UNIDADE A

REVISÃO DE CONCEITOS DE ORIENTAÇÃO A OBJETOS

Caro(a) Aluno(a):

Nesta disciplina, o principal objetivo é que os alunos possam analisar, projetar e desenvolver soluções de *software* sob o paradigma da orientação a objetos para sistemas Web, utilizando Java e suas diversas tecnologias. Para o bom acompanhamento desta disciplina, são necessários os conteúdos trabalhados nas disciplinas de Linguagem de Programação Orientada a Objetos, Análise e Projeto de Sistemas de Informação Orientados a Objetos e também da Linguagem de Programação Visual.

O objetivo básico desta unidade é revisar os conteúdos e conceitos envolvidos principalmente nas disciplinas de Linguagem de Programação Orientada a Objetos, Análise e Projeto de Sistemas de Informação Orientados a Objetos e, após, realizar uma introdução aos sistemas Web baseados em Java.

Conceitos de Orientação Objetos

Como visto nos conteúdos anteriores, o paradigma de programação orientado a objetos é o paradigma de programação que utiliza objetos (que possuem um estado e comportamento), criados a partir de modelos, para representar e processar dados, usando programas de computadores. As técnicas de programação incluem características como **abstração, encapsulamento, modularidade, polimorfimo** e **herança.** Desta forma, um programa orientado a objetos é composto de **objetos** que colaboram entre si para a realização de tarefas.

A seguir, revisaremos os principais conceitos de orientação a objetos.

Classes

Os programadores que utilizam o paradigma orientado a objetos criam e usam objetos a partir de classes, que são relacionadas diretamente com modelos. Desta forma, **classes** são estruturas das linguagens de programação OO para conter, para determinado modelo, os dados que devem ser representados e as operações que devem ser efetuadas com estes dados (estado e comportamento). Cada classe deve ter um nome que seja facilmente associado ao modelo que a classe representa.

Objetos

Os objetos possuem um estado e comportamento e são criados a partir de um modelo (a classe). Para a representação de dados específicos usando classes será necessária a criação de **objetos** ou **instâncias** desta classe. Um **objeto** ou **instância** é a materialização de uma classe (modelo utilizado para representar dados e executar ações). Para que os objetos ou instâncias possam ser manipulados, é necessária a criação de **referências** a esses objetos, que são basicamente variáveis do "tipo" classe.

Atributos

Os dados contidos em uma classe são conhecidos como **campos** ou **atributos** (variáveis em algumas linguagens de programação) daquela classe. Cada campo deve ter um nome e tipo, que será ou um tipo de dado nativo da linguagem ou uma classe existente na linguagem ou definida pelo programador.

Métodos

Os **métodos** são definidos na declaração de uma classe e definem o comportamento (operações) dos objetos daquela classe (funções em algumas linguagens de programação). Estes métodos operam sobre os atributos internos e servem como mecanismo primário para comunicação entre objetos. Métodos são geralmente chamados ou executados explicitamente a partir de outros trechos de código na classe que o contém ou a partir de outras classes. Em um programa OO, os objetos de um sistema **trocam mensagens** para que as tarefas sejam realizadas. Esta troca de mensagens é chamada de método.

Encapsulamento

O encapsulamento é um conceito chave para trabalhar com orientação a objetos. Os objetos possuem comportamento que diz respeito a operações realizadas por um objeto e também ao modo pelo qual essas operações são executadas. O **mecanismo de encapsulamento** é uma forma de restringir o acesso ao comportamento interno de um objeto, ou seja, caso um objeto necessite de colaboração de outro objeto para realizar uma tarefa, ele simplesmente envia uma mensagem a este último.

Herança

A herança é outra forma de abstração em orientação a objetos, onde classes semelhantes são agrupadas em hierarquias. Cada nível de hierarquia pode ser visto como um nível de abstração, em que cada classe em um nível da hierarquia herda as características das classes dos níveis acima. Esse mecanismo, também conhecido como generalização/especialização, facilita o compartilhamento de comportamento comum entre um conjunto semelhante de classes.

Polimorfismo

O polimorfismo indica a capacidade de abstrair várias implementações diferentes em uma única interface. A mesma computação funciona para objetos de muitas formas e adapta-se à natureza dos objetos.

