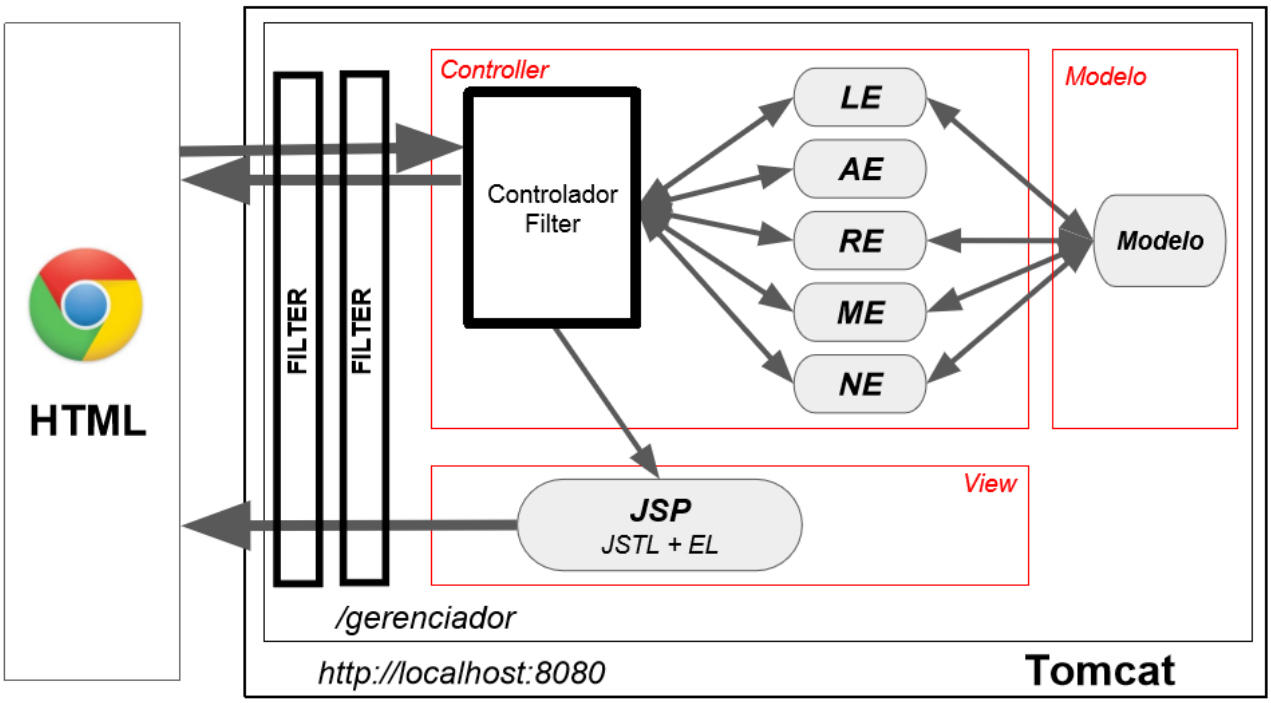
02

O que é um Web Service?

Transcrição

Bem vindo de volta ao nosso curso sobre o mundo Servlet. Repare que nosso projeto avançou muito e já possui uma boa estrutura - a mesma estrutura que será utilizada em seus projetos no futuro, somente trocando o **controlador** que criamos por um controlador do mercado (como o Spring MVC, o mais popular deles).

Esses controladores são muito mais sofisticados e ajudam a ler, converter e validar os parâmetros, além de popular o modelo.



Em projetos reais, o **modelo** (como acessar o banco de dados ou camada de persistência, o que não é o nosso foco nesse curso) também poderá ser um pouco diferente. Isso vai depender se você usará JCB puro para escrever o SQ manualmente, ou se usará uma biblioteca de mais alto nível, como JPA com

Hibernate, para gerar o SQL automaticamente. Nesses casos, o modelo terá mais classes especializadas que acessam o banco de dados.

