1 Arquitetura Distribuída

1.1 J2EE e Arquitetura Distribuída

Como sabemos, Java é uma excelente plataforma de desenvolvimento para softwares distribuídos, pois nasceu na grande rede. Um projeto, que foi lançado em 1995, não poderia desconsiderar que qualquer aplicativo da nova geração teria, em caso de sucesso, milhares de acessos por dia.

No cenário tecnológico atual, vários componentes participam da arquitetura de um software e cada vez mais eletro-eletrônicos, de uma maneira em geral, tornam-se dispositivos de acesso à grande rede. Com isso, cada vez mais, é importante considerar uma solução coorporativa com:

Escalabilidade

Um dos "ades" da engenharia de software que pertence à nova economia globalizada e conectada. Uma campanha de marketing pode fazer com que uma pequena empresa cresça violentamente do dia para a noite. Se sua solução não for capaz de acompanhar, o negócio pode ser prejudicado.

Portabilidade

A portabilidade faz com que não seja necessário comprar um servidor ou sistema operacional de nenhuma marca específica para seu software / solução.

Disponibilidade

Com uma linguagem e plataforma confiáveis, podemos contar com sistemas que rodam simultaneamente em mais de um servidor para prevenção de falhas. Sistemas conhecidos como cluster de servidores permitem que aplicativos fiquem distribuídos e à prova de falha no seu data-center.

Performance

Quando o assunto é computação distribuída, Java tem uma performance muito boa em comparação com outras tecnologias. Podemos dizer, por exemplo, que Fortran é mais rápido que Java para resolver um algoritmo complexo isoladamente, mas Java quando distribuído entre servidores não possui concorrente.

Baixo custo de manutenção

Código bem escrito, objetos bem definidos e documentados, herança, polimorfismo, interfaces da orientação a objeto e outros aspectos técnicos tornam um software simples de ser mantido no decorrer do tempo, quando bem planejado e escrito em Java. Sabemos que, cada vez mais, os aplicativos tendem a ter um ciclo de vida cada vez mais duradouro na empresa, por isto o custo de manutenção do software se torna cada vez mais importante.

1.2 Plataforma J2EE

J2EE é um padrão de desenvolvimento em camadas que disponibiliza uma série de serviços de infra-estrutura de alto-nível evitando o desenvolvimento de código complexo, aproximando os desenvolvedores do negócio em si.

A plataforma J2EE é composta por:

- **"Blueprints Design Guidelines for J2EE"** desenvolvido por técnicos altamente capacitados e reúne, em documentos e exemplos de código, as melhores práticas de desenvolvimento de aplicação J2EE.
- **J2EE Compatibility Test Suite**: Processo formal de teste de compatibilidade de Application Server J2EE garantindo a padronização entre os servidores de diferentes fabricantes.
- **J2EE Reference Implementation**: A plataforma inclui um servidor chamado de R.I. (reference implementation) implementado com 100% das funcionalidades especificadas. É a chave para validar aplicações J2EE. 100% free e com código fonte disponível.
- **APIs:** Todo poder e sucesso do J2SE + suporte para Enterprise JavaBeans, Java Servlets API, Java Server Pages, XML e Messaging.

Podemos dividir J2EE em duas partes:

- Infra-estrutura de Runtime para hospedagem de aplicações (servidor de aplicação)
- Um conjunto de APIs para construção de aplicações.

A **infra-estrutura de Runtime** é composta por uma conjunto de objetos e processos que **hospedam aplicações desenvolvidas** com as APIs **J2EE** (Servlets, JSP, EJBs etc.). Esta infra-estrutura é chamada de container.

O conjunto de APIs J2EE é definida, em sua maioria, por interfaces que podem ser empregadas pelos desenvolvedores das aplicações corporativas. Tais APIs possuem vínculos com o núcleo (kernel) do servidor que executam tarefas voltadas para o gerenciamento de recursos e infra-estrutura.

Podemos dizer que, ao desenvolvermos aplicações J2EE através do uso de APIs disponibilizadas no Java 2 Enterprise Edition (Reference Implementation), elas podem ser executados nos **servidores** de **aplicações** que tenha **implementado** as especificações técnicas dos servidores **J2EE**.