Diagramas UML

Na disciplina de Análise e Projeto de Sistemas de Informação Orientados a Objetos, foi estudado o processo de desenvolvimento de um sistema orientado a objetos e utilizada como ferramenta a linguagem UML (*Unified Modeling Language*). A linguagem UML foi criada para a modelagem e documentação de sistemas de *software* orientado a objetos. A UML é uma linguagem padrão para modelagem de *software* que possuem um conjunto de diagramas, que suportam a descrição e o projeto de *software*.

Todos os diagramas de UML são importantes, porém iremos focar a revisão em dois diagramas: diagramas de classe e casos de uso.

Diagramas de Classes

Os diagramas de classes mostram as diferentes classes que fazem parte de um sistema e como elas se relacionam. Uma classe é representada em UML por um retângulo, com o nome da classe, e podem também mostrar os atributos e operações (métodos) da classe. A Figura A.1 apresenta um exemplo de diagrama de classe.

As classes de um diagrama de classes podem possuir diferentes relacionamentos como:

Generalização

O relacionamento de generalização conecta classes generalizadas com outras mais especializadas. A generalização é um relacionamento de itens gerais (superclasses) e itens mais específicos (subclasses) e é considerado um como “é um tipo de”;

Associação

A associação é um relacionamento estrutural entre instâncias (especifica objetos de um item conectados a objetos de outro item). Uma pura associação entre duas classes representa um relacionamento estrutural entre pares em que as classes estão em um mesmo nível e uma não é mais importante que outras. Também é possível a partir de uma associação conectando duas classes, navegar do objeto de uma classe até o objeto de outra classe e vice-versa. Em associação um objeto “usa um” outro objeto;

Agregação

Em uma pura associação entre duas classes, essas classes estão em um mesmo nível e uma não é mais importante que outras. Contudo, em certas modelagens, é necessário representar uma classe, que representa um item maior (o “todo”) que é formada por itens menores (as “partes”). Na agregação um objeto é composto por (ou é parte de) outros objetos. Em outras palavras, uma agregação é uma associação mais específica;

Composição

A composição é uma forma de agregação, uma especialização da agregação, com propriedade bem definida e tempo coincidente como parte do todo. Na agregação, se a instância do todo for removida, suas partes não serão necessariamente removidas. Já na composição, se a instância do todo for removida, suas partes também deverão ser removidas.

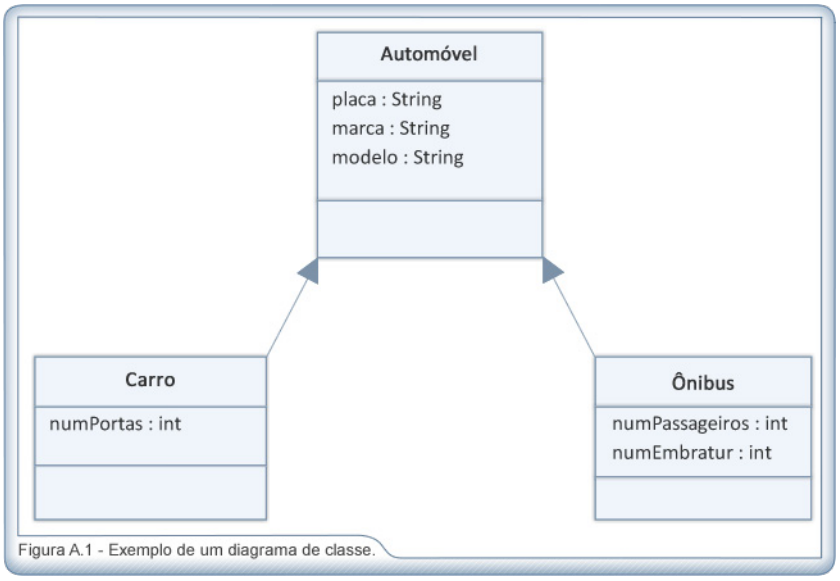

Figura A.1 - Exemplo de um diagrama de classe.

Casos de Uso

A funcionalidade do sistema é definida por um conjunto de casos de uso. Os casos de uso têm por objetivo caracterizar os requisitos funcionais do sistema e identificar entidades relevantes e sua interação com o sistema. Cada caso de uso representa uma sequência de ações e deve ser descrito textualmente. A descrição textual é um documento narrativo que descreve sequência de eventos/ações realizadas pelo sistema, quando estimulados por um ator que interage ou usa o sistema. Os casos de uso devem ser passíveis de compreensão tanto por desenvolvedores como por usuários e precisam ser completos, consistentes e não ambíguos.

