DESENVOLVIMENTO

20 NOV, 2013

Single Page Applications e outras maravilhas da web moderna

33923 visualizações

ALEXANDRE ROCHA MARCONDES (HTTPS://IMASTERS.COM.BR/PERFIL/ALEXANDRCONDES)
Tem 1 artigos publicados com 33923 visualizações desde 2013

ALEXANDRE ROCHA MARCONDES (HTTPS://IMASTERS.COM.BR/PERFIL/ALEXANDREROCHAMARCONDES)

1

atualmente trabalha na Dextra, como Desenvolvedor Senior e conta com mais de 13.000 horas de experiência em projetos ágeis, mais de 40.000 horas de experiência em desenvolvimento de sistemas e mais de 1.000 profissionais treinados em cursos e palestras, além de mais de 5 anos de experiência em inovação e empreendedorismo.

LEIA MAIS (HTTPS://IMASTERS.COM.BR/PERFIL/ALEXANDREROCHAMARCONDES)

O desenvolvimento para web tem mudado nos últimos anos e aplicações de apenas uma página com grande parte de seu código no cliente, em JavaScript, tem se tornado cada vez mais comuns. Este artigo explora o conceito de Single Page Application (SPA) e as ferramentas disponíveis para seu desenvolvimento.

Este é o primeiro artigo de uma série que irá desvendar os segredos do desenvolvimento moderno para web.

O estado do desenvolvimento web atual

A Internet tem mudado muito nos últimos anos e várias tecnologias surgiram, consolidadas por navegadores mais padronizados e links mais velozes. Enquanto na década de 90 as páginas eram basicamente estáticas, com muitas imagens e texto, na primeira década dos anos 2000 o Flash foi a tecnologia da vez e o vídeo expandiu na web. O que chamamos de web moderna ou “web 2.0”

faz uso de muito JavaScript, transferência de dados em background (por meio de AJAX e WebSockets) e aplicações interativas como o Gmail, Facebook, Twitter que disponibilizam APIs para desenvolvedores (veja exemplos em chromeexperiments.com (http://www.chromeexperiments.com/)).

As aplicações “web 2.0” têm algumas características que as diferenciam das anteriores:

→ São auto-contidas e tem um objetivo principal (muitas vezes fazem uso de uma técnica chamada Single Page Application);

→ Elas usam muito do HTML 5 (html5readiness.com (http://html5readiness.com/)) para ter a sensação de ser uma aplicação nativa (ou desktop);

→ São desenvolvidas pensando-se que, ao ficar offline, o usuário possa continuar a operar a aplicação (como numa aplicativo desktop);

→ Elas reconhecem o dispositivo em que estão sendo executadas (celular, desktop, laptop, etc);

→ Apresentam uma performance bem maior que as aplicações Web antigas e seu uso é muito mais agradável;

→ O cliente (navegador) se comunica de forma assíncrona com o servidor e muitas vezes o servidor envia dados sem haver uma requisição (push de conteúdo) usando AJAX, WebSockets ou Comet;

→ Apresentam um estilo de navegação sem links e elementos de navegação tradicionais da Web aparentes;

→ A aplicação é projetada num estilo client-side.

O papel do JavaScript na Web moderna

O JavaScript evoluiu muito desde sua integração nos browsers. Inicialmente ele era visto apenas como uma ferramenta para fazer botões funcionarem ou algo que os Web Designers precisavam fazer para criar menus. Com a evolução dos browsers e da linguagem HTML, o JavaScript ganhou força e muito poder de processamento.

A “web 2.0” veio trazer para o JavaScript uma posição de destaque. Segundo o ranking de linguagens da TIOBE (bit.ly/180tQck), uma das mais respeitadas empresas nesta área, o Javascript passou da 32a posição em 1998 para a 9a posição neste ano e 2013 e muito disso deve-se à Web moderna.

Javascript no Desktop

O JavaScript ganhou tanta força nos últimos anos que ganhou formas de ser executado sem browsers, no desktop, como um programa comum. O projeto Node.js é um dos grandes motores desta nova tecnologia. Ele permite não apenas executar aplicações JavaScript no desktop, mas pode também atuar como servidor HTTP e assim podemos desenvolver aplicações tanto com o cliente quanto o servidor na mesma linguagem.

Single Page Applications (SPA)

Com mais e mais profissionais estudando e usando JavaScript, grandes empresas como Google, Yahoo!, Apple e Microsoft passaram a desenvolver padrões e técnicas para incrementar a performance de seus scripts, separar e isolar porções de código e fazer uso de orientação à objeto mais avançada na linguagem. Uma das consequências dos estudos e experimentos realizados foram aplicações inteiramente contidas no browser que não precisam fazer requisições de novas páginas no servidor. As melhores práticas aprendidas foram transformadas no conceito de Single Page Applications, ou SPAs (saiba mais aqui (http://www.html5rocks.com/webappfieldguide/toc/index/)).

