Client-Side Web Refactoring Framework

Generic Refactorings

Los refactoring genéricos son aplicaciones que extienden el framework (utilizando el punto de extensión *AbstractGenericRefactoring*) y que contienen toda la lógica necesaria para aplicar un tipo de adaptación. Los refactorings genéricos no conocen a priori qué DOM se va a manipular, sino que solo publica una API que permite especificárselo cuando se quiere aplicar tal refactoring a un sitio Web. Esta API es, simplemente, un conjunto de clases (al menos una) y los métodos que estas tengan definidos.

Desarrollo

Los refactorings genéricos se desarrollan extendiendo la clase AbstractGenericRefactoring y redefiniendo algunos métodos como *#adaptDocument,* que es el método que se ejecutará para realizar modificaciones sobre un DOM.

```
SplitPage.prototype = new AbstractGenericRefactoring();

SplitPage.prototype.adaptDocument = function(doc){
        ….
}
```

Para desarrollar un refactoring genérico, es necesario tener conocimientos sobre desarrollo OO con JavaScript (https://developer.mozilla.org/es/docs/Introducci%C3%B3n_a_JavaScript_orientado_a_objetos). La cantidad de clases y la distribución de responsabilidades queda a criterio de quien define el refactoring genérico.

Sin embargo, es necesario especificar qué objetos pertenecen a la API del refactoring genérico, y para eso, debe definirse la variable *exportedObjects* como en la siguiente sentencia (el ejemplo esta basado en el refactoring SplitPage proporcionado):

```
var exportedObjects = {"GenericRefactoring":SplitPage,"SplitedSection":SplitedSection, "StaticLink":StaticLink};
```

En este arreglo asociativo se agregan los objetos que deben ser *conocidos* por quién vaya a utilizar este refactoring genérico. La la clave “GenericRefactoring” debe asignarse la clase que representa al refactoring genérico, es decir, la clase que hereda de AbstractGenericRefactoring, y este es el único par *clave:valor* mandatorio. El resto de los pares *clave:valor* corresponden a otras clases que van a ser necesarias para *instanciar* el refactoring genérico, por ejemplo mediante la creación de un refactoring instanciado.

Además, otra variable *metadata* permite especificar información sobre el refactoring:

```
var metadata = {"author":"Sergio Firmenich",
                "name":"Split Page",
                "description":"This generic refactoring splits a complex page into several smaller ones and add a
                               menu for accessing each of these sections.",
                "id":"splitpage-sfirmenich" };
```

Instantiated Refactorings

Refactorings instanciados son artefactos que mantienen una instancia de un refactoring genérico y que relacionan esa instancia con un conjunto de URLs (basadas en expresiones regulares) sobre las que esa instancia del refactoring genérico se aplicará.

Desarrollo

Para desarrollar un refactoring instanciado se debe heredar de la clase *AbstractInstanceRefactoring*, el punto de extensión del framework para este tipo de artefactos.

```
GMailListSplitWrapper.prototype = new AbstractInstanceRefactoring();
```

Al igual que con un refactoring genérico, algunos métodos deben ser redefinidos en la clase que hereda de *AbstractInstanceRefactoring*:

```
GMailListSplitWrapper.prototype.setTargetURLs = function(){
        this.addTargetURL(/https:\/\/mail.google.com\/mail\/h\/[\w]*\//);
        this.addTargetURL(/https*:\/\/mail.google.com\/mail[\w|\W|0-9|\/]*\/h\/[\w]*\/&?(st=)?[0-9]*/);
};
GMailListSplitWrapper.prototype.initialize = function(language){
        var refactoring = new SplitPage.SplitPage("GMAIL WebMail");
        var folders = new SplitPage.SplitedSection("Carpetas", refactoring);
        folders.addElement("html/body/table[2]/tbody/tr/td[1]/table[1]");
        this.abstract_refactoring = refactoring;
}
```

En el código fuente de ejemplo que se muestra arriba, puede apreciarse la definición de dos métodos ***#setTargetURLs*** y ***#initialize***. El primero, #***setTargetURLs*** requiere que sea redefinido porque es en este método donde el refactoring instanciado declara los sitios Web a los que ese refactoring aplica. Para hacerlo, puede enviarse a sí mismo el mensaje ***#addTargetURL*** con un parámetro que es una expresión regular. Pueden agregarse tantas expresiones regulares como sea necesario.

En el segundo método, ***#initialize***, es donde se hace efectiva la instanciación de un refactoring genérico y por ende, donde se utiliza la API del refactoring genérico que se desea instanciar. Los refactorings genéricos están en el *scope* del refactoring instanciado por medio de paquetes. Para acceder a las clases de un refactoring genérico, es necesario utilizar el nombre del paquete seguido de ".". El nombre del paquete lleva el mismo nombre que la clase principal del refactoring genérico. Esto significa que, por ejemplo en el caso del SplitPage, todas las clases exportadas en la variable *exportedObjects* se acceden mediante mediante el "paquete" SplitPage. Por ejemplo, la clase SplitPage, se accede con SplitPage.SplitPage, mientras que la clase SplitedSection se accede con SplitPage.SplitedSection. El uso de dichas clases depende de cómo haya sido definida el API del refactoring genérico.

Dentro del archivo donde se especifica el refactoring instanciado, es mandatorio definir la función (no un método de la clase que se especifica en el script, sino una función independiente) *#getAccessibilityAugmenter*. El único objetivo de esta función es retornar una instancia de la nueva clase que hereda de *AbstractInstanceRefactoring*.

```
function getAccessibilityAugmenter(){
        return new GMailListSplitWrapper();
};
```

De la misma manera que con los refactorings genéricos, para los refactorings instanciados también se debe definir la variable *metadata*, incluso con la misma estructura:

```
var metadata = {"author":"Sergio Firmenich",
                "name":"SplitPage for Gmail",
                "description":"Split Gmail Page into three main sections:
                               mails, folders and options. Also it adds some shortcuts to
                               relevant functionalities",
                "id":"splitPage-mainGmail-sfirmenich"};
```

Instalación de CSWR Framework

La instalación del framework puede llevarse a cabo tan solo con abrir el archivo .xpi con el navegador Web, lo cual ofrecerá al usuario la instalación. Luego de instalarlo y de reiniciar el navegador (reiniciar significa simplemente cerrarlo y abrirlo) ya esta listo para ser utilizado. Si el framework fue instalado correctamente, entonces encontrará una nueva opción llamada “Accesibilidad” en el menú principal del navegador.

Instalación de refactorings

La instalación de refactorings, tanto genéricos como instanciados, puede llevarse a cabo tan solo con abrir el archivo JavaScript correspondiente con el navegador Web (claro, una vez que el la extensión con la que se distribuye el framework ya este instalada). Al hacerlo, el framework ofrecerá al usuario instalar el refactoring cargado. Podrá ver en el administrador de refactorings (Accesibilidad->Administrar refactorings) aquellos que ya estén instalados.

Para aplicar refactorings, el framework tiene que estar activado. Para activar el framework, debe utilizarse la opción “Instalar todos” del mismo menú. Esto hará que el framework, al cargar un sitio Web, inicie el proceso de búsqueda de refactorings a aplicar y la aplicación de los mismos.

El orden con que se aplican los refactorings instanciados es el mismo orden con el que se lista en el administrador de refactorings. Este orden puede ser alterado utilizando dicho administrador. También desde el administrador los refactorings instanciados pueden activarse/desactivarse. Si un refactoring instanciado es desactivado, el framework no lo aplicará cuando se cargue el sitio Web correspondiente.