Os diagramas UML também fornecem os diagramas de casos de uso, que é o conjunto de notações gráficas que permite representar os casos de uso, atores e associações entre eles. A Figura A.2 apresenta um exemplo de diagrama de casos de uso.

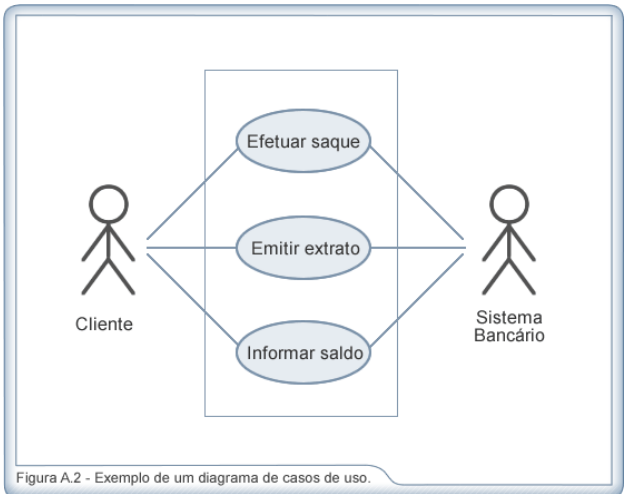

Figura A.2 - Exemplo de um diagrama de casos de uso.

Além dos conceitos de orientação a objetos e UML, você também deve lembrar como manipular estes conceitos na linguagem de programação Java. Alguns desses conceitos serão imprescindíveis para o entendimento dos conceitos e uso das ferramentas durante esta disciplina.

Nesta unidade, foram revisados de forma breve os principais conceitos envolvidos na Orientada a Objetos e dois tipos de diagramas UML. Na sequência, você irá conhecer um pouco do uso da Java nos sistemas Web.

INTRODUÇÃO AOS SISTEMAS WEB BASEADOS EM JAVA

Caro(a) Aluno(a):

Na unidade A, um dos objetivos é conhecer como a plataforma Java e a Web trabalham juntas. É importante entender que necessitamos de diversas tecnologias da plataforma Java. Desta forma, serão apresentados os principais conceitos relacionados com o desenvolvimento de aplicações Web baseada em Java, assim como a arquitetura Java EE.

Boa aprendizagem!

Introdução

Durante o curso de TSIaD, você percebeu que o desenvolvimento Web é uma das áreas que mais cresce desde 1990. Existe a tendência de que cada vez mais aplicações sejam voltadas para Web, sendo acessadas por um navegador por meio de uma intranet ou internet.

Dentre as vantagens de desenvolvimento Web, destacamos que podem existir clientes geograficamente distribuídos, manutenção centralizada sem necessitar distribuir ou instalar *softwares* e independência de plataforma (existe somente a necessidade de um navegador). Contudo, podem ser apontadas algumas desvantagens como a interface com o cliente de inferior qualidade em relação a uma interface de desktop e dependência em relação ao tráfego da rede. Porém, eu não afirmaria que estas duas últimas são grandes desvantagens. Atualmente, possuímos interfaces Web muito boas e também as velocidades de rede continuam aumentando.

Uma das linguagens mais utilizadas atualmente é Java, a qual realaciona-se com a web. Certamente um dos melhores mercados que Java se encontra é o da Web. Assim, apesar da popularidade do ambiente Web, o desenvolvimento com Java não é tão simples, uma vez que é necessário conhecer com profundidade um grande conjunto de APIs e diversas tecnologias como *servlets*, JSP, *containers*, entre outras.

Desenvolvimento em Camadas

As aplicações Web são estruturadas em camadas em que cada camada possui um papel fundamental na arquitetura dos sistemas. As aplicações Web possuem no mínimo duas camadas:

- Camada de apresentação e a lógica do negócio;
- Camada de persistência de dados (armazenamento).

