

DESENVOLVIMENTO *WEB*



sagah⁺

Servidores *web* (Apache Tomcat)

Rafael Leal Martins

OBJETIVOS DE APRENDIZAGEM

- > Conceituar servidores *web*.
- > Caracterizar o servidor *web* Apache Tomcat.
- > Descrever a implementação de um servidor Apache Tomcat.

Introdução

O servidor *web* é um componente fundamental em um sistema *web*. No desenvolvimento *web* Java, eles são ainda mais importantes, pois servem como contêineres de Servlets Java. Tomcat é um servidor *web*, mais especificamente um contêiner de *servlets*. Ele foi desenvolvido pela Apache Software Foundation, é distribuído como livre e implementa as tecnologias Java Servlet e JavaServer Pages (JSP).

Neste capítulo, você vai conhecer os servidores *web* e seu funcionamento. Além disso, vai ver o servidor *web* Apache Tomcat, suas características e utilizações. Por fim, vai estudar a configuração do Apache Tomcat e alguns exemplos práticos.

O que é um servidor *web*

Um dos protocolos de aplicação mais usados na internet é o Hypertext Transfer Protocol (HTTP). Ele permite que clientes *web* (navegadores ou *browsers*, na maioria dos casos) consigam transferir conteúdo *web* (documentos HTML, imagens, *scripts*, vídeos, etc.) para servidores que hospedam esses recursos. Esses servidores são chamados de servidores *web*.

Servidores de arquivos, de bancos de dados, de *e-mails* e de internet usam diferentes categorias de *softwares*. Cada uma dessas aplicações pode acessar arquivos armazenados em servidores físicos e usá-los para diferentes propósitos. A função de um servidor *web* é hospedar o conteúdo de *sites* na internet. Ele transfere conteúdo de seu sistema de arquivos local para atender aos pedidos do cliente.

Um servidor *web* é um programa que recebe solicitações HTTP (*requests*) do cliente e envia alguma resposta (*response*). Para entender o funcionamento do servidor *web*, é necessário entender como funciona o protocolo HTTP.

Segundo Kurose e Ross (2013), uma mensagem de solicitação típica seria:

```
GET /site1/exemplo1.html HTTP/1.1
Host: www.algumsite.com.br
Connection: close
User-agent: Mozilla/5.0
Accept-language: pt-br
```

A primeira linha é denominada linha de requisição HTTP, enquanto as linhas seguintes são chamadas de linhas de cabeçalho. A linha de requisição tem três campos: método HTTP, local e identificação do recurso solicitado (URL, do inglês Universal Resource Locator) e a versão do HTTP utilizado pelo cliente. O campo do método pode assumir vários valores diferentes, como GET, POST, HEAD, PUT e DELETE. A maioria das mensagens de requisição HTTP emprega o método GET, usado quando o navegador requisita um objeto do servidor, que é identificado no campo URL. Nesse exemplo, o navegador está requisitando o objeto /site1/exemplo1.html. A versão é autoexplicativa; nesse exemplo, o navegador executa a versão HTTP/1.1.

Vamos examinar as linhas de cabeçalho do exemplo. A linha de cabeçalho Host: www.algumsite.com.br especifica o *host* (servidor) no qual o objeto está armazenado. A linha de cabeçalho Connection: close serve para o cliente dizer ao servidor que não deseja usar conexões persistentes, ou seja, que quer que o servidor feche a conexão após o envio do objeto requisitado. A linha de cabeçalho User-agent: especifica o tipo de navegador que está fazendo a requisição ao servidor. Nesse caso, o cliente é o Mozilla/5.0, um navegador Firefox. Por fim, o cabeçalho Accept-language: pt-br mostra que o usuário prefere receber uma versão em português brasileiro do objeto, se este existir no servidor. Se não existir, o servidor deve enviar a versão *default*. O cabeçalho Accept-language: pt-br é apenas um dos muitos cabeçalhos de negociação de conteúdo disponíveis no HTTP.

