

Rutas básicas de un CRUD HTTP y su implementación mediante MVC con la tecnología de JSP & Servlets para tienda-informática

Desde una perspectiva pura de aplicación http + html (sin js) la aplicación sólo tiene disponible 2 métodos o verbos del protocolo http:

- **GET** - para obtener un recurso del servidor.

Este método es seguro en el sentido de que no realiza ningún cambio en los recursos alojados en el servidor, sólo realiza la lectura de los mismos.

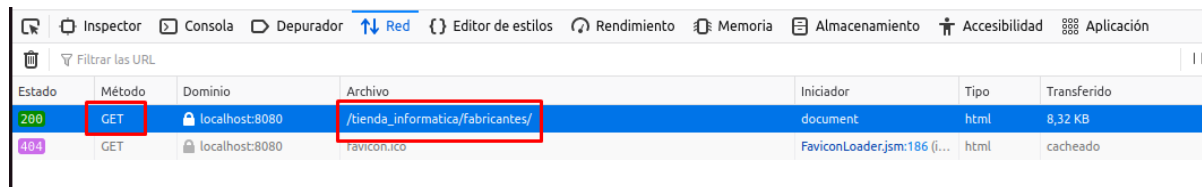
Características principales de las peticiones GET:

- se almacenan en caché.
- permanecen en el historial del navegador.
- se pueden marcar.
- nunca deben usarse cuando se trata de enviar datos confidenciales.
- tienen restricciones de longitud.
- solo se utilizan para solicitar datos (no modificar).

Cuando solicitas una URL desde la barra de navegación o un enlace de etiqueta , el navegador emitirá una petición GET al recurso de archivo indicado en la ruta. Por ejemplo en tienda_informatica tenemos que la solicitud del listado de fabricantes a través de la barra de navegación es mapeado a una petición GET.



Inspeccionando la petición HTTP emitida por el navegador se tiene el desglose método, dominio, archivo como el siguiente:

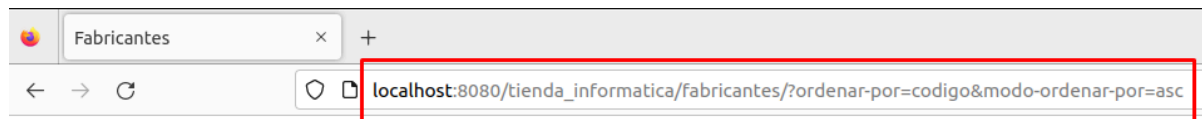


Por otro lado, los datos de una petición GET siempre van en la url de la petición a partir del signo de interrogación ? (bloque conocido como filtrado) como pares de nombre de parámetro = valor de parámetro.

Fijémonos en la siguiente petición donde al recurso que se solicita se le pasan 2 parámetros:

`http://localhost:8080/tienda_informatica/fabricantes/?ordenar-por=codigo&modo-ordenar-por=asc`

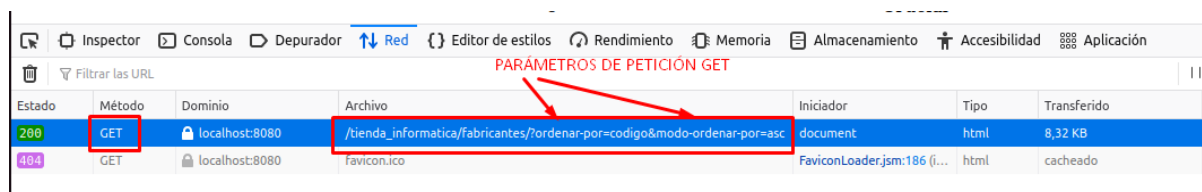
- **ordenar-por** con valor igual a **codigo**
- **modo-ordenar-por** con valor igual a **asc**



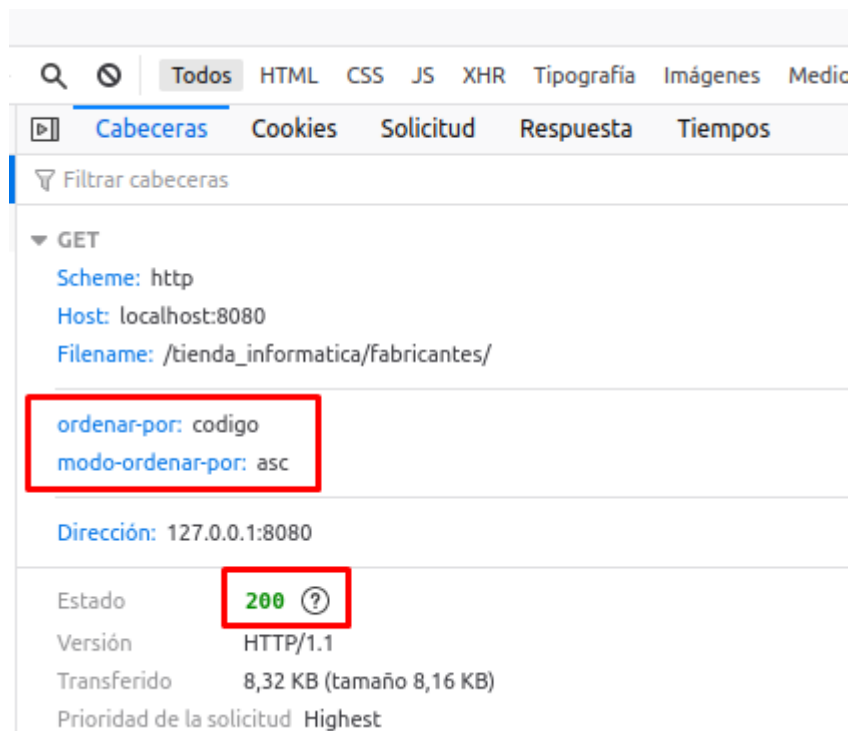
Fabricantes

Código	Nombre
1	Asus
2	Lenovo
3	Hewlett-Packard
4	Samsung
5	Seagate
6	Crucial

Inspeccionando la petición a detalle:



Se observa cómo se recogen por separado los parámetros de la petición del recurso solicitado:



- **POST** - para crear/editar/borrar un nuevo recurso en el servidor.

Este método no es seguro en el sentido de que realiza cambios en los recursos alojados en el servidor (crea/edita/borra).

Características principales de las peticiones POST:

- nunca se almacenan en caché.
- no permanecen en el historial del navegador.
- no se pueden marcar.
- no tienen restricciones en la longitud de los datos.

Los datos de una petición POST, a diferencia del caso de GET, viajan en el cuerpo de la petición. Estas peticiones en html (sin js) sólo pueden emitirse desde un formulario (etiqueta form) con action a la ruta donde se envía y el método utilizado. Fijémonos en la imagen siguiente:

The screenshot shows a web browser address bar with the URL `localhost:8080/tienda_informatica/fabricantes/crear?`. Below the address bar, a form is displayed with the title **Crear Fabricante**. The form contains a submit button labeled 'Crear' and an input field labeled 'Nombre' with the placeholder text 'Nuevo Fabricante'. The form's action is `/tienda_informatica/fabricantes/crear/` and its method is `post`.

****Visualizado de form sobre página mediante la extensión Web Developer**

<https://chrome.google.com/webstore/detail/web-developer/bfbameneiokkbodmiekhjnmfkcnldhbm?hl=es>

```
<div id="contenedora" style="float:none; margin: 0 auto;width: 900px;">
  <form action="/tienda_informatica/fabricantes/crear/" method="post">
    <div class="clearfix">
      <div style="float: left; width: 50%">...</div>
      <div style="float: none;width: auto;overflow: hidden;min-height: 80px;position: relative;">...</div>
    </div>
    <div class="clearfix">...</div>
    <div class="clearfix" style="margin-top: 6px;">...</div>
  </form>
```

Si pulsamos en el botón de enviar formulario input type submit veremos cómo viaja la petición POST:

The screenshot shows the Chrome DevTools network panel. A table of requests is visible:

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido
302	POST	localhost:8080	/tienda_informatica/fabricantes/crear/	document	html	9,13 KB
200	GET	localhost:8080	fabricantes	document	html	9,13 KB
404	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:186 (i...	html	cacheado

A red box highlights the POST request, and a red arrow points to the second GET request with the label "REDIRECCIÓN POR CABECERA LOCATION".

Below the network panel, the "Cabeceras" (Headers) tab is selected. It shows the details for the POST request:

- Estado: 302
- Versión: HTTP/1.1
- Transferido: 9,13 KB (tamaño 8,96 KB)
- Política de referencia: strict-origin-when-cross-origin
- Prioridad de la solicitud: Highest