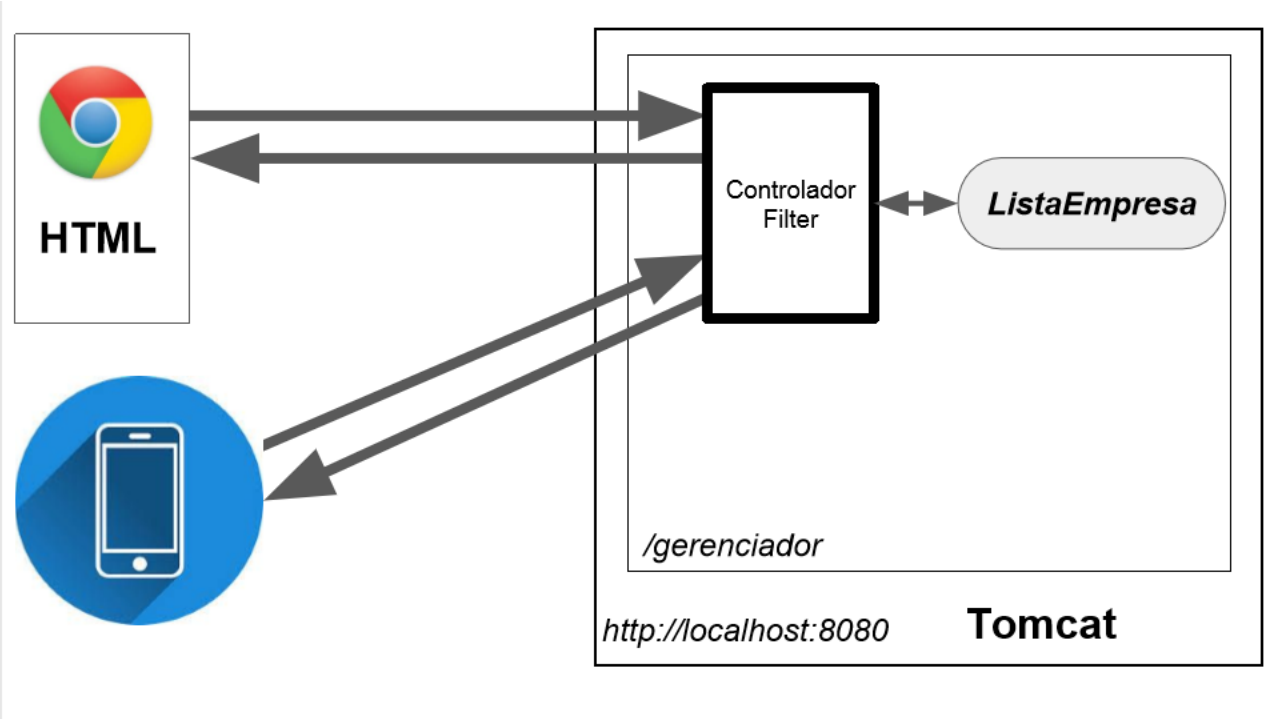
Em relação ao **JSP**, seu uso no dia-a-dia será basicamente o que vimos nesse curso. Ainda entrarão bibliotecas como o jQuery (para fazer o método AJAX), portanto é necessário aprender um pouco de javascript para se dar bem nesse cenário. Além disso, talvez você se depare com uma ou outra *tag* que ainda não conheça - o Spring MVC, por exemplo, tem algumas *tags* específicas que devem ser aprendidas para simplificar o uso dos controladores.

O Spring MVC e o VRaptor também facilitam muito o uso das ações, já que neles é possível agrupar várias ações em uma única classe, além de não precisarem de uma interface.

Os **filtros**, da forma que os conhecemos, são bastante úteis e você irá precisar deles em alguns momentos. No entanto, o Spring MVC já possui alguns filtros padrão para fazer, por exemplo, autenticação e autorização.

É importante entender os conceitos por trás do fluxo que construímos e da separação das camadas (**MVC**), que são fundamentais para qualquer aplicação web - inclusive, nosso modelo de aplicação se aplica a outras linguagens de programação importantes na carreira de um desenvolvedor, como .NET e PHP.

Repare que na nossa aplicação, até o momento, o `ControladorFilter` (ou o Servlet que criamos) devolve um *redirect* para o navegador, que automaticamente enviará uma requisição, ou devolve um HTML. Porém, tudo que o JSP devolve foi feito pensando no navegador. Mas e se o cliente que usa nossa aplicação não for um navegador, ou nem mesmo entenda HTML - por exemplo, um celular?



Tudo bem, os celulares têm navegadores. Porém, eles também têm várias outras aplicações que se comunicam sem terem um navegador embutido - por exemplo, qualquer jogo irá acessar um servidor e mostrar dados da conta do usuário naquele jogo sem o intermédio de um navegador.

A ideia agora é termos uma aplicação no celular que mostre a lista de empresas, mas não através de HTML (que contêm várias outras informações sobre a estrutura da página) - ou seja, precisaremos apresentar os dados da nossa aplicação de maneira diferente.

Para isso existem dois formatos bem famosos: **JSON** e **XML**. Ambos os formatos se preocupam em apresentar apenas os dados, sem nenhuma informação sobre a estrutura desses dados (como `<body>` e `<br>`, no caso do HTML).

Tanto JSON quanto XML são grandes *strings* com dados. Dessa forma, diferentes clientes podem apresentar esses dados de maneira distinta - sempre continuando com o protocolo HTTP, mas devolvendo um JSON ou um XML ao invés de devolver um HTML.

Também existem bibliotecas ("*frameworks*") JavaScript que fazem o MVC dentro do próprio navegador, como **Angular**, **React** e **Vue.js**. Dessa forma, o navegador carrega (praticamente) todo o HTML da aplicação logo de início. Em seguida, o servidor devolve apenas os dados, e a biblioteca gera o HTML dentro do navegador. Esses frameworks também precisam dos dados em formato JSON ou XML, pois não consomem HTML.

> Se você não entendeu, não tem problema! Nós temos vários cursos e formações sobre essas bibliotecas aqui na Alura (https://cursos.alura.com.br/formacoes)!

A partir desse momento nós criaremos nosso primeiro serviço web - ou "*Web Service*": uma funcionalidade que conseguimos chamar remotamente através do protocolo HTTP, usando, por exemplo, o Java Servlet, e que devolve os dados em um formato genérico, como JSON ou XML.

Quando utilizamos Angular ou outras bibliotecas JavaScript, é mais comum encontrarmos o JSON (que é um formato muito popular hoje em dia). Já em aplicações corporativas mais formais, é comum encontrar o XML.

> Dica: vale a pena conferir o nosso curso de Protocolo HTTP (https://cursos.alura.com.br/course/http-fundamentos), no qual também apresentamos conceitos de JSON, XML, API, criação de Web Services e serviços Rest. Quando mais você entende sobre protocolo HTTP, melhores serão suas aplicações Web, seus serviços Web e a sua carreira.

No próximo vídeo começaremos a implementar nosso primeiro serviço. Até lá!