As arquiteturas tradicionais cliente-servidor são de duas camadas. Nestas arquiteturas, uma das camadas é o servidor de banco de dados acessado por vários programas cliente. Cada programa cliente possui uma camada de apresentação (uma interface gráfica) e o código da lógica de negócio. A lógica de negócio é chamada assim, pois nos aplicativos comerciais contém as regras (regras de negócio) que são utilizadas para tomar decisões de negócio: quantos produtos oferecer, quanto cobrar, etc. A inteligência do aplicativo, logo a parte mais importante do *software*, está na lógica do negócio. A Figura A.3 apresenta um exemplo de modelo com duas camadas.

Figura A.3 - Arquitetura cliente-servidor de duas camadas.

Existem modelos arquiteturais propostos para melhorar a organização e a manutenção das aplicações Web. No modelo de duas camadas (apresentado na Figura A.3),o módulo do cliente é específico para o aplicativo. Quando a lógica do negócio se altera, um novo módulo cliente deve ser distribuído para os dos clientes envolvidos.

A arquitetura de camadas é muito utilizada para separar responsabilidades em uma aplicação moderna. Apesar de a ideia da divisão de uma aplicação em camadas ter sido popularizada nos anos 90, muitos desenvolvedores ainda não conhecem muito bem a técnica.

Um modelo que tem sido amplamente utilizado é o modelo em três camadas. A saber:

- Camada de apresentação - o nível mais alto da aplicação é a interface do usuário. A principal função da interface é traduzir as tarefas e resultados para que o usuário possa entender;
- Camada da lógica do negócio - esta camada coordena a aplicação, processa comandos, realiza decisões lógicas e executa cálculos. Também é responsável pela troca de dados entre as camadas de apresentação e de armazenamento (dados);
- Camada de armazenamento (dados) - nesta camada, a informação é armazenada e recuperada do banco de dados.

Ao contrário de um modelo de duas camadas, no modelo de três camadas a lógica do negócio é residente em um servidor. Quando a lógica se altera, o código do servidor é atualizado, enquanto a camada de apresentação (navegador) continua inalterada. A Figura A.4 apresenta uma arquitetura de três camadas.

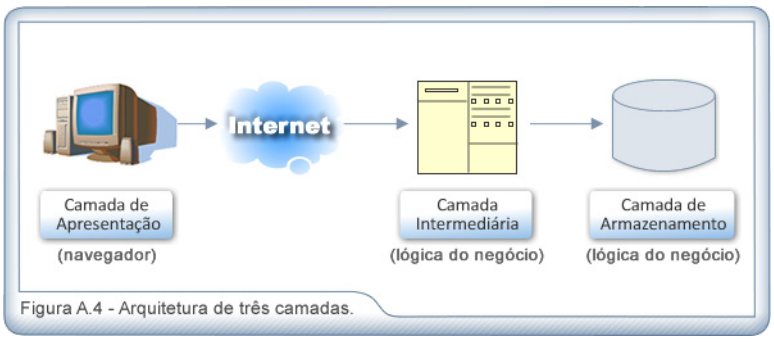

Figura A.4 - Arquitetura de três camadas.

A grande vantagem é separar a camada de apresentação da camada de lógica de negócio, apesar de realizar um maior número de operações de recuperação e passagem de valores.

Junto com a popularização da arquitetura de camadas, ressurgiu o modelo MVC (*Model-View-Controller*) de desenvolvimento. O modelo MVC e o de camadas são conceitos diferentes, que podem ser aplicados em conjunto ou não. Num Modelo MVC, os componentes são divididos em três: *View, Model e Controller*. A *View* é a parte exposta, o *Controller* é o controle sobre a comunicação que vem do usuário para o sistema e o *Model* representa o estado do sistema.

Arquitetura Java EE

Como visto na disciplina de LPOO, Java possui com grande conjunto de tecnologias que são agrupadas em diferentes plataformas. Estas plataformas agrupam programas relacionados que permitem o desenvolvimento e execução de programas escritos na linguagem de programação Java. As plataformas são divididas em quatro:

- Java Platform Enterprise Edition (JEE);
- Java Platform Standard Edition (JSE);
- Java Platform Micro Edition (JME);
- JAVA FX.

Todas as plataformas consistem de uma máquina virtual Java (JVM) e uma API. Quando a maioria das pessoas pensa na linguagem de programação Java, elas pensam na plataforma Java SE.