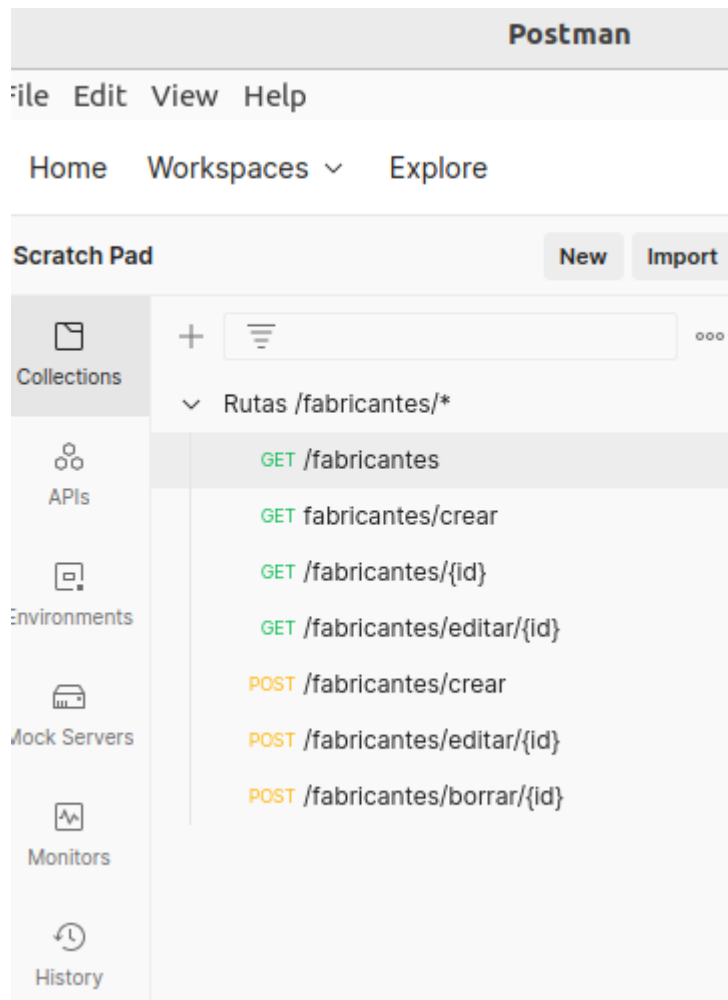
Under "Cabeceras de la respuesta (164 B)", the "Location" header is highlighted with a red box:

- Location: /tienda_informatica/fabricantes

**El segundo GET que emite el navegador se realiza en base a la respuesta del POST por la cabecera Location: /tienda_informatica/fabricantes que una vez procesada por el navegador desencadena un GET /tienda_informatica/fabricantes.

Rutas en POSTMAN

Las rutas del CRUD recogidas en la utilidad de Postman (<https://www.postman.com/downloads/>) son:

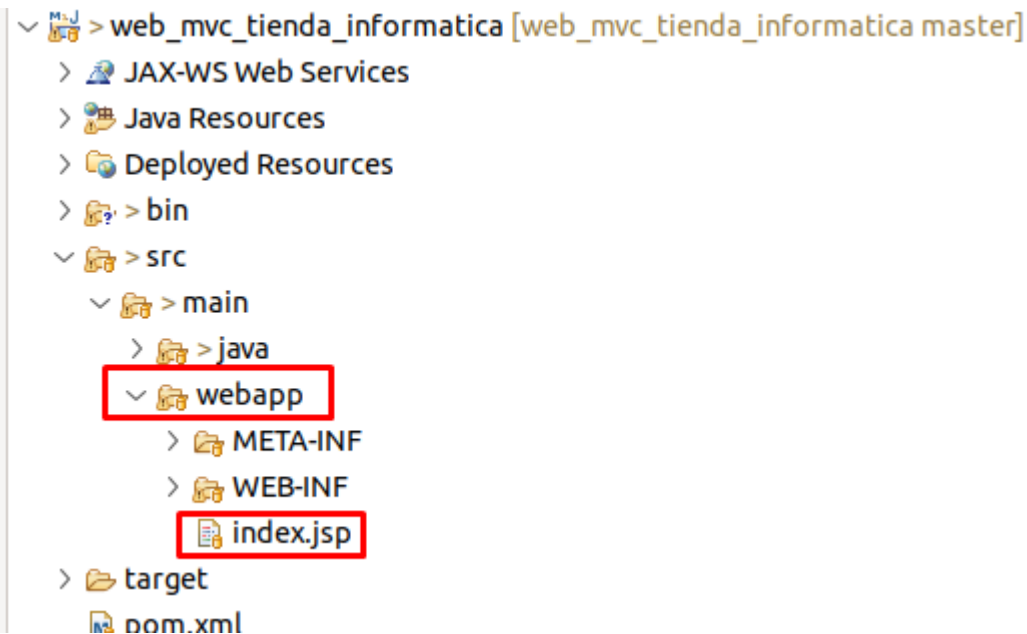


Se puede descargar el fichero de exportación de estas rutas desde:

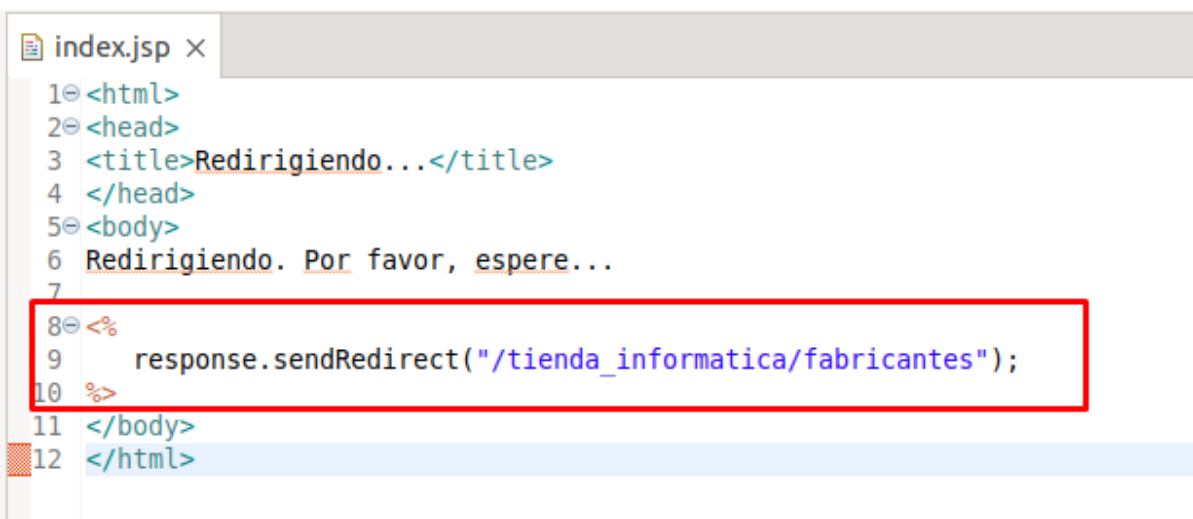
https://github.com/toteabe/web_mvc_tienda_informatica/blob/master/Rutas%20-fabricantes--.postman_collection.json

GET /fabricantes - fabricantes.jsp

En la aplicación existe un index.jsp a nivel raíz del directorio **webapp** de despliegue para Tomcat:



La solicitud en la barra del navegador de la URL http://localhost:8080/tienda_informatica/ cargará este **index.jsp** que contendrá un **scriptlet** java de redirección a la ruta GET /fabricantes, mediante el método **sendRedirect("/tienda_informatica/fabricantes")** del objeto response que el framework de JSP & Servlet hace disponible en cada jsp.



Viéndolo en el navegador con el inspector de peticiones de Red se obtiene la traza de peticiones:

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido
302	GET	localhost:8080	/tienda_informatica/	document	html	8,37 KB
200	GET	localhost:8080	fabricantes	document	html	8,33 KB
404	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:186 (i...	html	cacheado

EFEECTO DE `response.sendRedirect("/tienda_informatica/fabricantes")`

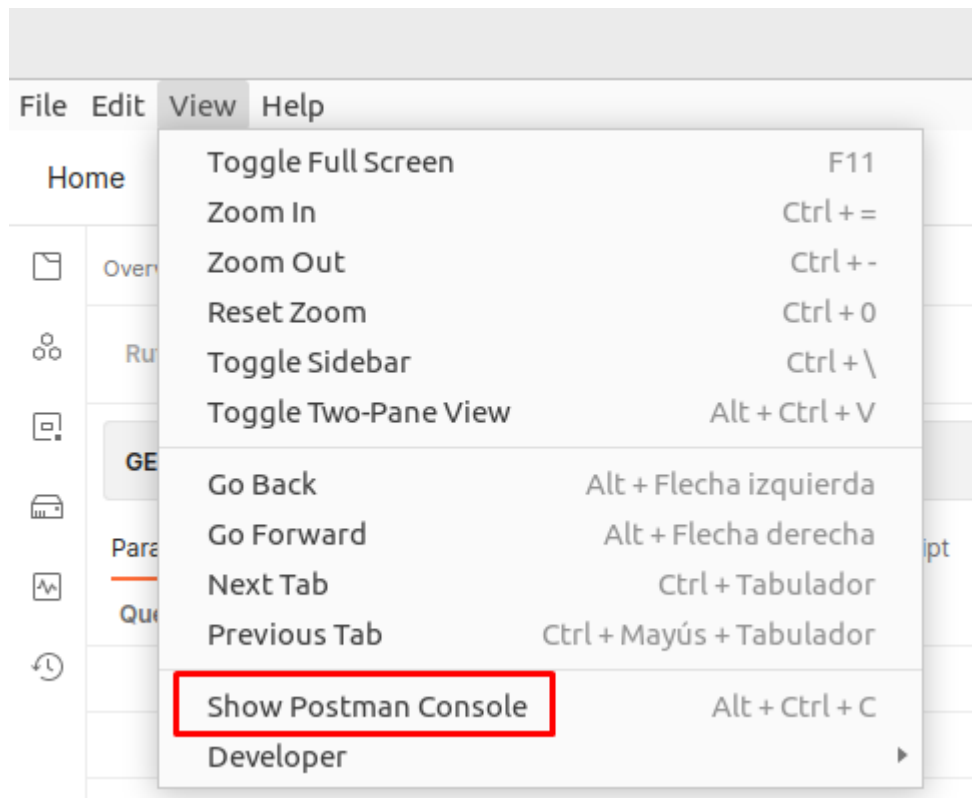
Cabeceras	Cookies	Solicitud	Respuesta	Tiempos
Filtrar cabeceras				
GET http://localhost:8080/tienda_informatica/				
Estado	302 ?			
Versión	HTTP/1.1			
Transferido	8,37 KB (tamaño 8,17 KB)			
Prioridad de la solicitud	Highest			
Cabeceras de la respuesta (208 B)				
Connection: keep-alive				
Content-Length: 0				
Content-Type: text/html; charset=ISO-8859-1				
Date: Sat, 12 Nov 2022 05:38:35 GMT				
Keep-Alive: timeout=20				
Location: /tienda_informatica/fabricantes				

REDIRECCIÓN
MEDIANTE LOCATION

Podemos testear la ruta con Postman de forma independiente la redirección mediante

GET	▼	http://localhost:8080/tienda_informatica/
-----	---	---

Y visualizarlo en la consola:



Consola:

```
► GET http://localhost:8080/tienda_informatica/
► GET http://localhost:8080/tienda_informatica/fabricantes
...
302 ↗
200 | 54 ms
```

El primer *GET* con respuesta con código de estado 302 de recurso movido a través de la cabecera de Location

El segundo *GET* ya obtiene el recurso, código de estado 200.