A resposta enviada pelo servidor teria o formato a seguir:

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 06 Sep 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Mon, 20 Sep 2021 16:11:03 GMT
Content-Length: 879
Content-Type: text/html
(dados...)
```

Essa mensagem de resposta tem três seções: uma linha inicial (linha de estado), seis linhas de cabeçalho e, em seguida, o corpo da entidade, que é a parte principal da mensagem e contém o objeto solicitado (*dados ...*). A linha de estado tem três campos: o de versão do protocolo, um código de estado e uma mensagem de estado correspondente. Nesse exemplo, ela mostra que o servidor está usando o HTTP/1.1 e que o servidor está enviando o objeto solicitado pelo cliente.

Nas linhas de cabeçalho, o servidor usa `Connection: close` para informar ao cliente que vai fechar a conexão após enviar a mensagem. A linha `Date:` indica a hora e a data em que a resposta HTTP foi criada e enviada pelo servidor. A linha `Server:` mostra que a mensagem foi gerada por um servidor *web* Apache. A linha `Last-Modified:` indica a hora e a data em que o objeto foi criado ou sofreu a última modificação. A linha `Content-Length:` indica o número de bytes do objeto que está sendo enviado. Por fim, a linha `Content-Type:` define que o objeto presente no corpo da mensagem é um texto HTML.



Exemplo

O servidor *web* Apache é um dos servidores *web* mais usados. Ele é um *software* de código aberto da Apache Software Foundation. Como qualquer outro servidor *web*, sua função é prover conteúdo *web* para os clientes. O Apache é um *software* multiplataforma. Portanto, ele funciona tanto em servidores Linux/Unix quanto em servidores Windows.

Quando um usuário deseja acessar um *site*, seu *browser* envia uma solicitação HTTP para o servidor, e o Apache devolve uma resposta com todos os arquivos solicitados (textos, imagens, etc.). O Apache tem uma estrutura baseada em módulos, que permitem que os administradores dos servidores ativem ou desativem suas funcionalidades. O Apache tem módulos para segurança (TLS/SSL), *cache*, reescritas de URL, autenticação de senhas, entre outros (APACHE, 2021).

Apache Tomcat

O Apache Tomcat é um contêiner Java Servlet de código aberto que implementa as APIs Java Servlet, JavaServer Pages (JSP) e WebSockets. O Tomcat é um projeto da Apache Software Foundation que foi lançado pela primeira vez em 1998, apenas quatro anos depois do próprio Java (APACHE Tomcat..., 2021). O Tomcat começou como uma implementação de referência para a primeira API Java Servlet e a especificação JSP. Um Java Servlet é uma classe Java que encapsula a lógica de código e negócios e define como as solicitações e respostas devem ser tratadas em um servidor Java. A JSP é uma tecnologia de renderização de visualização do lado do servidor. Como desenvolvedor, você escreve a página Servlet ou JSP e deixa o Tomcat lidar com o roteamento. Embora não seja mais a implementação de referência para essas tecnologias, a Tomcat continua sendo o servidor Java mais utilizado. O Tomcat é um servidor *web* que recebe o nome de contêiner, pois é usado principalmente para hospedar Servlets Java. Um servidor *web* típico, como o Apache ou o Nginx, é um servidor projetado para servir arquivos de conteúdo *web*, como HTML, Javascript, CSS etc.