Java EE é uma plataforma construída a partir da plataforma Java SE e consiste de uma série de especificações bem detalhadas. O Java EE fornece uma API e um ambiente de execução para desenvolvimento e execução de aplicações de larga escala que envolvem múltiplas camadas, escaláveis, confiáveis e seguras. Estas aplicações ganharam o nome de *enterprise applications*, pois são projetadas para solucionar problemas encontrados em grandes corporações. As *enterprise applications* não são somente úteis para grandes corporações e governos, mas também podem beneficiar desenvolvedores individuais e de pequenas empresas.

A arquitetura Java EE também utiliza um modelo multicamadas (camada de apresentação, camadas intermediárias e camada de armazenamento) onde o desenvolvimento de uma aplicação Java EE se concentra nas camadas intermediárias para tornar o gerenciamento da aplicação mais fácil, mais robusto e mais seguro.

A arquitetura multicamadas da Java EE é dividida nas seguintes camadas:

Camada cliente (*client-tier*)

Nesta camada, os componentes residem em um *container* que pode ser um navegador Web, uma *Applet* Java ou uma aplicação cliente (os componentes executam na máquina cliente);

Camada Web (*web-tier*)

Fornece a lógica à camada cliente que é implementada por uma JSP e *Servlets* (os componentes executam em um servidor Java EE);

Camada de negócio (*business-tier*)

Uma das camadas mais importantes para o desenvolvedor, pois trata de toda a lógica da aplicação. É aqui que se definem todas as regras do negócio, alocação de recursos, validação de dados de segurança (os componentes executam em um servidor Java EE);

Camada EIS (*Enterprise Information System*) (*EIS-tier*)

Nesta camada, encontram-se os sistemas de banco de dados, a integração com outros sistemas não J2EE.

Para desenvolvimento na plataforma J2EE, é necessário entender cada uma dessas camadas e seu funcionamento. A Figura A.5 apresenta a arquitetura multicamadas da Java EE.

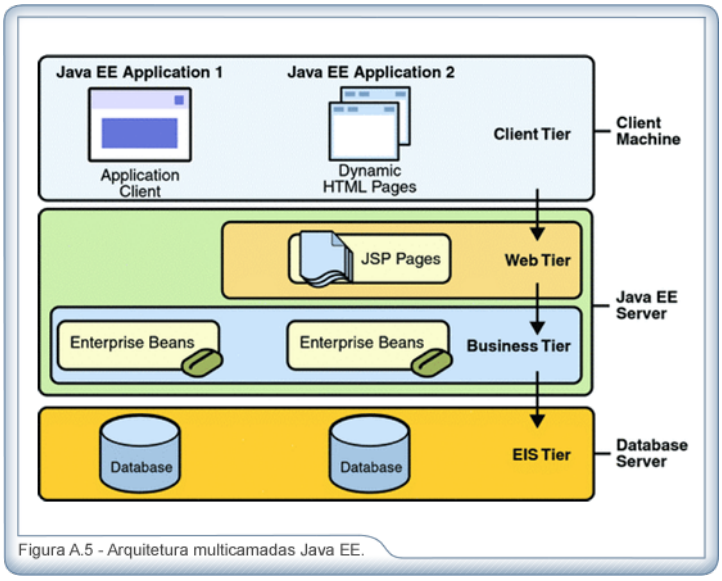

Figura A.5 - Arquitetura multicamadas Java EE.

Fonte: http://download.oracle.com/javaee/5/tutorial/doc

A arquitetura Java EE especifica um conjunto de tecnologias utilizadas nas diversas camadas. Tecnologias Java EE utilizadas na camada Web:

- Servlets;
- JavaServer Faces technology;
- JavaServer Faces Facelets technology;
- Expression Language;
- JavaServer Pages (JSP);
- JavaServer Pages Standard Tag Library;
- JavaBeans Components.

A camada de negócio consiste em fornecer um conjunto de regras de negócio para a aplicação. Esta camada também envolve um conjunto de tecnologias:

- Enterprise JavaBeans (enterprise bean) components;
- JAX-RS RESTful web services;
- JAX-WS web service endpoints;
- Java Persistence API entities.

A camada EIS consiste nos servidores de banco de dados, entre outras e também possui um conjunto de tecnologias envolvidas:

- The Java Database Connectivity API (JDBC);
- The Java Persistence API;
- The Java EE Connector Architecture;
- The Java Transaction API (JTA).