Contamos com as seguintes APIs na plataforma J2EE:

- JDBC Extension
- Enterprise JavaBeans (EJB)
- Java Servlets
- Java Server Pages
- Java Message Service
- Java Transaction API
- JavaMail

Anotações

1.2.1 Servidores / Containers de Componentes

Container são servidores de objetos modernos que oferecem serviços e infra-estrutura para a execução de uma aplicação distribuída.
Temos uma divisão de perfil de containers Java e J2EE:

Server-side

- **Web container** - para objetos dirigidos por HTTP (Servlets e JSP).
- **EJB container** - para objetos de negócio server-side.

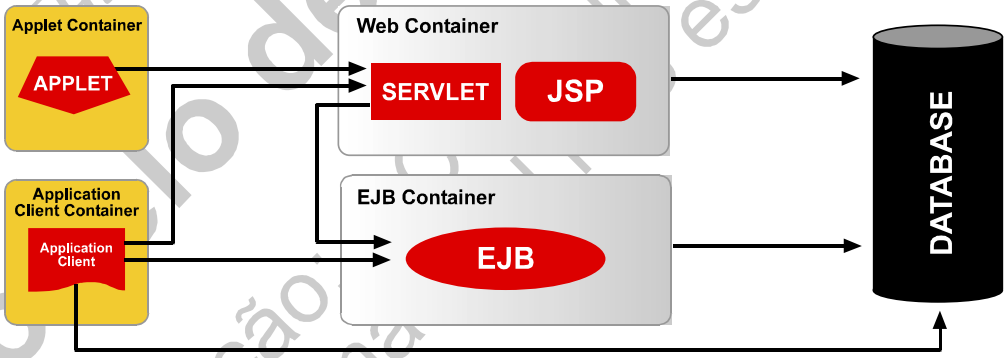
Client-side

- **Applet container** - para painéis gráficos desenvolvidos com AWT/Swing controlados por browser.
- **Application client container** - aplicações stand-alone AWT / Swing, podendo, opcionalmente, ser distribuídas por Java Web Start.

Um **container client-side** é responsável pelo ciclo de vida da aplicação, gerenciamento de eventos e outros. Já um container **server-side** gerencia,além do ciclo de vida de componentes, recursos e meios de acesso.
Configuramos no container os recursos que desejamos disponibilizar e nossas aplicações acessam tais recursos através de APIs de serviços, como no caso de um pooling de Conexões a Banco de Dados.

Na figura abaixo podemos visualizar dois containeres J2EE:

- **Web Container:** hospeda Servlets, JSPs (Java Server Pages), arquivos estáticos como HTMLs, JavaScript, XML, e pode utilizar qualquer classe Java, como por exemplo, APIs de envio de e-mail ou acesso a banco de dados.
- **EJB Container**: hospeda Enterprise JavaBeans (EJB), que por sua vez também podem utilizar uma série de APIs Java, como por exemplo, envio de e-mail e acesso a banco de dados.



Representamos aqui um banco de dados, mas com arquitetura J2EE podemos acessar arquivos, sistemas legados, ERPs, fila de mensagens e qualquer outra fonte de dados.

