts CSS/JS específicos	`<content>` + `<pageProperty>`	Para injetar um `.css` apenas na página que precisa dele.

Dica de Ouro de Arquitetura

Para manter sua aplicação Grails saudável:

1. Use **Layouts** (`layoutBody`) para a estrutura macro (esqueleto).

2. Use **Templates** (`render`) para quebrar o código do Layout em pedaços pequenos (`_header.gsp`, `_sidebar.gsp`, `_footer.gsp`).
3. Use **PageProperties** (`content`) apenas para exceções e personalizações finas que a View precisa enviar para o Layout.

Exemplos de uso

Aqui estão exemplos práticos simulando o seu sistema de **Contatos** para você ver a diferença real de uso.

Vou usar um diagrama visual para ajudar a entender como essas peças se encaixam e depois mostro o código.

1. O Padrão: `<g:layoutBody/>`

Cenário: Exibir a lista de contatos na tela principal e o formulário na tela de criação.

Arquivo: `grails-app/views/layouts/main.gsp` (O Pai)
HTML
```
<main class="col-md-9 ml-sm-auto col-lg-10 px-4">

    <g:layoutBody/>

</main>
```

Arquivo: `grails-app/views/contact/index.gsp` (O Filho)
HTML
```
<html>

<head><meta name="layout" content="main"/></head>

<body>

    <h1>Meus Contatos</h1>

    <table>

        <tr><td>João</td><td>1234-5678</td></tr>

        <tr><td>Maria</td><td>9876-5432</td></tr>

    </table>

</body>

</html>
```

2. A Injeção: `<content>` e `<g:pageProperty/>`

Cenário: Você quer que cada página tenha botões diferentes no canto superior direito do cabeçalho (Header).

- Na página **Index**: Botão "Novo Contato".
- Na página **Show**: Botão "Editar" e "Deletar".

Arquivo: `grails-app/views/layouts/main.gsp` (O Pai)
HTML

<h1 class="h2">Dashboard</h1>

<g:pageProperty name="page.botoesTopo" default=""/>

Arquivo: `grails-app/views/contact/index.gsp` (A Listagem)
HTML
<body>

<content tag="botoesTopo">

<g:link action="create" class="btn btn-primary">Criar Novo +</g:link>

</content>

</body>

Arquivo: `grails-app/views/contact/show.gsp` (A Visualização de um contato)
HTML
<body>

<content tag="botoesTopo">

<g:link action="edit" id="${contact.id}" class="btn btn-warning">Editar</g:link>

<g:link action="delete" id="${contact.id}" class="btn btn-danger">Excluir</g:link>

</content>

</body>

3. O Carimbo: `<g:render template="...">`

Cenário: Você tem um "Card de Contato" bonitinho (foto, nome e telefone) que você quer exibir na **Home**, na **Lista de Contatos** e na **Busca**.

Arquivo: `grails-app/views/shared/_contactCard.gsp` (O Molde/Template) *Nota: Não tem html, head ou body aqui. Só o pedaço.*
HTML
```
<div class="card" style="width: 18rem;">

    <div class="card-body">

        <h5 class="card-title">${pessoa.name}</h5>

        <p class="card-text">${pessoa.email}</p>

        <a href="#" class="btn btn-primary">Ligar</a>

    </div>

</div>
```

Arquivo: `grails-app/views/contact/index.gsp` (Onde usamos)
HTML
```
<body>

    <h1>Lista de Cards</h1>



    <div class="row">

        <g:each in="${contactList}" var="contatoAtual">



            <div class="col-md-4">

                <g:render template="/shared/contactCard" model="[pessoa: contatoAtual]"/>

            </div>



        </g:each>

    </div>

</body>
```

Resumo Visual

1. **LayoutBody:** É o container principal (o caminhão de mudança).

2. **PageProperty:** São caixas etiquetadas (Caixa "Cozinha", Caixa "Quarto") que você coloca em lugares específicos do caminhão.
3. **Render:** É a caixa de ferramentas que você tem igualzinha em várias casas diferentes.