Se puede probar directamente la ruta *GET /fabricantes* mediante Postman:

Overview GET /fabricantes + ...

Rutas /fabricantes/* / /fabricantes

GET http://localhost:8080/tienda_informatica/fabricantes

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Con la respuesta:

Body Cookies (1) Headers (5) Test Results Status: 200 OK Time: 65 ms Size: 8.14 KB

Pretty Raw Preview Visualize

Fabricantes

Crear

Código	Nombre	Acción
1	Otro fabricante	Ver Detalle Editar Eliminar
2	Lenovo	Ver Detalle Editar Eliminar
3	Hewlett-Packard	Ver Detalle Editar Eliminar
4	Samsung	Ver Detalle Editar Eliminar

A nivel del código de Servlet el tratamiento de esta ruta se encuentra mapeado a **FabricantesServlet** a través del fichero web.xml:

index.jsp FabricantesServlet.java web.xml x

```

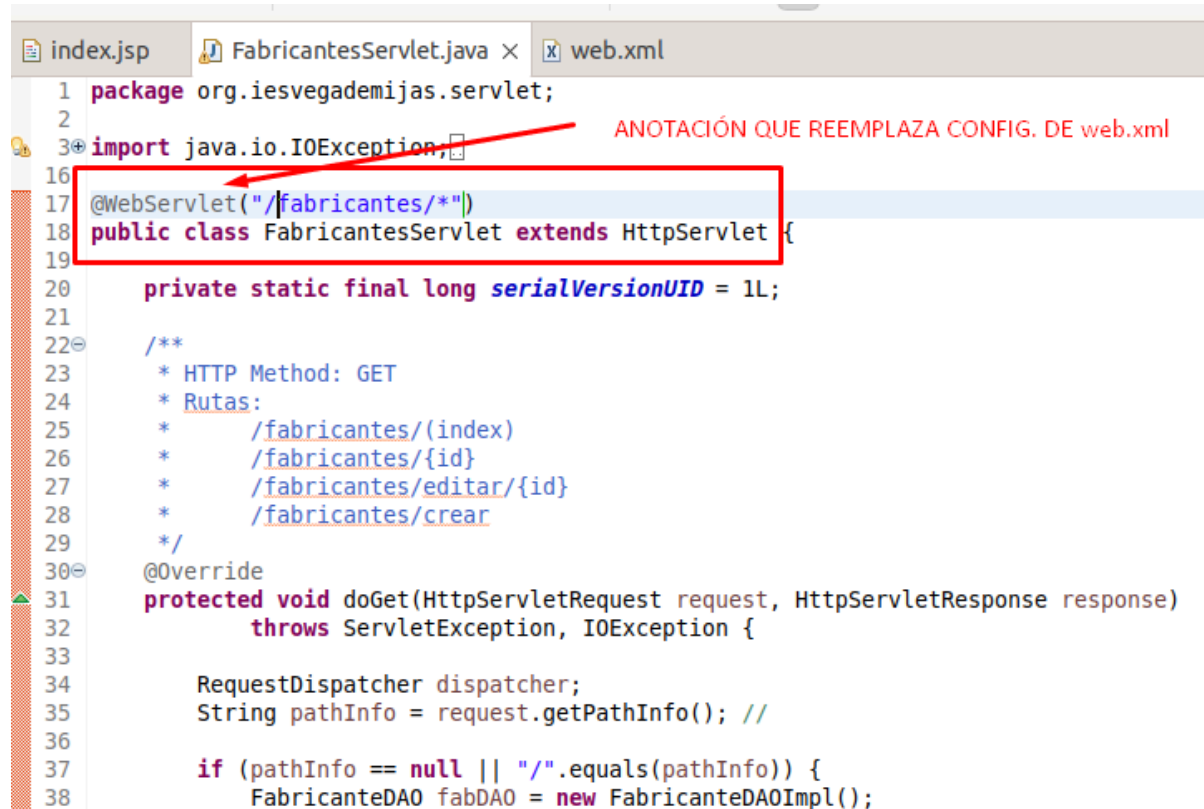
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3   <display-name>web_mvc_tienda_informatica</display-name>
4   <welcome-file-list>
5     <welcome-file>index.jsp</welcome-file>
6   </welcome-file-list>
7
8   <servlet>
9     <servlet-name>FabricanteServlet</servlet-name>
10    <servlet-class>org.iesvegamemijas.servlet.FabricantesServlet</servlet-class>
11  </servlet>
12  <servlet-mapping>
13    <servlet-name>FabricanteServlet</servlet-name>
14    <url-pattern>/fabricantes/*</url-pattern>
15  </servlet-mapping>
16
17 </web-app>

```

CLASE DEL SERVLET

RUTAS MAPEADAS A FabricanteServlet

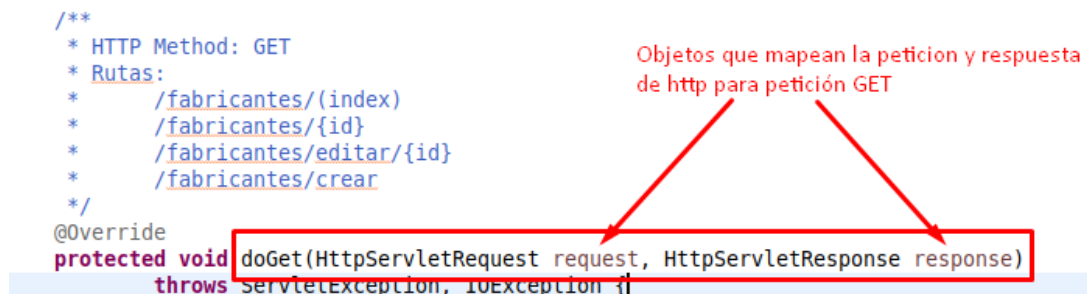
Si se está utilizando la especificación de **Servlet > 3.0** se puede utilizar la anotación **@WebServlet** prescindiendo de la configuración del fichero de web.xml.



```
1 package org.iesvega demijas.servlet;
2
3 import java.io.IOException;
4
5 @WebServlet("/fabricantes/*")
6 public class FabricantesServlet extends HttpServlet {
7
8     private static final long serialVersionUID = 1L;
9
10    /**
11     * HTTP Method: GET
12     * Rutas:
13     * /fabricantes/(index)
14     * /fabricantes/{id}
15     * /fabricantes/editar/{id}
16     * /fabricantes/crear
17     */
18    @Override
19    protected void doGet(HttpServletRequest request, HttpServletResponse response)
20        throws ServletException, IOException {
21
22        RequestDispatcher dispatcher;
23        String pathInfo = request.getPathInfo(); //
24
25        if (pathInfo == null || "/".equals(pathInfo)) {
26            FabricanteDAO fabDAO = new FabricanteDAOImpl();
27
28        }
29    }
30 }
```

Nótese que los respectivos verbos de HTTP se mapean a un método de **FabricanteServlet**:

- HTTP GET → doGet



```
/**
 * HTTP Method: GET
 * Rutas:
 * /fabricantes/(index)
 * /fabricantes/{id}
 * /fabricantes/editar/{id}
 * /fabricantes/crear
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
}
```

- HTTP POST → doPost

```

/**
 * HTTP Method: POST
 * Rutas:
 *     /fabricantes/crear
 *     /fabricantes/editar/{id}
 *     /fabricantes/borrar/{id}
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

```

Objetos donde se mapean la petición y respuesta de POST

FabricantesServlet tendrá una primera fase de tratamiento de la ruta para discriminar la acción a realizar. Si se trata de una petición

- GET /fabricantes → se trata en el doGet → se discrimina la ruta en una primera fase mediante:

```
String pathInfo = request.getPathInfo();
```

```

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    RequestDispatcher dispatcher;
    String pathInfo = request.getPathInfo();

    if (pathInfo == null || "/".equals(pathInfo)) {
        FabricanteDAO fabDAO = new FabricanteDAOImpl();

        //GET
        // /fabricantes/
        // /fabricantes

        request.setAttribute("listaFabricantes", fabDAO.getAll());
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
    }
}

```

Acceso a la ruta de la petición HTTP, para discriminar la acción a realizar.