Quando um aplicativo Java é compilado em JSPs, ele precisa de um contêiner Servlet para executar o código das páginas e retornar a saída para um navegador. O Tomcat é o contêiner no qual essas páginas são executadas. Ele é a implementação de referência de um contêiner de Servlet usado, na época, pela Sun Microsystems para implementar a especificação de Servlet. Isso garante que um aplicativo projetado para as especificações da API será executado no Tomcat sem problemas de compatibilidade. O Tomcat pode operar como um servidor autônomo, mas é otimizado para executar Servlets e JSPs dinâmicos. A Figura 1 mostra um acesso típico a esse tipo de conteúdo pelo servidor Tomcat.

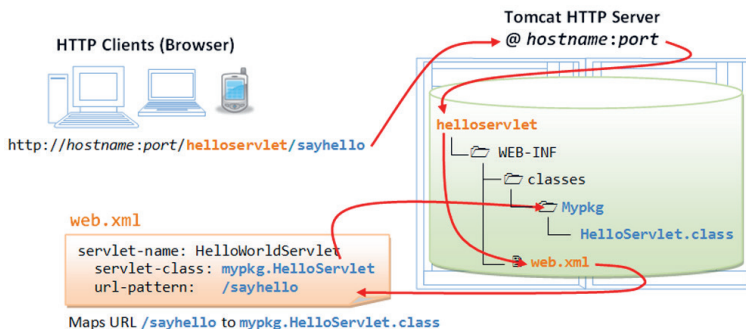


Figura 1. Funcionamento típico do Tomcat para Java Servlets.

Fonte: Shatnawi et al. (2017, p. 7).

Os clientes (*browsers*) usam solicitações HTTP estruturadas como endereços da *web* que fazem referência a páginas da *web* por meio de URLs relativos. As solicitações são tratadas por meio de páginas da *web* dinâmicas ou estáticas, que podem ser implementadas com base em diferentes tecnologias, como Servlet, JSP, HTML, etc.

Uma vez que o Tomcat recebe uma solicitação HTTP, ele procura em descritores de implantação (por exemplo, em um arquivo chamado `web.xml`) para identificar a página *web* correspondente relacionada à URL solicitada. Na Figura 1, vimos o processo de mapeamento de URL realizado pelo Tomcat. Há duas formas de definir URLs para páginas da *web*: por meio de um arquivo descritor `web.xml` ou por meio de anotações de implantação.

Na Figura 1, o arquivo `web.xml` mapeia a URL relativa acessada pelo cliente (`/sayhello`) para a classe Java (Servlet) a ser executada (`mypkg.HelloServlet`), cujo arquivo (`HelloServlet.class`) está armazenado em um diretório `WEB-INF` usado em aplicações JEE.

Em uma instalação padrão do Tomcat, há alguns diretórios-chave. Veja a seguir (APACHE Tomcat..., 2021).

- `/bin`: contém *scripts* de inicialização, desligamento, entre outros. Os arquivos `*.sh` (para sistemas Unix) são duplicatas funcionais dos arquivos `*.bat` (para sistemas Windows).
- `/conf`: diretório que contém arquivos de configuração. O arquivo mais importante aqui é o arquivo `server.xml`. É o arquivo de configuração principal para o Tomcat.
- `/logs`: os arquivos de registro estão aqui por padrão.
- `/webapps`: diretório para onde vão suas aplicações *web* quando são armazenadas.

Na documentação do Tomcat, há referências a duas propriedades; veja a seguir (APACHE Tomcat..., 2021).

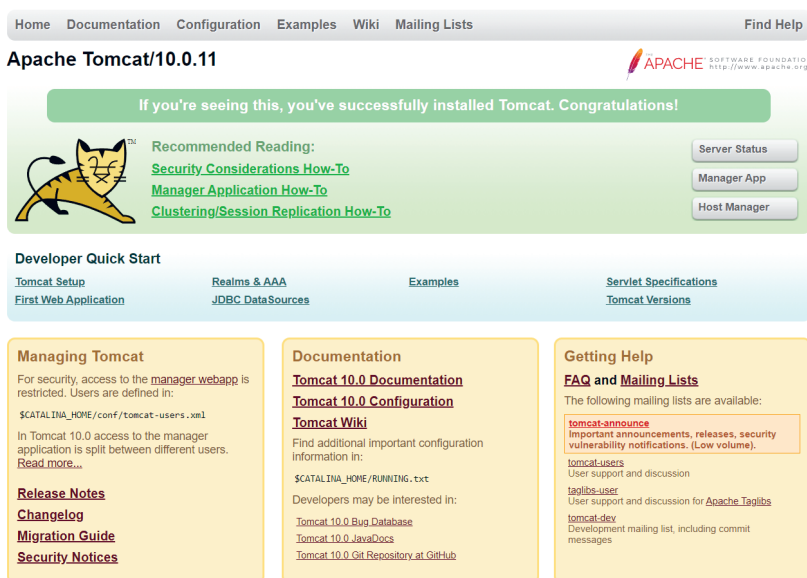
- `CATALINA_HOME`: representa o diretório raiz da instalação do Tomcat.
- `CATALINA_BASE`: representa o diretório raiz de uma configuração de tempo de execução de uma instância Tomcat específica. Se você quiser executar várias instâncias Tomcat em uma máquina, use essa propriedade.

Se você definir as propriedades para diferentes locais, o local `CATALINA_HOME` vai conter fontes estáticas, como arquivos `.jar` ou arquivos

binários. A localização `CATALINA_BASE` contém arquivos de configuração, arquivos de log, aplicativos implantados e outros requisitos de tempo de execução. A seguir, vamos estudar como é feita essa configuração na prática.

Configurando um servidor Tomcat


O servidor Tomcat pode ser baixado em seu site oficial (APACHE Tomcat..., 2021). Uma vez baixado, instalado e colocado em execução, é possível acessar sua página inicial, <http://localhost:8080>, que vai exibir a página da Figura 2.



Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/10.0.11 APACHE SOFTWARE FOUNDATION <http://www.apache.org/>