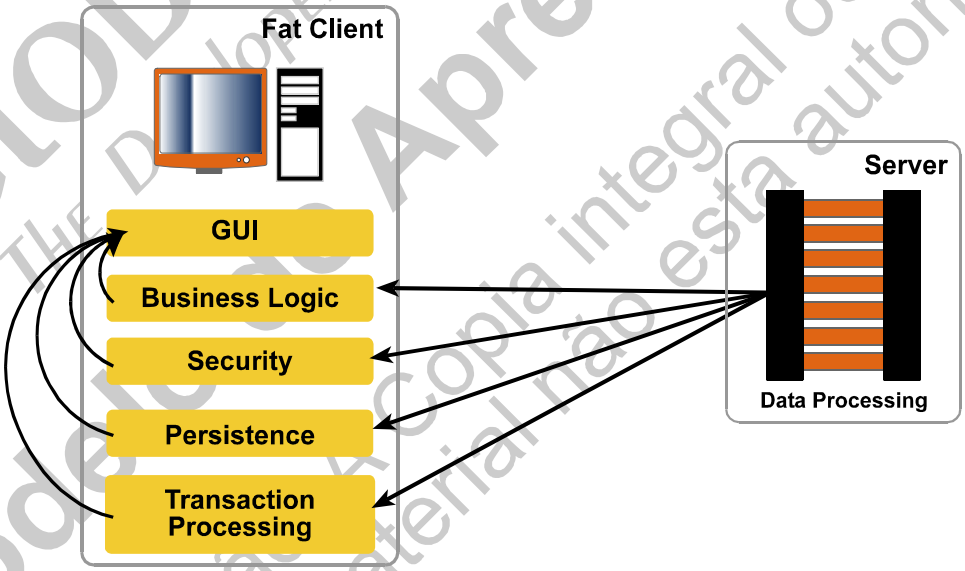
1.3 Análise de arquiteturas

1.3.1 Arquitetura em duas camadas: cliente / servidor

Arquitetura típica quando não temos o conceito de servidor intermediário, na qual temos muitas aplicações construídas. Exemplos de tecnologias tipicamente utilizadas:

- Visual Basic com Banco de Dados;
- Oracle Forms / Report com Oracle Database;
- Centura / Power Designer com Sybase;
- Delphi com Banco de Dados;

Por ser uma arquitetura que centraliza todo o processamento crítico no banco de dados, acaba sendo limitado para integração, além de tornar o código muito complexo devido à linguagem SQL.



Anotações

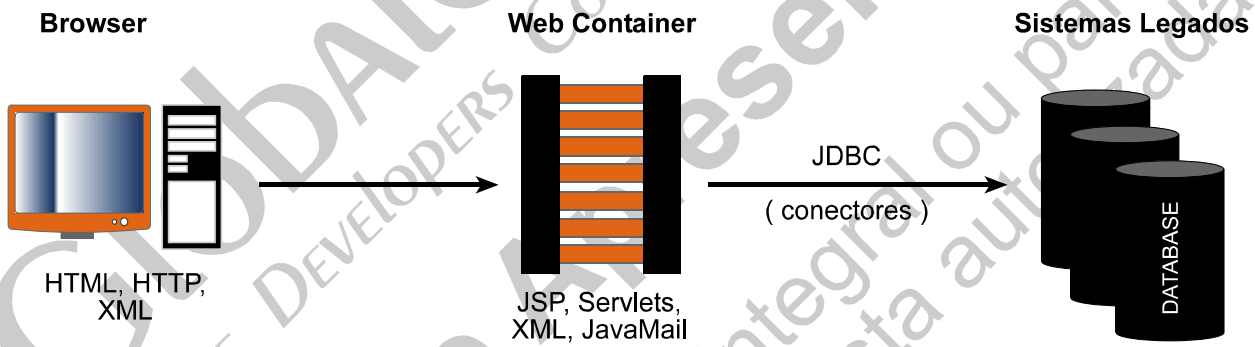
1.3.2 Arquitetura em três camadas: baseada na Web com cliente “magro”

Este padrão permite uma melhor distribuição de processamento e também usa um cliente mais enxuto para acessar a aplicação.

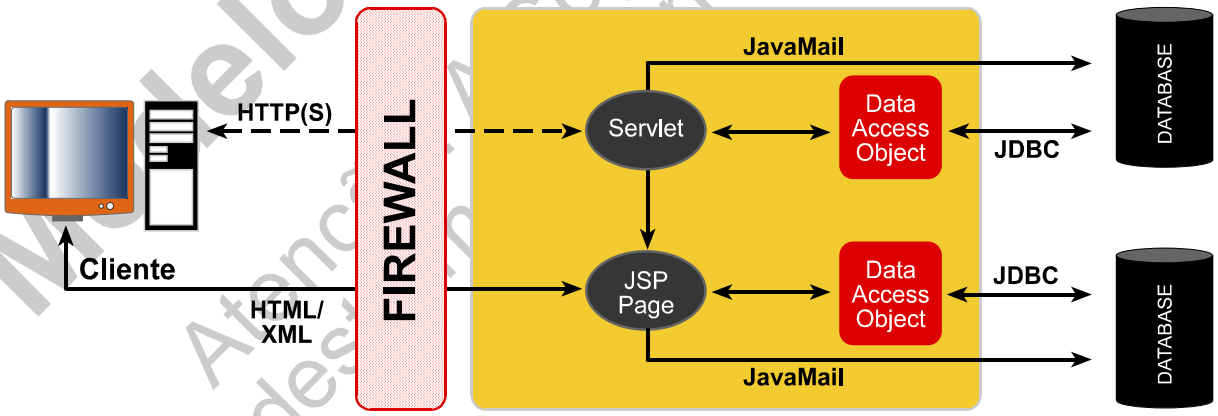
Esta arquitetura é o foco do treinamento deste módulo e talvez uma das arquiteturas mais procuradas e utilizadas para desenvolvimento para Internet.