Si **pathInfo** es null o igual a / significa que estás recibiendo ruta pura:

- /fabricantes
- /fabricantes/

```
if (pathInfo == null || "/".equals(pathInfo)) {  
    FabricanteDAO fabDAO = new FabricanteDAOImpl();
```

```
//GET  
// /fabricantes/  
// /fabricantes
```

Rutas mapeadas a este procesamiento.

```
request.setAttribute("listaFabricantes", fabDAO.getAll());  
dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
```

```
} else {
```

Esta ruta es el index del mantenimiento de fabricantes **GET** **/fabricantes/(index)** y debe devolver el listado y panel de operaciones CRUD completo. Para ello, se instancia el DAO de acceso a base de datos para la entidad Fabricante y se ejecuta un `getAll` que devuelve un listado con todos los fabricantes.

```
if (pathInfo == null || "/".equals(pathInfo)) {  
    FabricanteDAO fabDAO = new FabricanteDAOImpl();
```

```
//GET  
// /fabricantes/  
// /fabricantes
```

Devuelve List<Fabricante>

```
request.setAttribute("listaFabricantes", fabDAO.getAll());  
dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
```

```
} else {
```

Para acceder a la vista debes setear el **dispatcher** (distribuidor) hacia la vista interna (oculta, no accesible mediante ruta directa por estar alojada en WEB-INF) de **fabricantes.jsp** mediante:

- dispatcher =
 request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");

Los beans de Fabricante se pasan a la vista mediante un atributo en el request mediante:

- request.setAttribute("listaFabricantes", fabDAO.getAll());

```

if (pathInfo == null || "/".equals(pathInfo)) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();

    //GET
    // /fabricantes/
    // /fabricantes

    request.setAttribute("listaFabricantes", fabDAO.getAll());
    dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
} else {

```

Atributo para la vista donde se pasa la lista de fabricantes List<Fabricante>

Página vista donde se renderiza el listado y las operaciones CRUD

Al final de **doGet** está el ejecutador del **dispatcher**:

- `dispatcher.forward(request, response);`

```

19     }
20 }
21 }
22
23 dispatcher.forward(request, response);
24
25 }

```

Ejecutador de la redirección hacia vista interna /WEB-INF/jsp/fabricantes.jsp dado que el dispatcher se cargó con esta ruta en el procesamiento.

La página que se enviará bajo la ruta **GET /fabricantes** será **fabricantes.jsp** que tendrá el código html, css y de scriptlets Java para renderizar un listado de fabricantes con las operaciones CRUD.

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@page import="org.iesvegaademijas.model.Fabricante"%>
4 <%@page import="java.util.List"%>
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9   <meta charset="UTF-8">
10  <title>Fabricantes</title>
11  <style>
12    .clearfix::after {
13      content: "";
14      display: block;
15      clear: both;
16    }
17
18  </style>
19  </head>
20  <body>
21  <body>
22

```

Directiva que expresa que se utiliza código scriptlet de java

Directivas de import de los objetos java utilizados en scriptlets de dinamismo para el código html

El código de **scriptlet** se introduce en bloques `<% %>` como el siguiente que se encarga de recorrer la colección de fabricantes pasada por atributo request de listaFabricantes

The image shows a code editor with JSP code. Red boxes and arrows highlight specific parts of the code with explanatory text:

- Capa cabecera de la lista de fabricantes:** Points to the first `<div>` block containing headers for 'Código', 'Nombre', and 'Acción'.
- Scriptlet java que si existe listaFabricantes itera cada fabricante para representar la línea de lista de fabricante correspondiente.** Points to the `if` block that checks for the existence of 'listaFabricantes' and iterates over it.
- Línea de lista de fabricante específico iterado.** Points to the nested `<div>` blocks that generate the HTML for each manufacturer, including links for 'Ver Detalle', 'Editar', and 'Eliminar'.
- Scriptlet para el otro caso de no haber listaFabricantes con mensaje de No hay registros.** Points to the `else` block that outputs 'No hay registros de fabricante'.

Destacar que la ruta **GET /fabricantes/(index)** contiene las operaciones CRUD además:

- **crear** con ruta **GET /fabricantes/crear**
- **ver detalle** con ruta **GET /fabricantes/{id}**
- **editar** con ruta **GET /fabricantes/editar/{id}**
- **borrar** con ruta **POST/fabricantes/borrar/{id}**

```

23<div id="contenedora" style="float:none; margin: 0 auto;width: 900px;" >
24<div class="clearfix">
25<div style="float: left; width: 50%">
26<h1>Fabricantes</h1>
27</div>
28<div style="float: none;width: auto;overflow: hidden;min-height: 80px;position: relative;">
29
30<div style="position: absolute; left: 39%; top : 39%;">
31<form action="/tienda_informatica/fabricantes/crear">
32<input type="submit" value="Crear">
33</form>
34</div>
35
36</div>
37</div>
38<div class="clearfix">
39<hr/>
40</div>
41<div class="clearfix">
42<div style="float: left;width: 33%">Código</div>
43<div style="float: left;width: 33%">Nombre</div>
44<div style="float: none;width: auto;overflow: hidden;">Acción</div>
45</div>
46<div class="clearfix">
47<hr/>
48</div>
49</div>
50
51if (request.getAttribute("listaFabricantes") != null) {
52List<Fabricante> listaFabricante = (List<Fabricante>)request.getAttribute("listaFabricantes");
53
54for (Fabricante fabricante : listaFabricante) {
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Esta ruta **GET /fabricantes/(index)** actúa de index del **mantenimiento de Fabricantes** e implementa el **READ** del **CRUD** a partir de aquí se lanzan las restantes operaciones **CRUD**: **CREATE** (crear), **UPDATE** (editar) y **DELETE** (borrar).

GET /fabricantes/crear/ - crear-fabricante.jsp

Ruta que carga el formulario para crear un nuevo fabricante.

Se discrimina su tratamiento en FabricantesServlet mediante pathInfo como sigue:


```

if (pathInfo == null || "/" .equals(pathInfo)) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();

    //GET
    // /fabricantes/
    // /fabricantes

    request.setAttribute("listaFabricantes", fabDAO.getAll());
    dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");

} else {
    // GET
    // /fabricantes/{id}
    // /fabricantes/{id}/
    // /fabricantes/edit/{id}
    // /fabricantes/edit/{id}/
    // /fabricantes/create
    // /fabricantes/create/

    pathInfo = pathInfo.replaceAll("/$", "");
    String[] pathParts = pathInfo.split("/");

    if (pathParts.length == 2 && "crear".equals(pathParts[1])) {

        // GET
        // /fabricantes/create
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/crear-fabricante.jsp");

    } else if (pathParts.length == 2) {}
    FabricanteDAO fabDAO = new FabricanteDAOImpl();

```

Obtención de partes de la ruta mediante split

Discriminación para ruta crear

Vista a presentar

La vista **crear-fabricante.jsp** contiene el **formulario** para crear un nuevo fabricante mediante la **ruta POST /fabricantes/crear**

```

20
21<div id="contenedora" style="float:none; margin: 0 auto; width: 900px;" >
22<form action="/tienda_informatica/fabricantes/crear/" method="post">
23<div class="clearfix">
24<div style="float: left; width: 50%">
25<h1>Crear Fabricante</h1>
26</div>
27<div style="float: none; width: auto; overflow: hidden; min-height: 80px; position: relative;">
28
29<div style="position: absolute; left: 39%; top: 39%;">
30<input type="submit" value="Crear"/>
31</div>
32
33</div>
34</div>
35
36<div class="clearfix">
37<hr/>
38</div>
39
40<div style="margin-top: 6px;" class="clearfix">
41<div style="float: left; width: 50%">
42Nombre
43</div>
44<div style="float: none; width: auto; overflow: hidden;">
45<input name="nombre" />
46</div>
47</div>
48
49</form>
50</div>

```

Ruta POST /fabricantes/crear en el cuerpo de este POST irán los campos en este caso el input nombre

POST /fabricantes/crear

Ruta encargada de grabar un nuevo fabricante. Puesto que es un POST es atendida en el Servlet en el método doPost:

```

/**
 * HTTP Method: POST
 * Rutas:
 * /fabricantes/crear
 * /fabricantes/editar/{id}
 * /fabricantes/borrar/{id}
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    RequestDispatcher dispatcher;
    String __method__ = request.getParameter("__method__");

    //Se comenta el tratamiento de la ruta dado que con la técnica de parámetro oculto __method__ = null | put | delete, sobre la ruta de POST /fabricantes/*
    //ya se puede decidir
    //String pathInfo = request.getPathInfo();
    //pathInfo = pathInfo.replaceAll("/$", "");
    //String[] pathParts = pathInfo.split("/");

    if (__method__ == null)
        //Se comenta el tratamiento de la ruta dado que con la técnica de parámetro oculto __method__ = null | put | delete, sobre la ruta de POST /fabricantes/*
        //ya se puede decidir
        //String pathInfo = request.getPathInfo();
        //pathInfo = pathInfo.replaceAll("/$", "");
        //String[] pathParts = pathInfo.split("/");
        //Se crea uno nuevo.
        //Ruta POST /fabricantes/crear
        // parámetro oculto __method__ no se envía, equivalente a __method__ = null
        FabricanteDAO fabDAO = new FabricanteDAOImpl();

        String nombre = request.getParameter("nombre");
        Fabricante nuevoFab = new Fabricante();
        nuevoFab.setNombre(nombre);
        fabDAO.create(nuevoFab);

    } else if (__method__ != null && "put".equalsIgnoreCase(__method__))
        //Se crea uno nuevo.
        //Ruta POST /fabricantes/crear
        // parámetro oculto __method__ no se envía, equivalente a __method__ = null
        FabricanteDAO fabDAO = new FabricanteDAOImpl();

        String nombre = request.getParameter("nombre");
        Fabricante nuevoFab = new Fabricante();
        nuevoFab.setNombre(nombre);
        fabDAO.create(nuevoFab);

    } else if (__method__ != null && "delete".equalsIgnoreCase(__method__))
        //Se crea uno nuevo.
        //Ruta POST /fabricantes/crear
        // parámetro oculto __method__ no se envía, equivalente a __method__ = null
        FabricanteDAO fabDAO = new FabricanteDAOImpl();

        String nombre = request.getParameter("nombre");
        Fabricante nuevoFab = new Fabricante();
        nuevoFab.setNombre(nombre);
        fabDAO.create(nuevoFab);

    } else {
        System.out.println("Opción POST no soportada.");
        response.sendRedirect("/tienda_informatica/fabricantes");
    }
}

