# Navigation

* Hashbangs, Hashes: Normalmente los hashes los utilizamos dentro de un anchor link: ejm miarticulo.html#capitulo3 para que luego de hacer click y cargue la nueva página, no se muestre el inicio de la página sino desde donde comienza el contenido del capitulo3. Podemos hacerlo mismo para navegar en una misma página sin recargarla.

Ahora que tenemos una aplicación que funciona bien cargando el contenido utilizando AJAX podemos aprovechar estas urls con hash para brindar una navegación a nuestra aplicación: a nivel de cliente podemos detectar cada vez que se haya producido un cambio en el hash de la url, entonces si un usuario escribe en el navegador una url con hash, podemos obtener cual de ese hash y cargar el contenido por AJAX directamente.

¿Qué logramos con todo esto? Logramos que nuestro contenido que es cargado con AJAX pueda ser identificado a través de una URL. A las urls que sirven para este propósito se les llama normalmente AJAX URLS.

Qué son los hashbangs? Como ya es posible contar con urls para contenido dinámico, google permite que ese contenido pueda ser indexado por su buscador, pero con unas cuantas condiciones: la primera de ellas que si queremos que un contenido por ajax sea indexado tenemos que indicarlo colocando un signo de admiración o bang al lado del hash, el resto de condiciones ya son más complejas que requieren tener mucho cuidado con la implementación y salen de nuestro tema principal.

Un clásico ejemplo sobre la utilización de hasbangs es Twitter.

Si bien nosotros podríamos implementar manualmente todo este tema de los hash, existen varias librerías que nos permiten hacer esto de manera más sencilla.

* Routing: Seguramente los que han trabajado con Rails, sympony, play framework me entienden. Cuando hablamos de routing nos referimos en este caso a una tabla en la cual definimos varias urls y para cada urls un handler, de tal manera que apenas se ingrese o modifique la url, se buscará en esta tabla algún campo que coincida con la nueva url y se ejecutará su handler.
* HTML 5 History, nos ofrece un API para manipular la historia de navegación desde el cliente, parte de esa api ya esta disponible desde versiones anteriores y lo que se ha hecho es agregar más funcionalidad para poder utilizar AJAX con las URLS sin necesidad de los hash. History.js nos permite utilizar HTML5 History más fácilmente y si el navegador no soporta HTML5 genera automáticamente urls con hash.

# Offline Data

Podría mostrarse primero creando un servicio odata getProduct, luego cambiarlo filtrando directamente al GetProducts. No olvidar que luego de filtrar con odata es products[0]

Consideraciones luego de pasar a upshot:

* 2 Urls de imágenes ahora debe llevar paréntesis en product.Name()
* En el ChooseProductViewModel, en el método goToPlacerOrder, ahora es product.Id()