Devemos sempre tomar cuidado com o modismo de utilizar browser / HTML como cliente, pois o HTML muitas vezes restringe ou dificulta o desenvolvimento de uma boa interface gráfica.
Quando a solução for baseada na Web utilizando HTML, sugerimos que tenha sempre um web-designer na equipe de desenvolvimento.

Vale lembrar que podemos ter soluções baseadas na Web que utilizam clientes “gordos”, desta forma a acessibilidade da solução não fica prejudicada e podemos ter clientes mais específicos. Para isto, podemos utilizar Servlets ou JSPs, gerando XML para clientes Macromedia Flash, Swing ou qualquer outro.



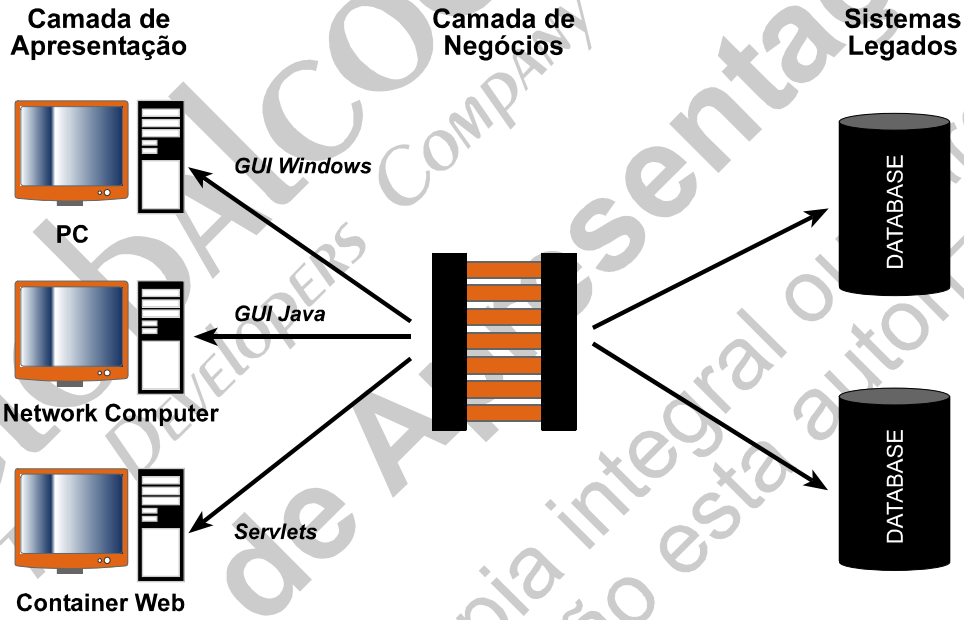
Uma das vantagens que soluções baseadas na Web apresentam é a segurança, já que permite a utilização de criptografia nos dados (HTTPS) sem exigir trabalho adicional com o firewall.



1.3.3 Arquitetura em três camadas: utilizando EJB

Esta arquitetura favorece a distribuição das regras de negócio da solução, além de facilitar a manutenção do código. Temos clientes que podem ser programados com AWT ou Swing que utilizam objetos remotos em um container EJB.

Todo código de acesso a dados (SQL) fica no servidor intermediário e o objeto se beneficia de diversos serviços oferecidos pelo container.