```

Técnica de __method__ oculto para discriminar las operaciones:
 POST __method__ == null --> crear
 POST __method__ == PUT --> editar
 POST __method__ == DELETE --> borrar

Crear

Todas las rutas POST en la aplicación realizan una redirección al listado principal ruta **GET /fabricantes/(index).**

```

index.jsp  FabricantesServlet.java  web.xml  fabricantes.jsp  crear-fabricante.jsp
121
122 if (__method__ == null
123     //Se comenta el tratamiento de la ruta dado que con la técnica de parámetro oculto __method__ = null | put | delete, sobre la ruta de POST /fabricantes/*
124     //ya se puede decidir
125     //String pathInfo = request.getPathInfo();
126     //pathInfo = pathInfo.replaceAll("/$", "");
127     //String[] pathParts = pathInfo.split("/");
128     //Se crea uno nuevo.
129     //Ruta POST /fabricantes/crear
130     // parámetro oculto __method__ no se envía, equivalente a __method__ = null
131     FabricanteDAO fabDAO = new FabricanteDAOImpl();
132
133     String nombre = request.getParameter("nombre");
134     Fabricante nuevoFab = new Fabricante();
135     nuevoFab.setNombre(nombre);
136     fabDAO.create(nuevoFab);
137
138 } else if (__method__ != null && "put".equalsIgnoreCase(__method__))
139     //Se crea uno nuevo.
140     //Ruta POST /fabricantes/crear
141     // parámetro oculto __method__ no se envía, equivalente a __method__ = null
142     FabricanteDAO fabDAO = new FabricanteDAOImpl();
143
144     String nombre = request.getParameter("nombre");
145     Fabricante nuevoFab = new Fabricante();
146     nuevoFab.setNombre(nombre);
147     fabDAO.create(nuevoFab);
148
149 } else if (__method__ != null && "delete".equalsIgnoreCase(__method__))
150     //Se crea uno nuevo.
151     //Ruta POST /fabricantes/crear
152     // parámetro oculto __method__ no se envía, equivalente a __method__ = null
153     FabricanteDAO fabDAO = new FabricanteDAOImpl();
154
155     String nombre = request.getParameter("nombre");
156     Fabricante nuevoFab = new Fabricante();
157     nuevoFab.setNombre(nombre);
158     fabDAO.create(nuevoFab);
159
160 } else {
161     System.out.println("Opción POST no soportada.");
162     response.sendRedirect("/tienda_informatica/fabricantes");
163 }
164
165 }
166
167
168
169

```

Crear

Redirección

GET /fabricantes/{id} - detalle-fabricante.jsp

Ruta para ver detalle de un fabricante. En el tratamiento de FabricantesServlet sobre el pathInfo se accede a ella por la discriminación de condición:

- `pathParts.length == 2`

```
pathInfo = pathInfo.replaceAll("/$", "");
String[] pathParts = pathInfo.split("/");

if (pathParts.length == 2 && "crear".equals(pathParts[1])) {

    // GET
    // /fabricantes/create
    dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/crear-fabricante.jsp");
```

```
} else if (pathParts.length == 2) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();
    // GET
    // /fabricantes/{id}
    try {
        request.setAttribute("fabricante", fabDAO.find(Integer.parseInt(pathParts[1])));
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/detalle-fabricante.jsp");
    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
    }
}
```



```
} else if (pathParts.length == 3 && "editar".equals(pathParts[1])) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();

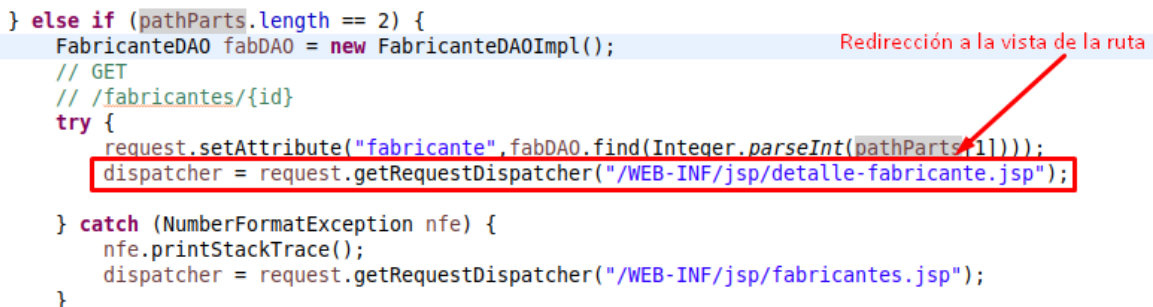
    // GET
    // /fabricantes/editar/{id}
    try {
        request.setAttribute("fabricante", fabDAO.find(Integer.parseInt(pathParts[2])));
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/editar-fabricante.jsp");
    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
    }
}

} else {
```

Esta ruta tiene asociada la vista /WEB-INF/jsp/fabricantes.jsp, a la que redirige el dispatcher seteado mediante:

- `dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/detalle-fabricante.jsp");`

```
} else if (pathParts.length == 2) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();
    // GET
    // /fabricantes/{id}
    try {
        request.setAttribute("fabricante", fabDAO.find(Integer.parseInt(pathParts[1])));
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/detalle-fabricante.jsp");
    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
    }
}
```



Esta vista espera el bean fabricante como atributo request para mostrar el detalle.

```
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

<div class="clearfix">
  <hr/>
</div>

<% Optional<Fabricante> optFab = (Optional<Fabricante>)request.getAttribute("fabricante");
  if (optFab.isPresent()) {

    <div style="margin-top: 6px;" class="clearfix">
      <div style="float: left; width: 50%">
        <label>Código</label>
      </div>
      <div style="float: none; width: auto; overflow: hidden;">
        <input value="<%= optFab.get().getCodigo() %>" readonly="readonly"/>
      </div>
    </div>
    <div style="margin-top: 6px;" class="clearfix">
      <div style="float: left; width: 50%">
        <label>Nombre</label>
      </div>
      <div style="float: none; width: auto; overflow: hidden;">
        <input value="<%= optFab.get().getNombre() %>" readonly="readonly"/>
      </div>
    </div>

    } else {

      request.sendRedirect("fabricantes/");

    }
  } %>
```

Scriptlet java que chequea existencia de fabricante para presentar

Si no hay fabricante, redirección a ruta GET /fabricantes/(index)

Nota HTML: fíjate que al ser un detalle los campos van en readonly.

GET /fabricantes/editar/{id} - editar-fabricante.jsp

Ruta para acceder al formulario de edición de un fabricante. En FabricantesServlet se discrimina por:

- `pathParts.length == 3 && "editar".equals(pathParts[1])`

```

pathInfo = pathInfo.replaceAll("/", "");
String[] pathParts = pathInfo.split("/");

if (pathParts.length == 2 && "crear".equals(pathParts[1])) {

    // GET
    // //fabricantes/create
    dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/crear-fabricante.jsp");

} else if (pathParts.length == 2) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();
    // GET
    // //fabricantes/{id}
    try {
        request.setAttribute("fabricante", fabDAO.find(Integer.parseInt(pathParts[1])));
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/detalle-fabricante.jsp");

    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
    }

} else if (pathParts.length == 3 && "editar".equals(pathParts[1]) ) {
    FabricanteDAO fabDAO = new FabricanteDAOImpl();

    // GET
    // //fabricantes/editar/{id}
    try {
        request.setAttribute("fabricante", fabDAO.find(Integer.parseInt(pathParts[2])));
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/editar-fabricante.jsp");

    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
        dispatcher = request.getRequestDispatcher("/WEB-INF/jsp/fabricantes.jsp");
    }

} else {

```

El dispatcher carga para esta ruta la vista `jsp editar-fabricante.jsp` que espera el bean `fabricante` como parámetro del request para establecer los datos en el formulario de edición.

```

<form action="/tienda informatica/fabricantes/editar/" method="post" >
  <input type="hidden" name="__method__" value="put" />
  <div class="clearfix">
    <div style="float: left; width: 50%">
      <h1>Editar Fabricante</h1>
    </div>
    <div style="float: none; width: auto; overflow: hidden; min-height: 80px; position: relative;">
      <div style="position: absolute; left: 39%; top: 39%;">
        <input type="submit" value="Guardar" />
      </div>
    </div>
  </div>
  <div class="clearfix">
    <hr/>
  </div>
  <%= Optional<Fabricante> optFab = (Optional<Fabricante>)request.getAttribute("fabricante");
  if (optFab.isPresent()) {
    <div style="margin-top: 6px;" class="clearfix">
      <div style="float: left; width: 50%">
        <label>Código</label>
      </div>
      <div style="float: none; width: auto; overflow: hidden;">
        <input name="codigo" value="<%= optFab.get().getCodigo() %>" readonly="readonly"/>
      </div>
    </div>
    <div style="margin-top: 6px;" class="clearfix">
      <div style="float: left; width: 50%">
        <label>Nombre</label>
      </div>
      <div style="float: none; width: auto; overflow: hidden;">
        <input name="nombre" value="<%= optFab.get().getNombre() %>" />
      </div>
    </div>
  </div>
  <%= } else { %>
    request.sendRedirect("fabricantes/");
  }
  </div>

```

Form para enviar los cambios en el registro de fabricante

Técnica de parámetro oculto __method__ para indicar que se trata de un verbo PUT. Los forms de html 5 sólo soportan los verbos HTTP GET y POST. Esta técnica

Recogida del bean de fabricante

Preseteo de los campos del formulario

Parámetros a recibir en FabricantesServlet

POST /fabricantes/editar/{id}

Esta ruta envía los datos del formulario de la vista editar-fabricante.jsp cargado en la ruta GET /fabricantes/editar/{id} para actualizar el registro del fabricante. Se discrimina en el método doPost de FabricantesServlet.

