02

Explorando Views e perspectivas

Vimos até agora várias maneiras de tornar o editor do Eclipse mais produtivo, com diversos atalhos e templates. Mas o Eclipse não para por aí. Além dessas, ele possui várias caixas de conteúdo, chamadas Views, que mostram certos aspectos do nosso projeto: uma classe, a árvore de diretórios, pendências, etc.

Para começar nossa exploração, selecione a perspectiva de Java: no canto superior direito, clique no ícone e escolha a opção Java. Ou, melhor ainda, usando o ctrl + 3 Java e selecione a opção que abre a perspectiva.

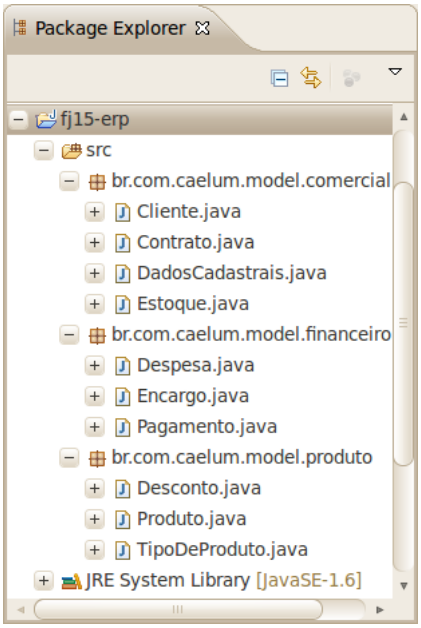
Logo de cara, o Eclipse traz algumas views já abertas. No centro, temos o editor. Ele dispensa apresentações, já que o utilizamos até agora para escrever as nossas classes. À esquerda, temos a view Package Explorer, que nos permite visualizar nossas classes Java agrupadas em pacotes e source folders.

Olhando para a estrutura do projeto é frequente encontrarmos pacotes cheios demais ou classes que deveriam estar em outro local. Mover classes causa mais trabalho do que pode parecer: é preciso alterar o package no qual ele se encontra e, assim, toda outra classe que utiliza tal tipo tem que alterar o import. Felizmente, o Eclipse ajuda muito nessa refatoração.

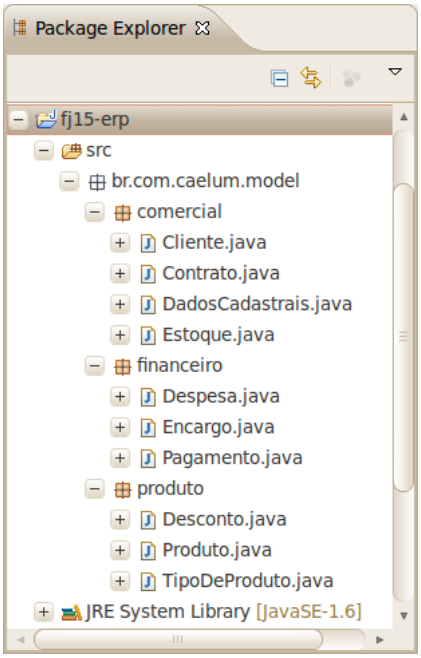
Apesar de existir um menu para fazer tal mudança (faça ctrl + 3 move), se sabemos para qual pacote uma determinada classe deve ser movida, a forma mais fácil de fazê-lo é arrastar a classe para o pacote correto. Dessa forma, precisamos localizar a classe na estrutura de diretórios e arrastá-la para o lugar certo.

O Package Explorer possui algumas configurações interessantes que nos ajudam nessa busca. Já sabemos abrir uma classe pelo nome dela com o ctrl + shift + T e, com ela aberta no editor, a view é capaz de linkar com a estrutura de diretórios. Note o símbolo das setinhas ( ): se ele estiver selecionado, a classe aberta no editor será mostrada na listagem de diretórios. Então, basta arrastá-la e soltar no pacote escolhido.

A apresentação padrão dos pacotes de um projeto é a flat, que entende cada pacote como uma pasta. Veja a figura abaixo que exemplifica a visualização flat:



Mas conforme um projeto cresce, podemos sentir a necessidade de, por exemplo, dividir as classes de modelos em sub-pacotes com menos classes em cada. Experimente mudar a apresentação de pacotes trocando com a seta para baixo (e Package Presentation) de Flat para Hierarchical:



Também é bastante comum que um projeto vá acumulando warnings conforme o desenvolvimento acontece. Alguns deles são realmente úteis: nos dizem quando um membro privado nunca é acessado e, portanto, pode ser deletado, por exemplo. Há, contudo, outros warnings que apenas avisam que deveríamos ter colocado um serial version id. O problema de deixar esses avisos (em amarelo) espalhados pelo nosso projeto é que eles sujam as classes e tornam o ctrl + . (ctrl + ponto) inútil: esse atalho busca os erros ou warnings de uma classe. Dessa forma, se houver muitos warnings bobos, serão muitas paradas até chegar no problema que realmente interessa.