Anotações

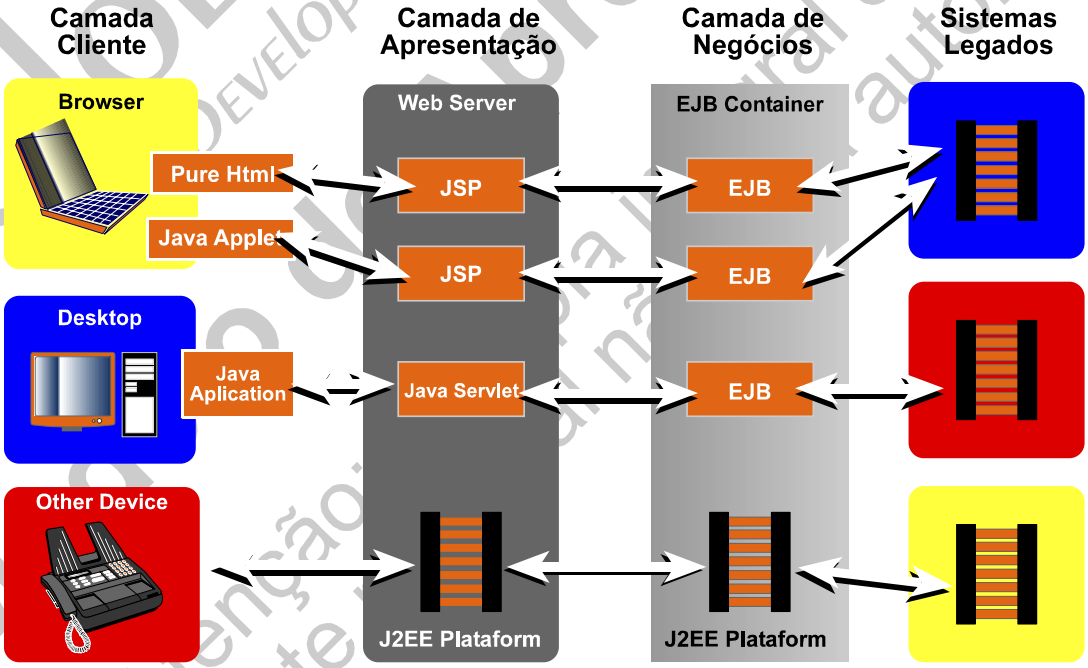
1.3.4 Arquitetura em n-Camadas: baseada na Web, com EJB

J2EE é tipicamente consistido nas seguintes camadas:

- **Camada Cliente:** Browser / HTML, Applets, aplicações Java (Swing/AWT), Flash e outros.
- **Camada Web:** É responsável pela comunicação com a camada de negócios. Geralmente possui lógica de apresentação e formatação de dados. Tipicamente gera HTML ou XML e é implementada com Servlets e JSPs.
- **Camada de Negócios:** Geralmente a "inteligência" da aplicação está nesta camada. Na maioria das ocasiões esta camada é implementada utilizando EJBs. O container EJB vai prover serviços de persistência, transações e gerenciamento de ciclo de vida.
- **Camada de Dados ou Integração:** É a camada responsável pela comunicação com os bancos de dados e sistemas legados.

A plataforma J2EE simplifica o desenvolvimento de soluções corporativas, provendo infra-estrutura e padronização.

Como mostra a figura abaixo, ao término de um projeto J2EE temos um conjunto de componentes para apresentação de dados e componentes de regras de negócio reutilizáveis, utilizando tanto o container Web quanto o container EJB.



2 Desenvolvimento para Web

2.1 Container Web

Aplicações Web J2EE utilizam o conceito de container. O container Web, além de gerar conteúdo dinâmico através da execução dos JSPs e Servlets, oferece os seguintes serviços:

- Gerenciamento dos recursos utilizados pelos componentes. **Exemplo**: Conexões com o banco de dados.
- Gerenciamento do ciclo de vida dos componentes.
- Gerenciamento de sessões de usuários.
- Gerenciamento de segurança da aplicação.