If you're seeing this, you've successfully installed Tomcat. Congratulations!

 Recommended Reading:
[Security Considerations How-To](#)
[Manager Application How-To](#)
[Clustering/Session Replication How-To](#)

Server Status
 Manager App
 Host Manager

Developer Quick Start
[Tomcat Setup](#) [Realms & AAA](#) [Examples](#) [Servlet Specifications](#)
[First Web Application](#) [JDBC DataSources](#) [Tomcat Versions](#)

Managing Tomcat
 For security, access to the [manager webapp](#) is restricted. Users are defined in:
`$CATALINA_HOME/conf/tomcat-users.xml`
 In Tomcat 10.0 access to the manager application is split between different users.
[Read more...](#)
[Release Notes](#)
[Changelog](#)
[Migration Guide](#)
[Security Notices](#)

Documentation
[Tomcat 10.0 Documentation](#)
[Tomcat 10.0 Configuration](#)
[Tomcat Wiki](#)
 Find additional important configuration information in:
`$CATALINA_HOME/RUNNING.txt`
 Developers may be interested in:
[Tomcat 10.0 Bug Database](#)
[Tomcat 10.0 JavaDocs](#)
[Tomcat 10.0 Git Repository at GitHub](#)

Getting Help
FAQ and Mailing Lists
 The following mailing lists are available:
[tomcat-announce](#)
 Important announcements, releases, security vulnerability notifications. (Low volume).
[tomcat-users](#)
 User support and discussion
[taglibs-user](#)
 User support and discussion for [Apache Taglibs](#)
[tomcat-dev](#)
 Development mailing list, including commit messages

Figura 2. Página inicial do servidor Tomcat em execução.

Fonte: Apache Tomcat... (2021, documento *on-line*).

Uma vez instalado, o Tomcat será armazenado em seu sistema local. Em um sistema Windows, por padrão, ele fica na pasta Arquivos de programa (Program files). Em um Mac ou Linux, pode estar na pasta `/user/var/opt` ou `/User/<>/Application`. Nesse diretório, você vai ver os diferentes diretórios e arquivos usados pelo Tomcat. Os principais são os seguintes.

- `bin`: onde ficam os arquivos binários e os *scripts* de inicialização da Tomcat.
- `conf`: local da configuração global aplicável a todos os *webapps*. A instalação padrão normalmente fornece um arquivo denominado `catalina.policy` para especificar a política de segurança das aplicações. Há também dois arquivos de propriedades (`catalina.properties` e `logging.properties`) e quatro arquivos XML de configuração, que são:
 - `server.xml`: arquivo de configuração principal do Tomcat;
 - `web.xml`: arquivo que contém os descritores globais de implantação de aplicativos *web*;
 - `context.xml`: arquivo com as opções globais de configuração específicas do Tomcat;
 - `tomcat-users.xml`: um banco de dados de usuários, senhas e papéis (*roles*) para autenticação e controle de acesso.

O diretório `conf` também tem um subdiretório para cada *engine* utilizada, como, por exemplo, Catalina. Você pode colocar as informações de contexto específicas do *host*, funcionando de forma semelhante ao `context.xml`, mas nomeando como `webapp.xml` para cada *webapp* no *host*.

- `lib`: diretório que mantém o arquivo JAR que está disponível para todos os *webapps*. A instalação padrão inclui `servlet-api.jar` (Servlet), `jasper.jar` (JSP) e `jasper-el.jar` (EL). Arquivos JAR externos podem ser colocados aqui, como o *driver* MySQL JDBC e JSTL.
- `logs`: contém os arquivos de *log* do Tomcat.
- `webapps`: é o diretório base de aplicativos *web* do *host* `localhost`.

A configuração padrão contida no arquivo `server.xml` é mostrada a seguir.