As SPAs são aplicações completas, desenvolvidas em JavaScript, que funcionam quase como se estivessem sendo executadas nativamente no desktop. O Google foi pioneiro nesta tecnologia e o mundo o seguiu. Atualmente, a maior parte das aplicações “web 2.0” usam este modelo: o Gmail, a busca do Google, o Google Drive, Facebook, o Twitter, o FourSquare, o Instagram, blogs, sites corporativos, dentre outros.

Práticas modernas de desenvolvimento web

As práticas usadas no desenvolvimento de aplicações Web modernas envolvem criar uma aplicação completa no cliente, para isso a maior parte usa o padrão Model-View-Controller (MVC), Model-View-Presenter (MVP) ou Model-View-ViewModel (MVVM). Algumas das vantagens de se usar um destes modelos de desenvolvimento são:

→ Facilidade de manutenção;

→ Possibilidade de se ter várias Views para um modelo de dados;

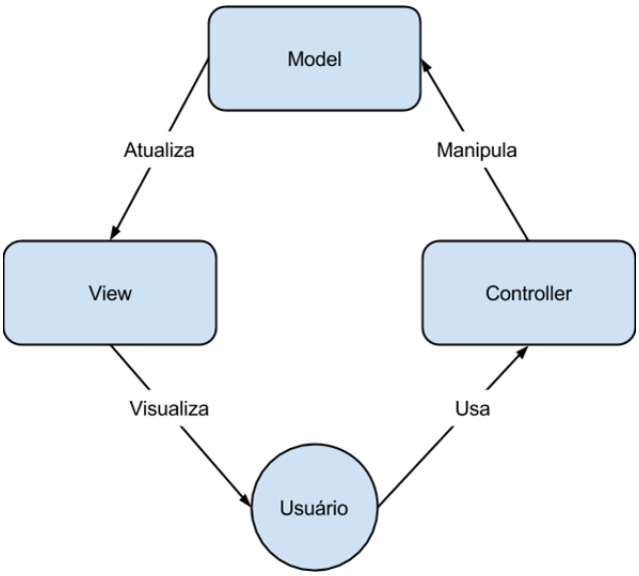
→ Separação bem clara entre interface visual e persistência de dados;

→ Isolamento das regras de negócio;

→ Possibilidade de alterar a forma com que uma interface para o usuário se comporta apenas alterando o Controller;

→ Model-View-Controller (MVC).

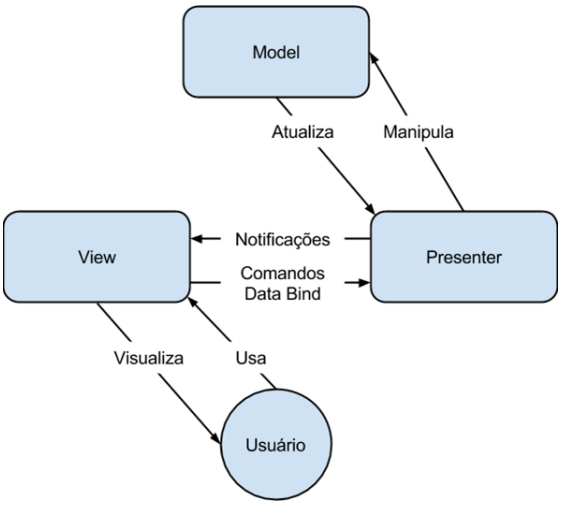
É um padrão de arquitetura de aplicações que separa a representação das informações das interações do usuário com estas informações. O Modelo representa as informações no sistema (módulos ou classes JavaScript), a View é a forma com que estes dados são apresentados ao usuário (interface gráfica ou página que o usuário vê) e o Controller é o intermediador entre as duas partes, sendo invocado pelas interações que o usuário faz na View.



(https://imasters.com.br/?attachment_id=50714)

Model-View-Presenter (MVP)

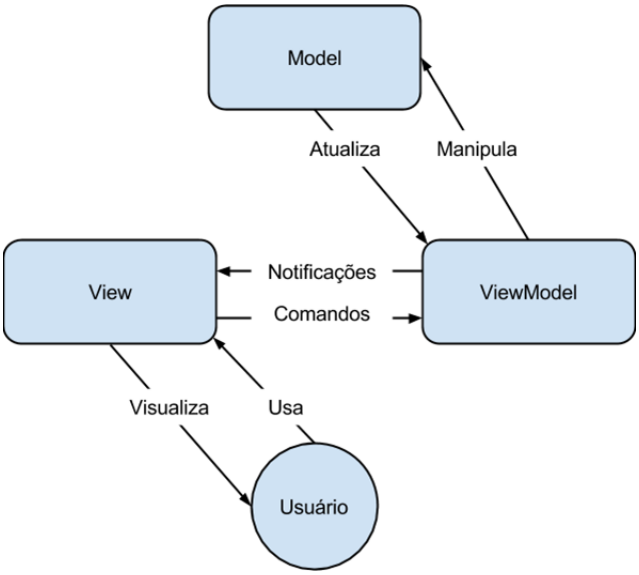
Muito similar ao MVC, este modelo de desenvolvimento criar um fluxo de eventos linear onde apenas a View interage com o usuário e o Presenter controla a execução dos comandos e a manipulação de dados. Este modelo é uma evolução do MVC no que diz respeito à separação de responsabilidades e consequentemente, do código em si.