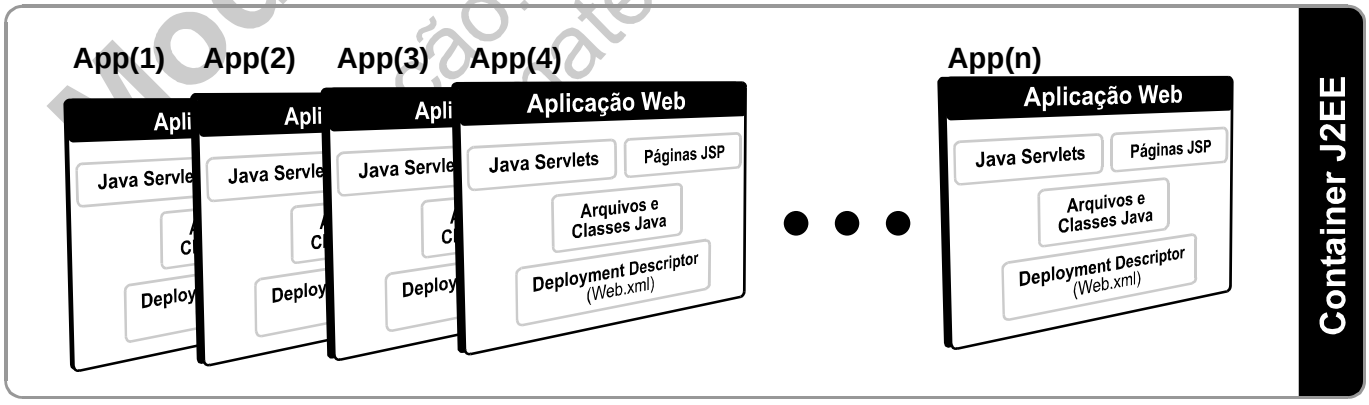
As duas principais APIs do container Web são: **Java Servlets** e **Java Server Pages**.

A **implementação de referência** de container Web é o Tomcat. Observe que há alguns servidores que se denominam compatíveis com o J2EE, mas que não passaram pelo teste de Compatibilidade. Em caso de dúvidas, pode-se encontrar uma lista dos containers J2EE na seguinte URL
http://java.sun.com/j2ee/compatibility_1.3.html

2.1.1 Aplicações Web

Quando desenvolvemos uma aplicação para um container Web, devemos desenvolver:

- **Componentes Web:** Servlets, JSP, JavaBeans etc
- **Deployment Descriptors:** arquivos XML que descrevem os componentes desenvolvidos para registro no container.



Anotações

2.2 Tomcat

O Tomcat foi originado a partir de um projeto da Sun chamado Java Web Server e foi doado para a Apache Foundation.

Versionamento do servidor Tomcat:

Tomcat	Servlet	JSP
3.xx	2.2	1.1
4.xx	2.3	1.2
5.xx	2.4	2.0

Estas APIs possuem versionamento independente, ou seja, não precisam acompanhar o J2EE.

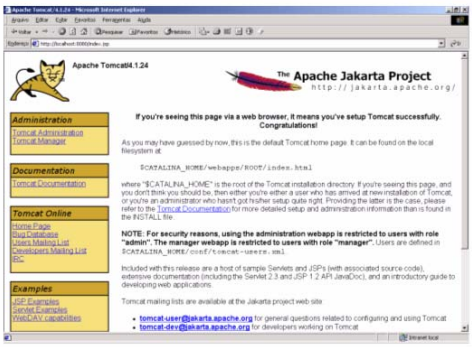
O Tomcat é um container Java e vem acompanhado de um servidor Web chamado **Coyote**, no entanto, em aplicações corporativas freqüentemente é necessário utilizar um servidor Web mais robusto, como é o caso do **Apache Web Server**.
Para maiores informações sobre a configuração do Tomcat com o Apache Web Server, consulte a documentação do Tomcat:
http://jakarta.apache.org/tomcat/tomcat-3.2-doc/tomcat-apache-howto.html

2.2.1 Instalação

URL para download: `http://jakarta.apache.org`

- Existe uma versão especial para Windows que possui um programa de instalação do Tomcat, mas também está disponível um arquivo .zip que pode ser descompactado em qualquer sistema operacional, sendo este o padrão para instalá-lo:
- Configure a variável de ambiente **CATALINA_HOME** em seu sistema operacional, atribuindo o nome do diretório onde o Tomcat foi instalado.
- Inicialização:

Windows: Execute o **startup.bat** presente no diretório **%CATALINA_HOME%\bin**.
Linux: Execute o **startup.sh** no diretório **$CATALINA_HOME/bin**, digitando: **./startup.sh**



- Para testar se o Tomcat está funcionando, abra o navegador e coloque o seguinte endereço: `http://localhost:8080`

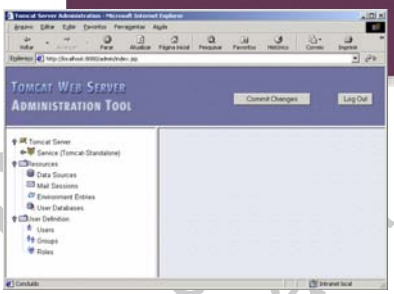
Observação: Por padrão, o Tomcat vem configurado para trabalhar na porta 8080.