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    RequestDispatcher dispatcher;
    String __method__ = request.getParameter("__method__");

    //Se comenta el tratamiento de la ruta dado que con la técnica de parámetro oculto __method__ = null
    //ya se puede decidir
    //String pathInfo = request.getPathInfo();
    //pathInfo = pathInfo.replaceAll("/$", "");
    //String[] pathParts = pathInfo.split("/");

    if (__method__ == null)
        //&& pathParts.length == 2 && "crear".equals(pathParts[1])
        ) {
        // crear: se crea uno nuevo.
        //Ruta POST /fabricantes/crear
        // parámetro oculto __method__ no se envía, equivalente a __method__ = null
        FabricanteDAO fabDAO = new FabricanteDAOImpl();

        String nombre = request.getParameter("nombre");
        Fabricante nuevoFab = new Fabricante();
        nuevoFab.setNombre(nombre);
        fabDAO.create(nuevoFab);

    } else if (__method__ != null && "put".equalsIgnoreCase(__method__))
        //&& pathParts.length == 3 && "editar".equals(pathParts[1])
        ) {
        // editar:
        //Dado que los forms de html sólo soportan method GET y POST utilizo parámetro oculto para indicar
        //Ruta POST /fabricantes/crear/{id}
        // parámetro oculto __method__ se envía con verbo PUT
        doPut(request, response);

    } else if (__method__ != null && "delete".equalsIgnoreCase(__method__))
        //pathParts.length == 3 && "borrar".equals(pathParts[1])
        ) {
        // borrar:
        //Dado que los forms de html sólo soportan method GET y POST utilizo parámetro oculto para indicar
        //Ruta POST /fabricantes/borrar/{id}
        // parámetro oculto __method__ se envía con verbo DELETE
        doDelete(request, response);
    }
}

```

Si inspeccionamos doPut:

```

@Override
protected void doPut(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    FabricanteDAO fabDAO = new FabricanteDAOImpl();
    String codigo = request.getParameter("codigo");
    String nombre = request.getParameter("nombre");
    Fabricante fab = new Fabricante();

    try {

        int id = Integer.parseInt(codigo);
        fab.setCodigo(id);
        fab.setNombre(nombre);
        fabDAO.update(fab);

    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
    }

}

```

Los métodos POST de cualquier ruta realizan una redirección a la ruta GET /fabricantes/ al final del método.


```

fabDAO.create(nuevoFab);

} else if ( __method__ != null && "put".equalsIgnoreCase( __method__ )
    //&& pathParts.length == 3 && "editar".equals(pathParts[1])
    ) {
    // editar:
    //Dado que los forms de html sólo soportan method GET y POST utilizo parámetro oculto __method__ ;
    //Ruta POST /fabricantes/editar/{id}
    // parámetro oculto __method__ se envía con verbo PUT
    doPut(request, response);

} else if ( __method__ != null && "delete".equalsIgnoreCase( __method__ )
    //pathParts.length == 3 && "borrar".equals(pathParts[1])
    ) {
    // borrar:
    //Dado que los forms de html sólo soportan method GET y POST utilizo parámetro oculto para indica
    //Ruta POST /fabricantes/borrar/{id}
    // parámetro oculto __method__ se envía con verbo DELETE
    doDelete(request, response);

} else {

    System.out.println("Opción POST no soportada.");

}

response.sendRedirect("/tienda_informatica/fabricantes");
}

```

Redirección por navegador a ruta GET /fabricantes

POST /fabricantes/borrar/{id}

En la vista fabricantes.jsp asociada a la ruta GET /fabricantes se tienen los formularios de borrar fabricante por cada línea de fabricante del listado.

```

47 </ul>
50
51 if (request.getAttribute("listaFabricantes") != null) {
52     List<Fabricante> listaFabricante = (List<Fabricante>)request.getAttribute("listaFabricantes");
53
54     for (Fabricante fabricante : listaFabricante) {
55
56
57         <div style="margin-top: 6px;" class="clearfix">
58             <div style="float: left; width: 33%;>=< fabricante.getCodigo()</div>
59             <div style="float: left; width: 33%;>=< fabricante.getNombre()</div>
60             <div style="float: none; width: auto; overflow: hidden;">
61                 <form action="/tienda_informatica/fabricantes/<%= fabricante.getCodigo()>" style="display: inline;">
62                     <input type="submit" value="Ver Detalle" />
63                 </form>
64                 <form action="/tienda_informatica/fabricantes/eliminar/<%= fabricante.getCodigo()>" style="display: inline;">
65                     <input type="submit" value="Eliminar" />
66                 </form>
67                 <form action="/tienda_informatica/fabricantes/borrar/<%= fabricante.getCodigo()>" method="post" style="display: inline;">
68                     <input type="hidden" name="__method__" value="delete"/>
69                     <input type="hidden" name="codigo" value="<%= fabricante.getCodigo()>" />
70                     <input type="submit" value="Eliminar" />
71                 </form>
72             </div>
73         </div>
74
75     }
76 } else {
77     No hay registros de fabricante
78 }
79
80 </div>
81

```

Formulario de con ruta de borrar POST /fabricantes/borrar/{id}

Técnica de parámetro oculto __method__ a DELETE para indicar operación de borrado sobre envío POST de navegador.

En el controlador FabricantesServlet se discrimina la operación de borrado de la ruta POST /fabricantes/borrar/{id} mediante el siguiente código en el doPost:


```

127
128 if ( __method__ == null
129     //&& pathParts.length == 2 && "crear".equals(pathParts[1])
130 ) {
131     // crear: se crea uno nuevo.
132     // Ruta POST /fabricantes/crear
133     // parámetro oculto __method__ no se envía, equivalente a __method__ = null
134     FabricanteDAO fabDAO = new FabricanteDAOImpl();
135
136     String nombre = request.getParameter("nombre");
137     Fabricante nuevoFab = new Fabricante();
138     nuevoFab.setNombre(nombre);
139     fabDAO.create(nuevoFab);
140
141 } else if ( __method__ != null && "put".equalsIgnoreCase( __method__ )
142     //&& pathParts.length == 3 && "editar".equals(pathParts[1])
143 ) {
144     // editar:
145     // Dado que los forms de html sólo soportan method GET y POST utilizo parámetro oculto
146     // Ruta POST /fabricantes/editar/{id}
147     // parámetro oculto __method__ se envía con verbo PUT
148     doPut(request, response);
149
150 } else if ( __method__ != null && "delete".equalsIgnoreCase( __method__ )
151     //&& pathParts.length == 3 && "borrar".equals(pathParts[1])
152 ) {
153     // borrar:
154     // Dado que los forms de html sólo soportan method GET y POST utilizo parámetro oculto
155     // Ruta POST /fabricantes/borrar/{id}
156     // parámetro oculto __method__ se envía con verbo DELETE
157     doDelete(request, response);
158
159 } else {
160     System.out.println("Opción POST no soportada.");
161 }
162
163 response.sendRedirect("/tienda_informatica/fabricantes");
164
165 }
166

```

Acción sobre doDelete

Técnica de parámetro oculto __method__ igual a delete para borrar

En el doDelete se tendrá el código correspondiente que invoca a la capa DAO para borrar:

```

@Override
protected void doDelete(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    RequestDispatcher dispatcher;
    FabricanteDAO fabDAO = new FabricanteDAOImpl();
    String codigo = request.getParameter("codigo");

    try {
        int id = Integer.parseInt(codigo);

        fabDAO.delete(id);