Uma view muito útil para ajudar na limpeza do código é a **Problems**, na parte de baixo do Editor, a aba Problems. Ela localiza os warnings e erros do projeto e linka para o arquivo ou projeto em que acontecem. Então, aqueles que têm correção rápida aparecem com um ícone de lâmpada, ou seja, você pode usar o ctrl + 1 para corrigir. As outras abas que vêm por padrão são a Javadoc e Declaration, que mostram o Javadoc e a declaração do elemento onde o cursor está posicionado, respectivamente.

Na parte da direita temos mais duas views: Task List, que faz parte do plugin Mylyn (http://eclipse.org/mylyn/), e o Outline. O Mylyn é um gerenciador de tarefas que é capaz de se comunicar com as principais ferramentas de ALM, como o Jira, Bugzilla e Trac. O Outline, por sua vez, mostra o perfil de uma classe, isto é, os membros da classe. Nossa recomendação é fechá-las ou minimizá-las, já que essas views diminuem consideravelmente o espaço útil do editor. Para acessar as informações do Outline basta, no editor, usar o atalho ctrl + O.

É frequente também que queiramos marcar um trecho de código para revisão futura e isso pode ser feito usando alguns comentários especiais que vêm pré-configurados no Eclipse:

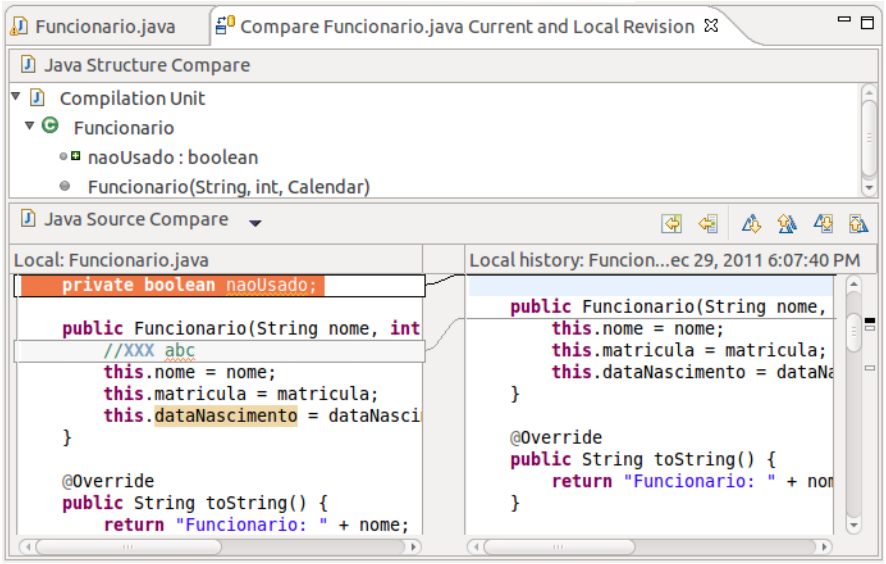
```
// TODO algo que precisa ser feito
// FIXME algum bug conhecido que precisa ser corrigido o mais rápido possível
// XXX alguma gambiarra que precisa ser refatorada
```

Existem várias outras views que não vêm abertas por padrão, mas que são úteis para determinadas operações. Uma delas é a Tasks (ctrl + 3 Tasks), que lista todas as tarefas criadas a partir de comentários especiais no código, facilitando a revisão daqueles problemas.

Tais tasks são bem flexíveis e você pode até definir seus próprios comentários especiais e suas prioridades a partir da configuração Task Tags (use o ctrl + 3!).

Ainda outro problema comum é quando acidentalmente deletamos uma classe ou, de tanto alterar um trecho de código, nos perdemos e o que mais queríamos era voltar para como aquela classe estava, digamos, meia hora atrás. Mesmo sem você pedir, o Eclipse guarda as alterações que você fez em qualquer arquivo no chamado Local History. Isso significa que você consegue visualizar o seu arquivo como ele era, por exemplo, há 3 horas atrás e pode até voltar para essa versão do arquivo, caso você tenha feito alguma alteração indesejada (e que não tenha conseguido voltar com o ctrl + Z).