Algumas das tecnologias apontadas acima, como JSP, servelts e JDBC, serão trabalhadas com mais profundidade. Porém, existem dois conceitos importantes que você deverá aprender: servidores de aplicação (*Java EE Servers*) e *containers* (*Java EE Containers*).

A comunicação entre as camadas e os servidores pode ser observada na Figura A.6. No caso de uma aplicação acessada por um navegador, a comunicação deve ser feita com a camada Web para depois acessar a camada de negócio e a camada EIS. Já uma aplicação cliente Java, pode acessar diretamente a camada de armazenamento.

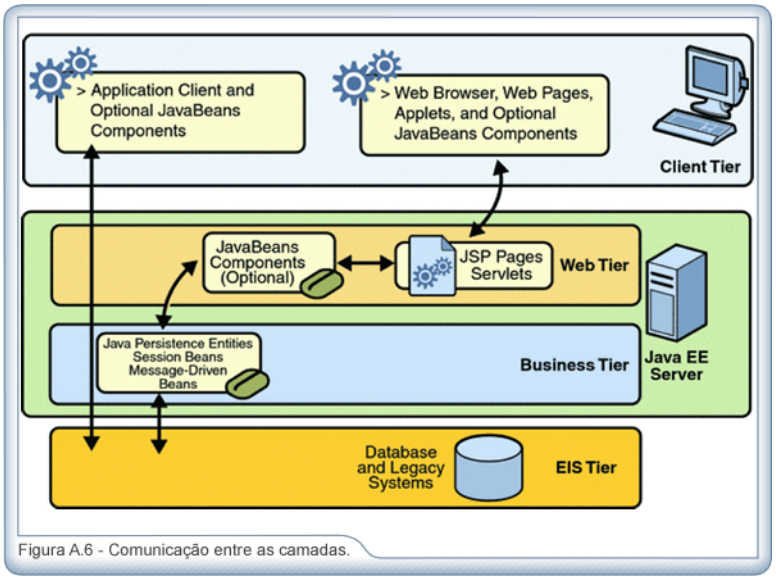

Figura A.6 - Comunicação entre as camadas.

Fonte: http://download.oracle.com/javaee/5/tutorial/doc

Servidores de Aplicação

Um servidor de aplicação Java EE implementa as APIs da plataforma Java EE e fornece os serviços padronizados de Java EE. Tais servidores ganham este nome (servidores de aplicação) porque eles fornecem (servem) a dados de aplicação para os clientes. Existem diversas implementações destes servidores. Alguns desses servidores estão listados a seguir:

- RedHat, JBoss Application Server;
- Sun, GlassFish;
- Apache, Apache Geronimo;
- IBM, IBM Websphere Application Server;

Algumas implementações programam apenas uma parte das especificações do Java EE, como o Apache Toncat que só programa JSP e Servlets, portanto não é totalmente correto chamá-lo de servidor de aplicação; o correto é utilizar o termo *application server web profile*. Ressalto que, nesta disciplina, será utilizado o Apache Toncat.

Os servidores de aplicação suportam diferentes tipos de componentes Java. Um servidor de aplicação fornece serviços para estes componentes na forma de um *container.*

Containers

Os Java EE *containers* são interfaces entre um componente Java e as funcionalidades específicas fornecidas por uma plataforma para suportar tal componente. A funcionalidade do container é definida pela plataforma e é diferente para cada tipo de componente.

Para nós, interessa um *container* que suporte as funcionalidades para o desenvolvimento de aplicações Web

como JSP, Servlets, JSTL e JSF. Este *container* (*servlet container*) não necessita implementar todo do Java EE. Existem diversos *servltes container*. Nós utilizaremos o Apache Toncat. A Figura A.7 apresenta a relação de servidores de aplicação e *containers*.