    } catch (NumberFormatException nfe) {
        nfe.printStackTrace();
    }

}

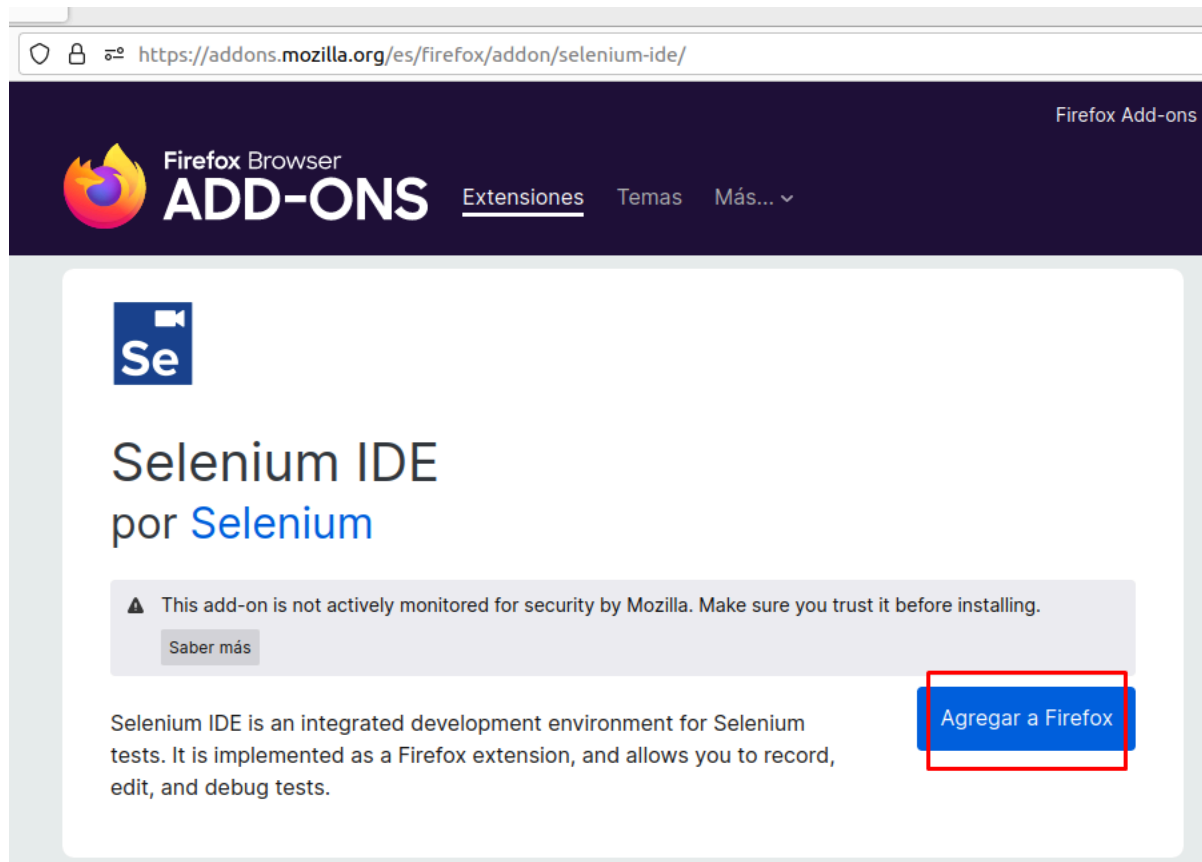
```

Se recoge del formulario el parámetro de ID (código) del elemento a borrar.

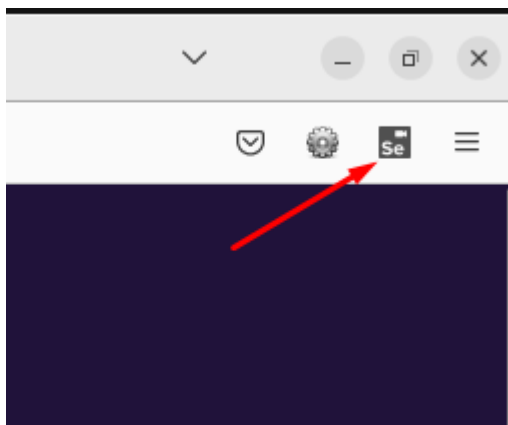
Se realiza el borrado en base de datos.

Tests de autonavegación mediante Selenium IDE

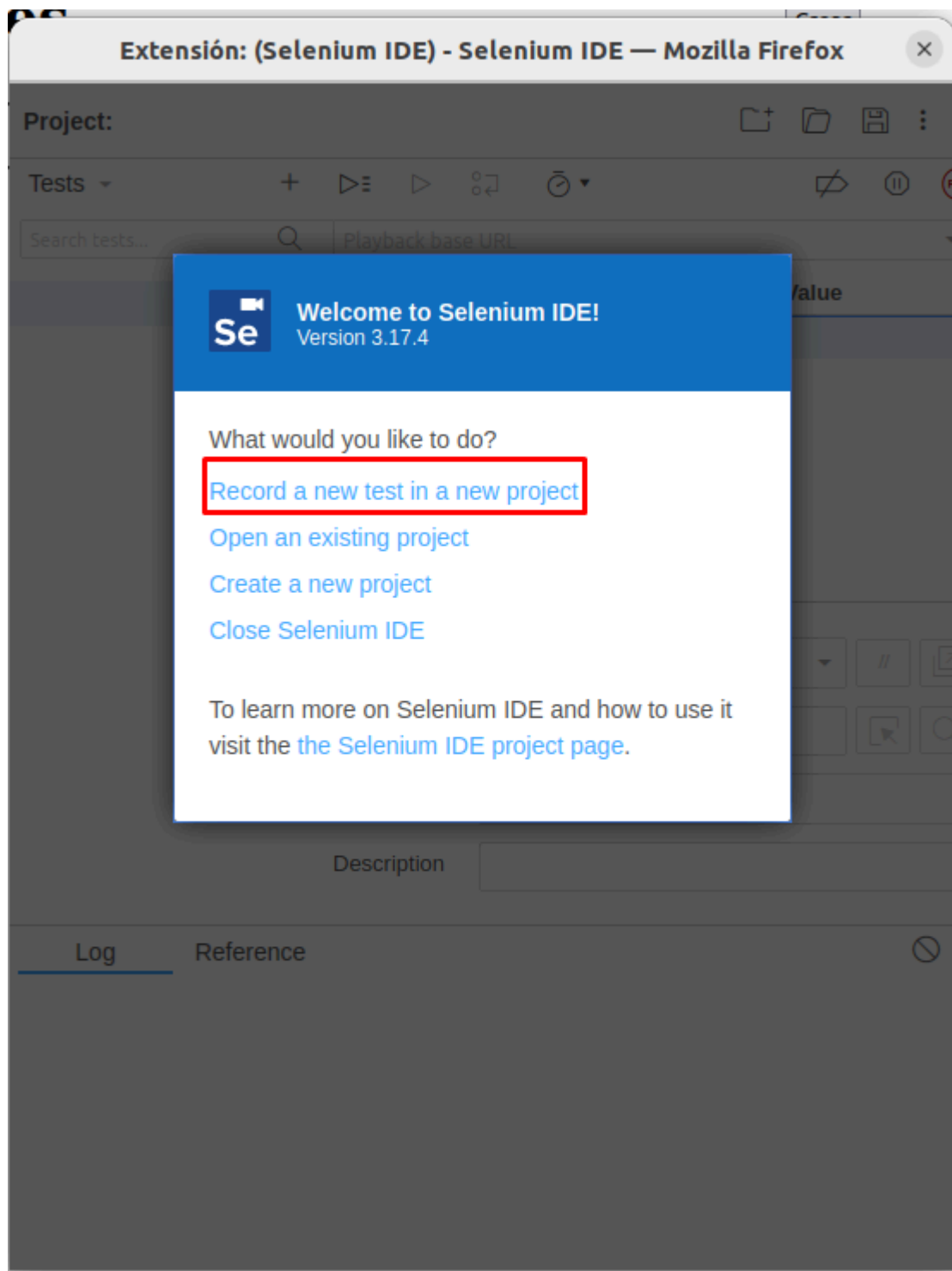
Añade al navegador la extensión de Selenium IDE.



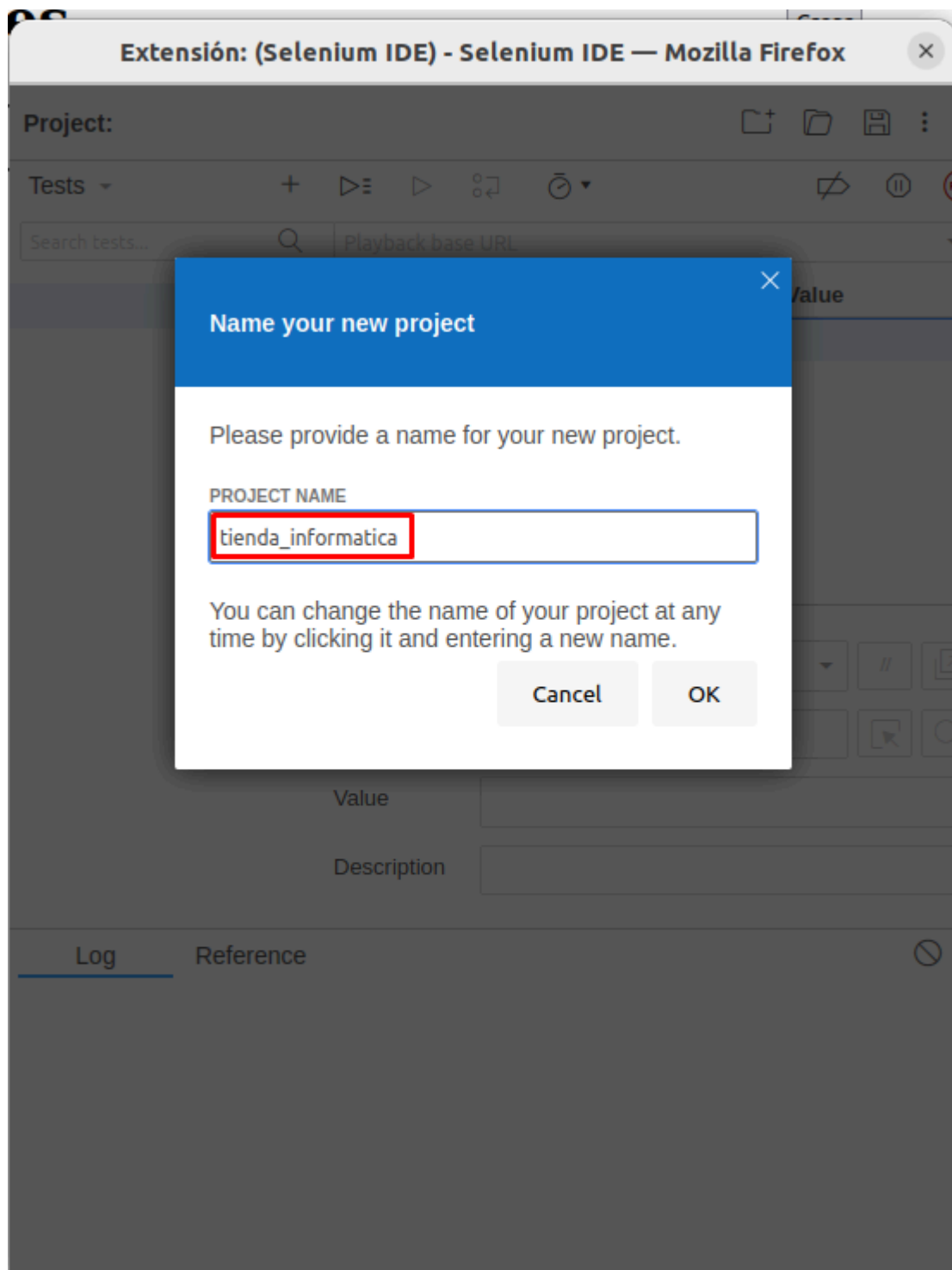
Te aparecerá un icono de Se en el panel de extensiones.