- Se a página inicial do Tomcat aparecer, a instalação do Tomcat foi feita com sucesso.

2.2.2 Administração

Tomcat Administration

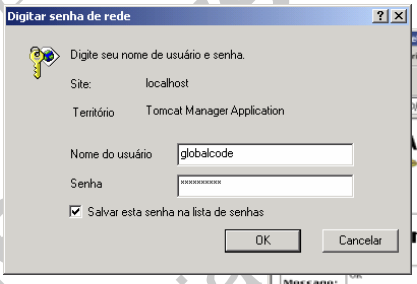
A partir da versão 4.1, o Tomcat tem uma aplicação de administração `http://localhost:8080/admin`

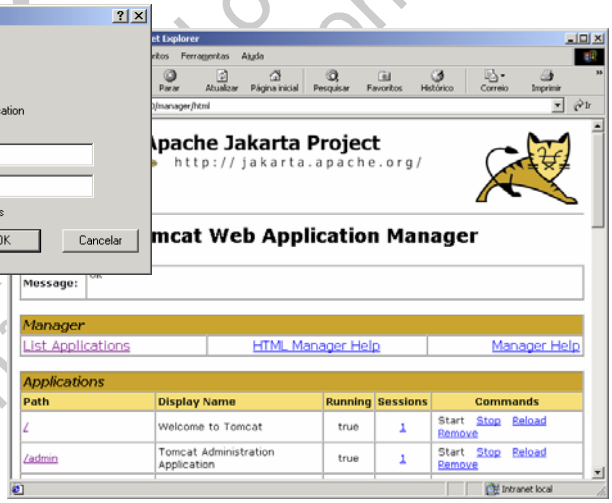


Para fazer o login nesta aplicação, será necessário editar o arquivo **tomcat-users.xml** que, como veremos mais a frente, contém as regras de segurança. Embora estejamos editando um arquivo XML (o que parece muito inseguro), é possível fazer autenticação no banco de dados ou em servidores de nomes e diretórios. **Exemplo**: Microsoft Active Directory ou qualquer outro diretório LDAP.

Tomcat Manager

Esta aplicação está disponível através da URL: `http://localhost:8080/manager/html`





Esta aplicação faz o gerenciamento de todas as aplicações instaladas neste servidor, permitindo que seja possível iniciar e parar uma aplicação específica sem a necessidade de reinicializar o servidor inteiro.

Também será necessário adicionar um usuário no arquivo **tomcat-users.xml** para fazer o login nesta aplicação.

Anotações

tomcat-users.xml

Este documento pode ser encontrado em $**CATALINA_HOME/conf** e nele podemos definir **roles** para usuários, ou seja, grupos aos quais os usuários pertencem. Cada usuário pode pertencer a um ou mais **roles**. Enquanto no arquivo de configuração da aplicação configuramos os roles que podem acessá-la, como será visto mais adiante.

Neste arquivo estamos indicando que o usuário `globalcode` com a senha `globalcode` poderá acessar todas as aplicações que estiverem acessíveis para a role `admin` e `manager` (que é exatamente o caso das aplicações **Tomcat Administration** e **Tomcat Manager**).

```
1  <?xml version='1.0' encoding='utf-8'?>
2  <tomcat-users>
3    <role rolename="tomcat"/>
4    <role rolename="manager"/>
5    <role rolename="admin"/>
6    <user username="tomcat" password="tomcat" roles="tomcat,admin,manager"/>
7    <user username="both" password="tomcat" roles="tomcat,role1,manager"/>
8    <user username="role1" password="tomcat" roles="role1"/>
9    <user username="globalcode" password="globalcode" roles="admin,manager"/>
10 </tomcat-users>
```