```

<?xml version="1.0" encoding="UTF-8"?>

<Server port="-1" shutdown="SHUTDOWN">

  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />

  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />

  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />

  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />

  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

  <GlobalNamingResources>

    <Resource name="UserDatabase" auth="Container"

      type="org.apache.catalina.UserDatabase"

      description="User database that can be updated and saved"

      factory="org.apache.catalina.users.MemoryUserDatabaseFactory"

      pathname="conf/tomcat-users.xml" />

  </GlobalNamingResources>

  <Service name="Catalina">

    <Connector port="8080" protocol="HTTP/1.1"

      connectionTimeout="20000"

      redirectPort="8443" />

    <Engine name="Catalina" defaultHost="localhost">

      <Realm className="org.apache.catalina.realm.LockOutRealm">

        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"

          resourceName="UserDatabase"/>

      </Realm>

      <Host name="localhost" appBase="webapps"

        unpackWARs="true" autoDeploy="true">

        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

          prefix="localhost_access_log" suffix=".txt"

          pattern="%h %l %u %t &quot;%r&quot; %s %b" />

      </Host>

    </Engine>

  </Service>

</Server>

```

O arquivo XML define toda a configuração dentro de um conjunto de *tags* `<Server></Server>`, contendo propriedades de porta e desligamento (*shutdown*). Dentro dessa seção, são definidos vários *listeners*, componentes que “escutam” e respondem a certos eventos. Em seguida, vem a seção `GlobalNamingResources`, que define recursos que ficam disponíveis globalmente para o serviço. No exemplo, define-se o recurso `UserDatabase`, indicando o arquivo xml a ser usado como banco de dados de usuários do servidor Tomcat.

Um *Service* associa um ou mais conectores (*Connectors*) a uma *engine*. Nesse caso, cria-se um *Connector* para a porta 8080 TCP, com tempo de *timeout* de 20.000 ms, podendo ser direcionado para 8.443 caso se deseje utilizar HTTPS. Isso está associado à *engine* Catalina, executando no *host* *localhost*. *Realms* são usados como bancos de dados de usuários, senhas e papéis. Eles podem ser associados a *engines*, *hosts*, etc.

A seção `<host>` define um *host* virtual dentro do sistema. Cada *site* em um servidor *web* tem seu próprio *host* virtual. Nessa *tag*, define-se a propriedade *appBase*, que define o diretório onde devem ficar as aplicações *web* a serem implantadas em um virtual *host*. É possível especificar um nome de caminho absoluto ou um nome de caminho relativo ao diretório `$CATALINA_BASE`. Se não for especificado, o padrão de *webapps* será usado.

O arquivo *web.xml* também tem um papel fundamental na configuração de um servidor Tomcat. É nele que configuramos os mapeamentos dos Servlets para os caminhos de URL acessados pelos clientes. Os elementos principais desse arquivo são as seções `<servlet>` e `<servlet-mapping>`, que indicam ao Tomcat qual classe Java contém o Servlet a ser “servido” aos clientes e o mapeamento da URL de acesso para executar esse Servlet. Um exemplo dessas duas seções é mostrado a seguir.

```
<servlet>
  <servlet-name>hello</servlet-name>
  <servlet-class>teste.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>hello</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```

Nesse exemplo, define-se o Servlet *hello*, contido na classe *teste.HelloServlet*, além do mapeamento da URL */hello* a esse Servlet. Assim, quando um usuário acessar a URL `http://localhost:8080/hello`, o Tomcat vai executar a classe Java mapeada e vai retornar o resultado para o usuário.

Como visto ao longo do capítulo, Tomcat é um *servlet* Java que define como as solicitações de servidor são tratadas, fornecendo um ambiente de servidor *web* HTTP para executar o código das aplicações Java. Ele é leve, flexível, seguro e conta com uma documentação completa e disponível na internet, o que facilita sua instalação, configuração e uso.

Referências

APACHE: http server project. *The Apache Software Foundation*, 2021. Disponível em: <http://httpd.apache.org/>. Acesso em: 16 out. 2021.

APACHE Tomcat 10: documentation index. *The Apache Software Foundation*, 2021. Disponível em: <http://tomcat.apache.org/tomcat-10.0-doc/index.html>. Acesso em: 16 out. 2021.

KUROSE, J. F.; ROSS, K. W. *Redes de computadores e a Internet: uma abordagem top-down*. 6. ed. São Paulo: Pearson, 2013.

SHATNAWI, A. et al. *How to implement dependencies in server pages of JEE web applications*. Montreal: Latece Laboratoire/UQAM, 2017. (Rapport de recherche Latece, n. 1).



Fique atento

Os *links* para *sites da web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integridade das informações referidas em tais *links*.

Conteúdo:

sagah⁺