(https://imasters.com.br/?attachment_id=50716)

Model-View-ViewModel (MVVM)

Este modelo é uma derivação do MVP em que o Presenter fica dividido em binder e ViewModel. O binder geralmente é realizado automaticamente pelo Framework disponibilizado pela solução.



(https://imasters.com.br/?attachment_id=50717)

Podemos dividir os projetos de Web moderna com Single Page Application em dois grandes grupos: os que usam bibliotecas em seu desenvolvimento e os que usam Frameworks em seu desenvolvimento.

	Views	Roteamento de URL	Arm
AngularJS	Templates baseados em DOM mandatórios	Opcional	Opci
Batman	Templates baseados em DOM mandatórios	Mandatório	Man
Ember	Templates baseados em strings mandatórios	Mandatório	Cust
Backbone	De terceiros	Opcional	Cust
Knockout	Templates baseados em DOM opcionais	De terceiros	De t
Spine	Templates baseados em strings mandatórios	Opcional	Opci

Frameworks

Os Frameworks te dão uma infraestrutura e você recebe em troca várias operações comuns que são realizadas por ele de forma transparente. Geralmente envolvem além de uma forma específica de codificar, padrões de nomes, estruturas de arquivos ou padrões específicos de implementação que devem ser seguidos.

AngularJS

O Angular.js é um framework muito utilizado. Ele é mantido e usado internamente pelo Google e é licenciado usando MIT. Ele usa bastante atributos HTML customizados para fazer o data binding.

Ele provê o modelo parecido com Model-View-ViewModel, templates baseados no DOM que podem ter observers, roteamento básico de URL e persistência de dados. Existe um plugin do Chrome para auxiliar no debug que permite explorar os modelos e um plugin para o framework de testes Jasmine.

Conceitualmente ele é um polyfill[1] (/Users/Mariana%20Anselmo/Downloads/iMasters-SinglePageApplicationseoutrasmaravilhasdaWebmoderna.docx#_ftn1) entre o que os navegadores podem realizar atualmente e o que eles farão nativamente em alguns anos (binding declarativo e observers) para que possamos codificar desta maneira desde já.

Este framework não apresenta impacto na arquitetura do seu servidor ou na nomenclatura e organização dos seus arquivos. Você pode até mesmo usá-lo em uma pequena sessão da página, não há a necessidade de implementar na página toda – pode ser encontrado em angularjs.org (http://angularjs.org/).

Ember

Este projeto está tendo muita evolução, o que faz com que o seu código seja atualizado quase que diariamente, o que não o torna pronto para produção.

Ele é desenvolvido por Yehuda katz, um dos contribuidores do jQuery e do Rails, pela empresa dele e pela equipe do Ember. Este Framework tem tudo o que você precisa para construir uma aplicação Web ambiciosa e tem seu licenciamento com MIT. É o maior destes Frameworks, tanto em funcionalidades quanto em quantidade de código. Ele tem controles que foram muito bem pensados para decompor a hierarquia da página e que se ligam bem com um sistema de roteamento baseado em máquinas-de-estado. O Ember.Data é uma biblioteca de acesso à dados bem sofisticada.

O Ember foi inspirado pelo Rails e pelo Cocoa e tem regras bem definidas sobre nomes de arquivos, URLs e etc, mas tudo isso pode ser sobrescrito se você quiser. O projeto apresenta templates para Ruby on Rails, mas você pode usar outras plataformas de servidor se codificá-las manualmente. O Framework apresenta soluções comuns para problemas comuns, o que faz com que você possa pensar no que é único na sua aplicação – pode ser encontrado em emberjs.com (http://emberjs.com/).

Batman

Este projeto é mantido pela equipe da empresa de plataforma de e-commerce Shopify e é um MVC em JavaScript quase que exclusivamente para desenvolvedores Rails e CoffeeScript e é licenciado com MIT. É o Framework mais rígido com suas convenções e você é obrigado a usar o esquema de arquivos e URL dele ou é melhor escolher outro Framework. No entanto, ele é bem completo e tem models, views e controllers extremamente ricos, que contam com observers. Se você usa Rails e CoffeeScript você estará em casa usando o Batman e seu modelo de templates baseados em DOM – pode ser encontrado em batmanjs.org (http://batmanjs.org/).