2.3 Estrutura de diretórios tipicamente utilizados em projetos web

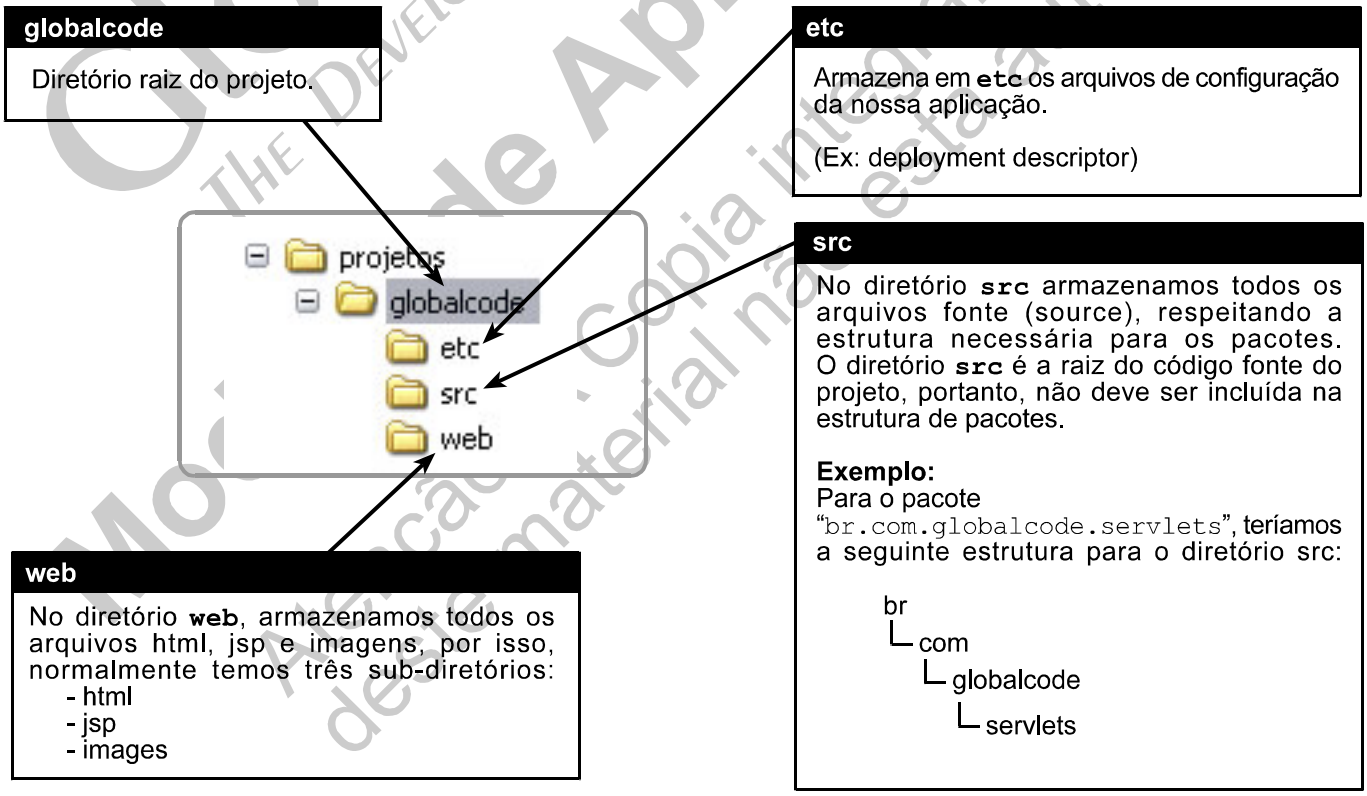
Normalmente trabalhamos com, no mínimo, duas estruturas de diretório:

- **Diretórios de desenvolvimento:** Nesta estrutura de diretórios armazenamos o código fonte do projeto e geralmente criamos o projeto na nossa IDE "apontando" para esta estrutura de diretórios.
- **Diretórios no container Web:** Esta estrutura de diretórios normalmente fica localizada dentro do container Web. Armazenamos aqui o código que será executado, não sendo necessário deixar os arquivos fontes nestes diretórios.

Diretórios de desenvolvimento

A estrutura de desenvolvimento é a estrutura que utilizaremos para criar nossos projetos, normalmente fora da estrutura de diretórios do Tomcat ou de qualquer outro servidor de aplicações.

Esta estrutura não é obrigatória, mas é usual, por isto estaremos utilizando esta estrutura em nossos laboratórios.



Anotações