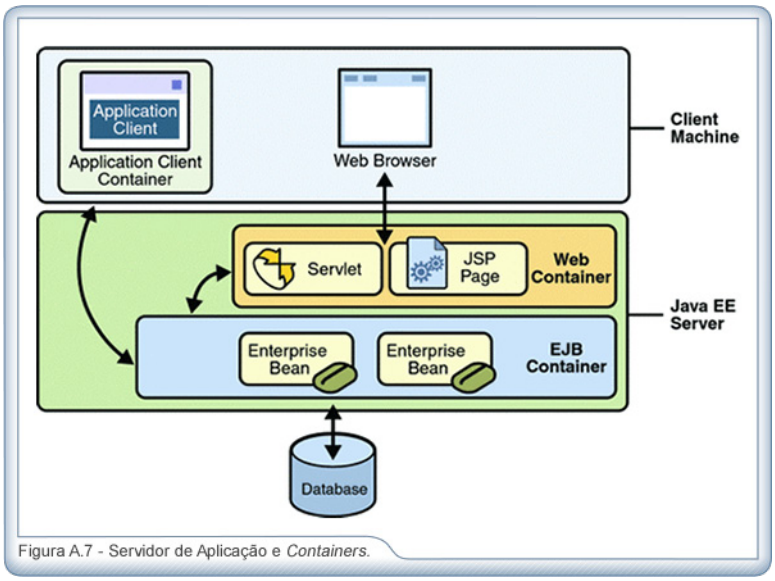

Figura A.7 - Servidor de Aplicação e *Containers*.

Fonte: http://download.oracle.com/javaee/5/tutorial/doc

Dica:

Estude mais sobre Java EE. Existe muita documentação na internet. Um bom tutorial pode ser encontrado no site da Oracle:

http://download.oracle.com/javaee/5/tutorial/doc.

Instalação do Apache Toncat

O Apache Toncat será necessário para o desenvolvimento das aplicações Web dinâmicas com Java. O Toncat é o *container* Web mais utilizado na Web e foi desenvolvido pela *Apache Software Foundation* com *software* de código aberto. O Toncat pode ser obtido pelo site:

http://tomcat.apache.org/

Contudo, tanto o Toncat quanto o GlassFish (um servidor de aplicação da Oracle) podem ser instalados juntamente com o NetBeans.

O NetBeans possui um versão para download que já contém o Apache Toncat e GlassFish. Para realizar o donwload acesse:

http://netbeans.org/downloads/index.html

A Figura A.8 salienta a opção de download que deve ser selecionada.

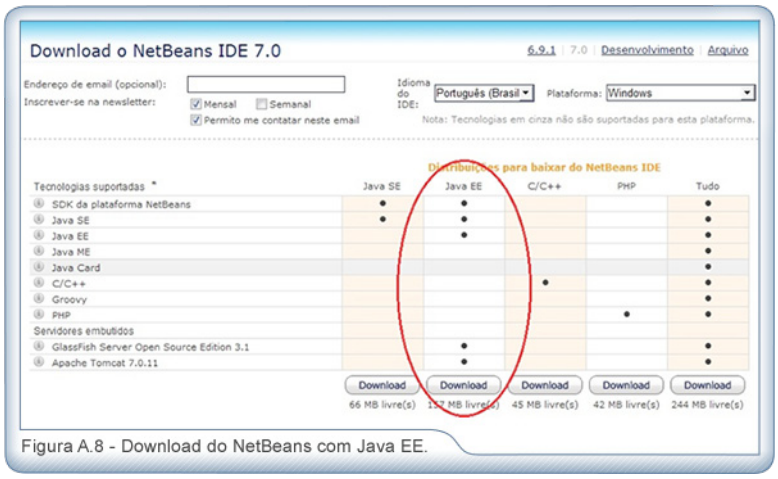

Figura A.8 - Download do NetBeans com Java EE.

Nessa unidade, foram apresentados os principais conceitos relacionados com o desenvolvimento de aplicações Web baseada em Java. Também foi introduzida a arquitetura Java EE. Na próxima unidade, serão apresentados os documentos XML e você aprenderá como ler e criar esses documentos em Java.

Atividade 1

Prezado(a) aluno(a), a atividade da Unidade A trata do desenvolvimento de sistema orientado a objetos utilizando Java. Desenvolva o sistema na linguagem Java e os execute no NetBeans. Também apresente o diagrama de casos de uso e diagrama de classes.

1. Modele e implemente os elementos de um **sistema para controle de eventos**. O sistema deverá permitir o cadastro de diferentes atividades como palestras e cursos. Tal sistema também deverá permitir o cadastro de participantes em diferentes modalidades como palestrante e ouvinte.