Bibliotecas

As bibliotecas se integram com a estrutura que já existe no seu projeto e proveem funcionalidades específicas. Esta opção envolve muito mais codificação, mas permitem uma flexibilidade muito maior de implementação ou opções de troca de componentes.

Backbone

O Backbone está em produção por mais de dois anos e é uma biblioteca que oferece roteamento, persistência e Model-View em JavaScript. É uma das menores bibliotecas disponíveis e está publicada com o licenciamento MIT. Ela é mantida por Jeremy Ashkenas e pela equipe do DocumentCloud.

Ela não tem estruturas grandes, mas disponibiliza o mínimo necessário para que você construa a sua infraestrutura. O Backbone foca na integração model/REST e na estrutura para se ter o modelo MVC, mas você deve implementar o seu próprio mecanismo de renderização das Views. Se o seu projeto não usa o padrão RESTFul provavelmente você terá de reescrever os métodos .sync e .parse .

Por ser uma biblioteca pequena ela é de fácil entendimento e implementação, não apresenta impacto na estrutura de arquivos ou do servidor. Você tem liberdade para integrar com partes da página ou um todo – pode ser encontrado em backbonejs.org (http://backbonejs.org/).

Knockout

O Knockout é o oposto do Backbone e também está em produção há dois anos. Enquanto o Backbone focou no Model, o Knockout usa MVVM e é focado na View. Ele tem wrappers observable para propriedades em JavaScript e usa o atributo data-bind para fazer o binding das propriedades no seu HTML. Se você quer o binding de Views do Knockout e o binding de Models do Backbone, você pode usar o KnockBack (http://kmalakoff.github.io/knockback/), que combina ambas as bibliotecas.

O Knockout é mantido por uma equipe de três pessoas e tem um licenciamento MIT. A biblioteca é focada em UI rica com templates baseados em DOM com bindings declarativos, modelos com observer e detecção automática de dependências. Ela não é estrita com roteamento de URL e acesso à dados, não causa impacto na arquitetura de servidor ou na estrutura de arquivos. A grande vantagem no quesito UI é que a biblioteca é compatível até com o IE 6 e pode ser aplicada em apenas partes da página – pode ser encontrada em knockoutjs.com (http://knockoutjs.com/).

Spine

Esta biblioteca MVC é pequena e bem simples, mas se integra muito bem com o PhoneGap e tem excelentes recursos para desenvolvimento de WebApps. Ele é muito similar ao Backbone, tem Controllers explícitos e é escrito em CoffeeScript.

Ela é mantida por Alex MacCaw e foi criada à partir de um exemplo criado para um livro da O'Reilly e cresceu tornando-se um projeto Open Source – pode ser encontrada em spinejs.com (http://spinejs.com/).

Conclusão

Criar um site moderno, usando as novas tendências e tecnologias exige um bom conhecimento de JavaScript e a dedicação para aprender um novo modelo de arquitetura, onde o navegador não é mais apenas uma janela para a visualização da aplicação, mas parte ativa e essencial no processamento, manipulação e controle das informações. Antigamente os navegadores atuavam como terminais burros na Web com um design melhorado, atualmente eles são parte essencial do processamento distribuído e o servidor tornou-se um repositório de informações e algoritmos complexos.

As escolhas são simples:

- → Se você quer criar sua própria estrutura, o Backbone pode ser uma boa opção como base;
- → O Knockout se destaca se você quer uma interface rica;
- → Se você quer desenvolver para Ruby on Rails, o Ember e o Batman são excelentes opções;

→ Se você quer integrar com o PhoneGap, o Spine pode ser a sua melhor opção;

→ Se você quer o que tem de mais moderno e quer fazer debug no Chrome, o AngularJS te oferece tudo isso.

---

[1] (/Users/Mariana%20Anselmo/Downloads/iMasters-SinglePageApplicationseoutrasmaravilhasdaWebmoderna.docx#_ftnref1) polyfill: uma biblioteca que nivela os recursos de um navegador. Especialmente usada para dar recursos do HTML 5 e CSS 3 para navegadores antigos e para o Internet Explorer. Geralmente a biblioteca verifica se aquele recurso está disponível e, caso não esteja, ele provê uma implementação padrão.

SAIBA MAIS
(HTTPS://IMASTERS.COM.BR/PERFIL/ALEXANDREROCHAMARCONDES)

Alexandre Rocha Marcondes

1 Artigo(s)

atualmente trabalha na Dextra, como Desenvolvedor Senior e conta com mais de 13.000 horas de experiência em projetos ágeis, mais de 40.000 horas de experiência em desenvolvimento de sistemas e mais de 1.000 profissionais treinados em cursos e palestras, além de mais de 5 anos de experiência em inovação e empreendedorismo.