Essa view também é usada se o projeto estiver sob algum Controle de Versão (http://www.caelum.com.br/curso/online/git/), como o SVN (http://www.eclipse.org/subversive/) ou o Git (http://eclipse.org/egit/). Você pode linkar o histórico com o editor aberto () e usar o editor de comparação do eclipse, selecionando duas versões, botão direito do mouse e Compare with each other, ou selecionando uma versão, botão direito e Compare current with local.



Em conjunto com as views, temos o conceito de Perspectiva. Cada perspectiva agrupa algumas views para focar numa determinada fase ou tipo de desenvolvimento de software. Com isso, temos um ambiente de trabalho mais adequado para evoluir nosso projeto com o máximo de ajuda possível da IDE.

A perspectiva Java é boa para qualquer tipo de projeto, mas o Eclipse possui perspectivas mais focadas em determinados tipos de projeto ou situações. Se você está desenvolvendo uma aplicação web, ou Java EE, pode usar a perspectiva Java EE. Esta é focada em montar as diversas partes que podem compor uma aplicação completa. Para mudar de perspectiva podemos usar o atalho ctrl + F8 (segure o ctrl e use F8 quantas vezes for necessário para chegar à opção Java EE).

Para começar a usar essa perspectiva de verdade, precisamos de um projeto mais complexo do que um Java Project, por exemplo um projeto web (http://www.caelum.com.br/curso/fj-21-java-web/). Vamos criar um usando ctrl + N e escolhendo Dynamic Web Project. Isso abrirá um wizard para configurarmos esse novo projeto. Além do nome, podemos configurar coisas importantes como o servidor onde pretendemos rodar essa aplicação, ou seja, o Target Runtime.

Como não temos nenhum runtime configurado, precisamos criá-lo usando o botão New Runtime. Podemos escolher qualquer um dos servidores disponíveis (é possível baixar adaptadores de outros servidores que não estão na lista), por exemplo o Apache Tomcat 7.0. Se você ainda não tem um Tomcat na máquina, podemos usar o botão Download and Install para que o próprio Eclipse faça o download e o descompacte. Se você já tem um, basta selecionar um Tomcat que já está instalado na máquina com o botão Browse.

Há ainda outras configurações são a versão da API de Servlet (Dynamic web module version), configurações do servidor e a possibilidade de adicionar o projeto a um EAR ou Working Set. Ao clicar em Next, podemos escolher as pastas que serão source folders e a pasta onde as classes serão compiladas. Na próxima tela configuramos o nome de contexto (Context root) e a pasta que será a raiz do conteúdo web (Content directory). Podemos também gerar o `web.xml`, que não é obrigatório, se você escolheu 3.0 como module version.

Repare que o projeto foi criado e mostrado na view Project Explorer, não no Package Explorer como na perspectiva anterior. Essa view também mostra os arquivos do projeto, mas dá foco aos elementos de configuração do projeto, como o Deployment Descriptor representando aquilo que você configura no `web.xml`.

Com o projeto criado, queremos, por uma questão de auditoria, escrever no terminal a URL de cada requisição feita para o seu software, independentemente de qual servlet responda a ela. Para interceptar todas as requisições, poderíamos passar em cada uma das nossas servlets adicionando o log de requisições, mas, mais fácil que isso, é configurar um filtro que responda para todas as urls. Para criar seu `Filter`, use o atalho de criação ctrl + N Filter. Vamos escolher o pacote `br.com.caelum.empresa.web` e o nome da classe `LoggerFilter`. Na tela seguinte, mudaremos o mapeamento para "/*", que indica que todas as requisições devem passar por esse filtro. Agora esse novo filtro já está listado em Deployment Descriptor Filters.

No método `doFilter`, vamos colocar um log simples:

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) {
    System.out.println("Recebendo uma requisição para " + request.getRequestURL());
    chain.doFilter(request, response);
}
```

Para finalizar ver se nosso filtro funciona, precisamos criar pelo menos uma JSP (ctrl + N JSP File) que chamaremos de `index.jsp` na pasta `WebContent`. Dentro do `body` vamos colocar um título:

```
<h1>Nossa aplicação web</h1>
```

E, agora, só falta testarmos! Para ver a aplicação rodando dentro do Eclipse é preciso colocá-la em um servidor. Já tinhamos configurado o Runtime do Tomcat, mas precisamos também configurar uma instância, onde podemos escolher algumas aplicações que vão rodar dentro dela. Para isso temos a view Servers, que mostra todas as instâncias de servidor configuradas no Eclipse. Botão direito nessa view > New > Server, ou ctrl + N Server, abre o wizard de configurar um novo servidor. Basta agora escolher o tipo do servidor (o Tomcat 7.0), o server runtime que configuramos anteriormente e passar para a próxima tela onde podemos incluir aplicações no servidor. Vamos escolher a fj15-web e finalizar.

Finalmente, inicie o servidor para ter a aplicação rodando (ctrl + alt + R em cima dele) e acessá-la pelo seu navegador. Uma outra opção para ver seu projeto rodando é, com foco no `index.jsp`, mandar rodá-lo no servidor com o atalho alt + shift + X R. Não é necessário configurar nada na tela que abriu. E agora vemos a JSP aberta num navegador dentro do Eclipse!

Ao iniciar o servidor, foi aberta a view Console, mostrando a saída desse servidor. É uma das views mais importantes, pois conseguimos ver o que está acontecendo na nossa aplicação. Uma das características dessa view é que ela sempre ganhará foco quando algo for impresso no console. Isso acontece mesmo que o foco esteja, por exemplo, na aba Servers, mesmo que a view esteja minimizada, ou ainda se ela estiver fechada. Esse comportamento pode ser um tanto quanto incômodo quando lidamos com aplicações em servidores porque sempre que acontece uma alteração em classe, o Tomcat precisa recarregar o projeto - e, a cada vez que ele faz isso, o Eclipse rouba o foco e o dá para o Console.

Se você não quiser ser perturbado por esse roubo de foco, podemos mudar esse comportamento usando os ícones  , que quando desmarcado evita dar foco ao Console sempre que a saída padrão (stdout) é mudada. Para ter o mesmo efeito com a saída de erro (`stderr`), use o ícone  .

Enquanto o servidor está no ar, gostaríamos de rodar a classe `TesteImportador`. Se não tivermos mudado a configuração, o Console vai deixar de mostrar a saída do Tomcat e vai trocar para a saída dessa classe. Isso acontece porque a mesma view mostra todos os consoles abertos durante a execução do Eclipse. Se você quiser continuar vendo a saída do servidor e ainda quiser ver a saída da outra classe ao mesmo tempo, pode abrir outra view de Console com o ícone  e escolhendo New Console View. É possível deixar os dois consoles visíveis arrastando um deles para outro canto da tela.

Ainda assim, você vai notar que ambos os consoles vão ter o comportamento de mostrar a saída da última execução. Agora, se queremos forçar a view a ficar sempre em uma saída específica, por exemplo o do Tomcat, basta habilitar o ícone  , chamado de Pin console, ou seja, prender a saída daquela execução ao console.

Para aplicações Java EE (mais informações no curso FJ-31 (http://www.caelum.com.br/curso/fj-31-java-ee-web-services/)), podemos criar um projeto usando alt + shift + N e então Enterprise Application Project. Como fizemos no projeto web, precisamos do runtime de um servidor para rodar esse projeto - só que dessa vez o TomCat apenas não basta! Para trabalhar com outras especificações que não as de Servlets e JSPs, um simples servlet container como o TomCat não resolve, é preciso que tenhamos um Application Server completo. Clicando em New Runtime podemos selecionar um dos servidores disponíveis. Apesar das diferenças, as views usadas são as mesmas para a maior parte do trabalho. O destaque vai para os wizards de criação com facilidades diferentes para criar elementos como EJBs, Web Services e arquivos de configuração.

Por último, ainda é bastante comum que um sistema seja planejado pelo seu banco de dados. Apesar dessa prática ir contra o design incremental e seja uma indicação clara de Big Design Up Front, se essa é a sua realidade, o Eclipse consegue ajudar no mapeamento de entidades (http://www.caelum.com.br/curso/fj-25-persistencia-jpa2-hibernate/). Se já tivermos um banco de dados criado e precisarmos criar as classes equivalentes às tabelas, podemos criar um projeto JPA (ctrl + N JPA Project) e fazer engenharia reversa de um banco que já existe (ctrl + N JPA Entities from Tables). Outras facilidades dessa view são gerar o `persistence.xml` e criar novas entidades (ctrl + N JPA Entity), por exemplo.

Esse tipo de projeto é associado à perspectiva JPA com várias views pertinentes que mostram as estruturas das entidades, das tabelas, de forma que você não precisa lembrar qual é configuração correta para determinada personalização que você precisa fazer. Apenas faça. O Eclipse ajuda.