Sobre la página de tienda_informatica pulsa en el icono de Se y realiza una grabación de un nuevo test para un nuevo proyecto.



Dale el nombre al proyecto de tienda_informatica

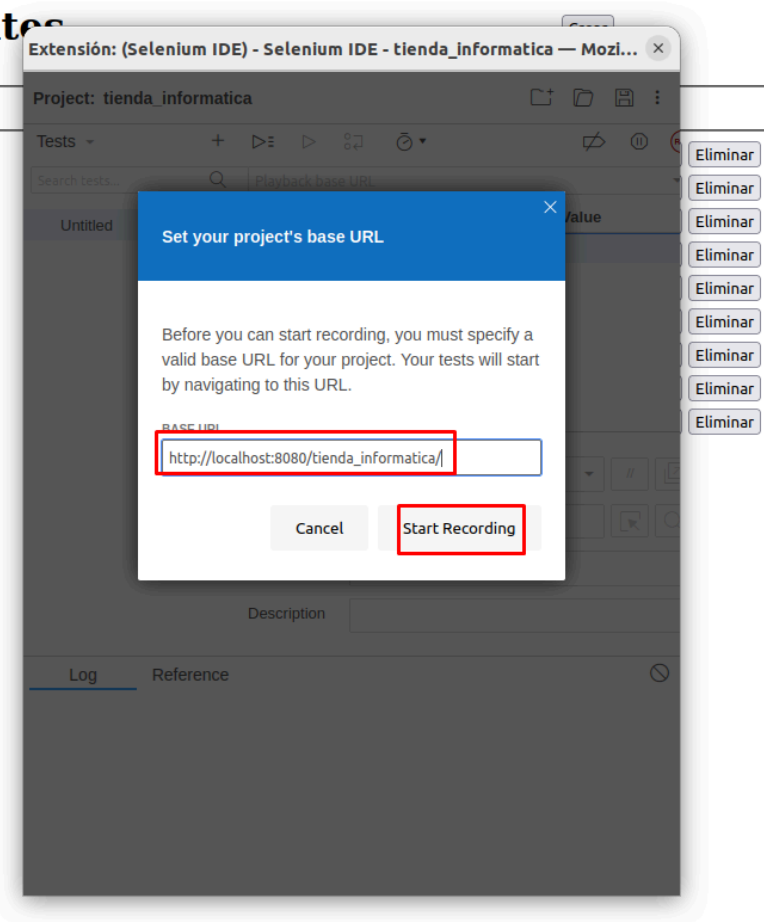


Dale la ruta base de la tienda_informatica e inicia la grabación del test:

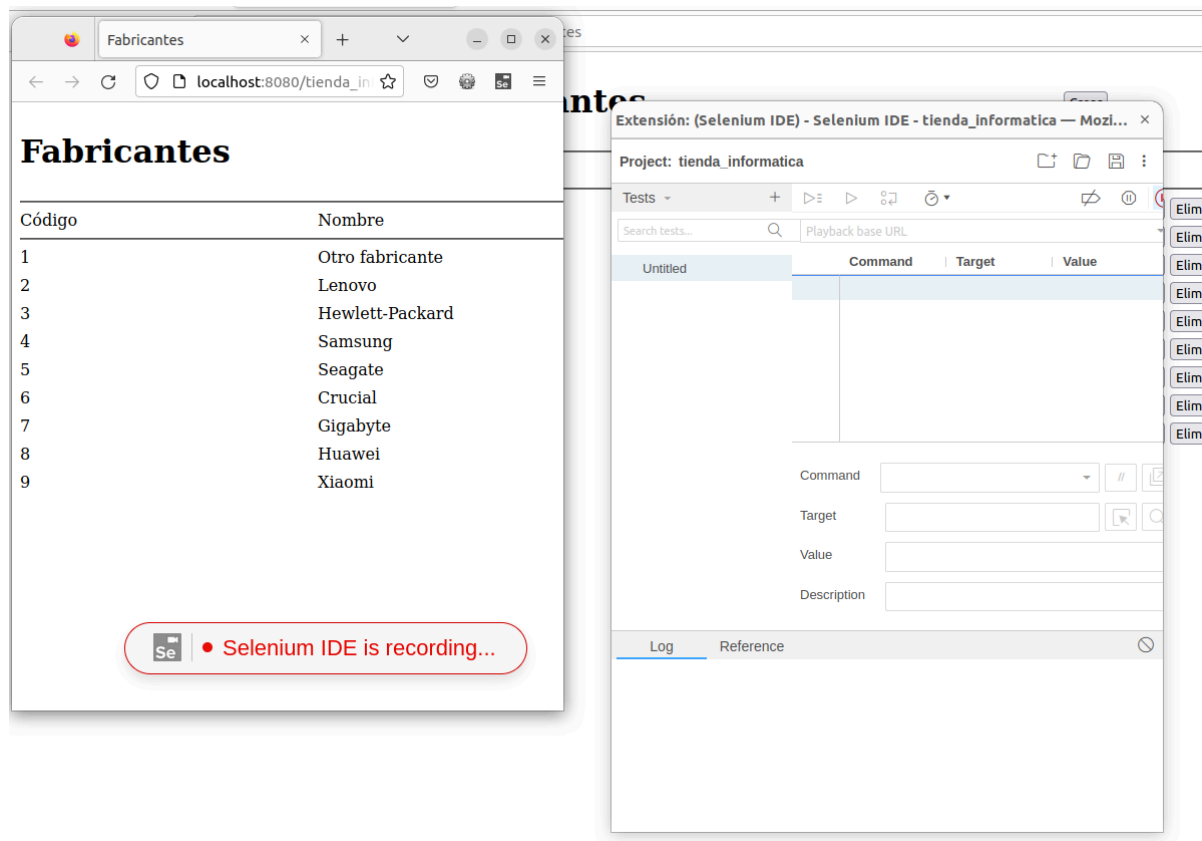
Fabricantes

Código

1
2
3
4
5
6
7
8
9

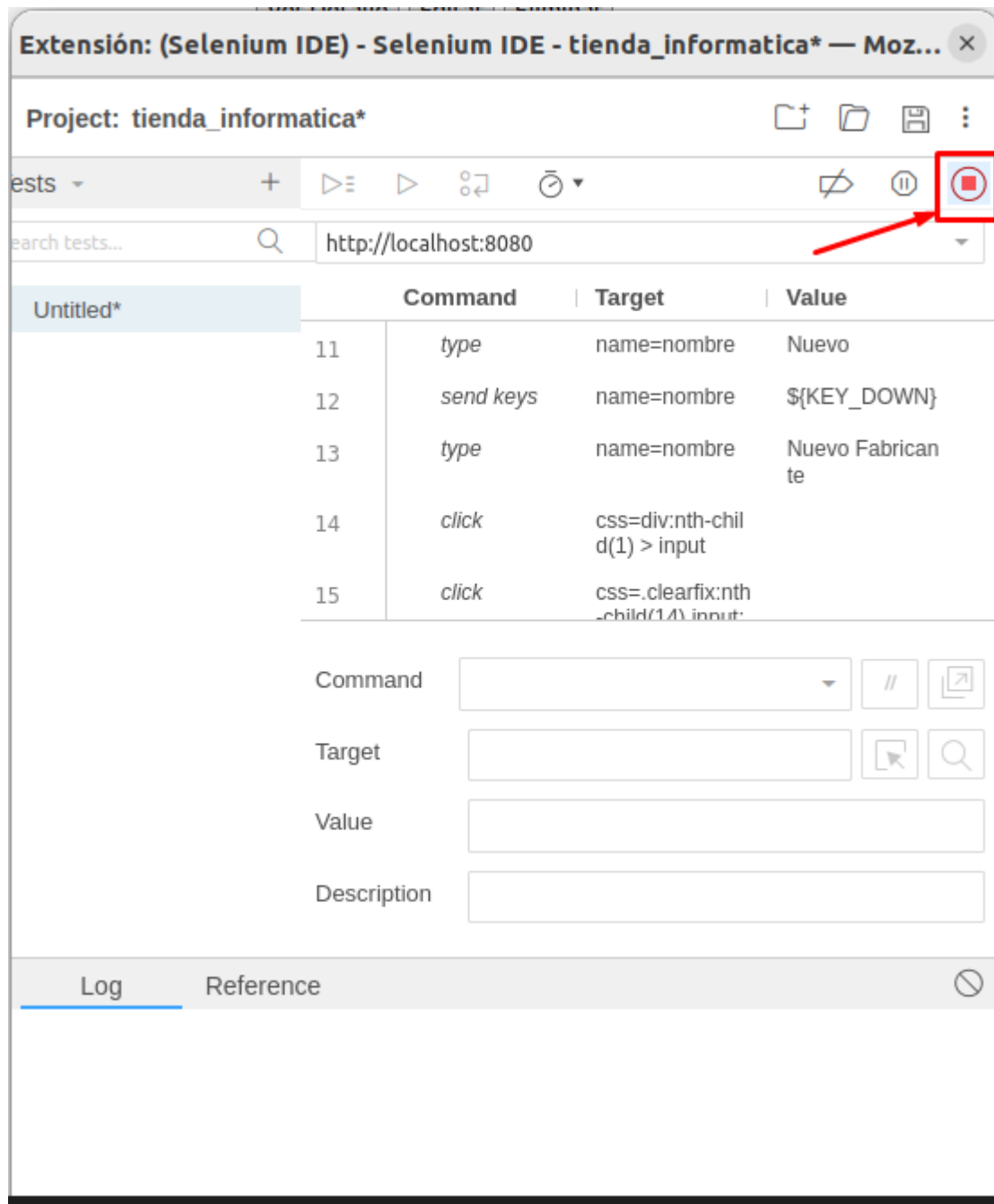


Se abrirá una ventana con la aplicación tienda_informatica con mensaje de Selenium IDE is recording...

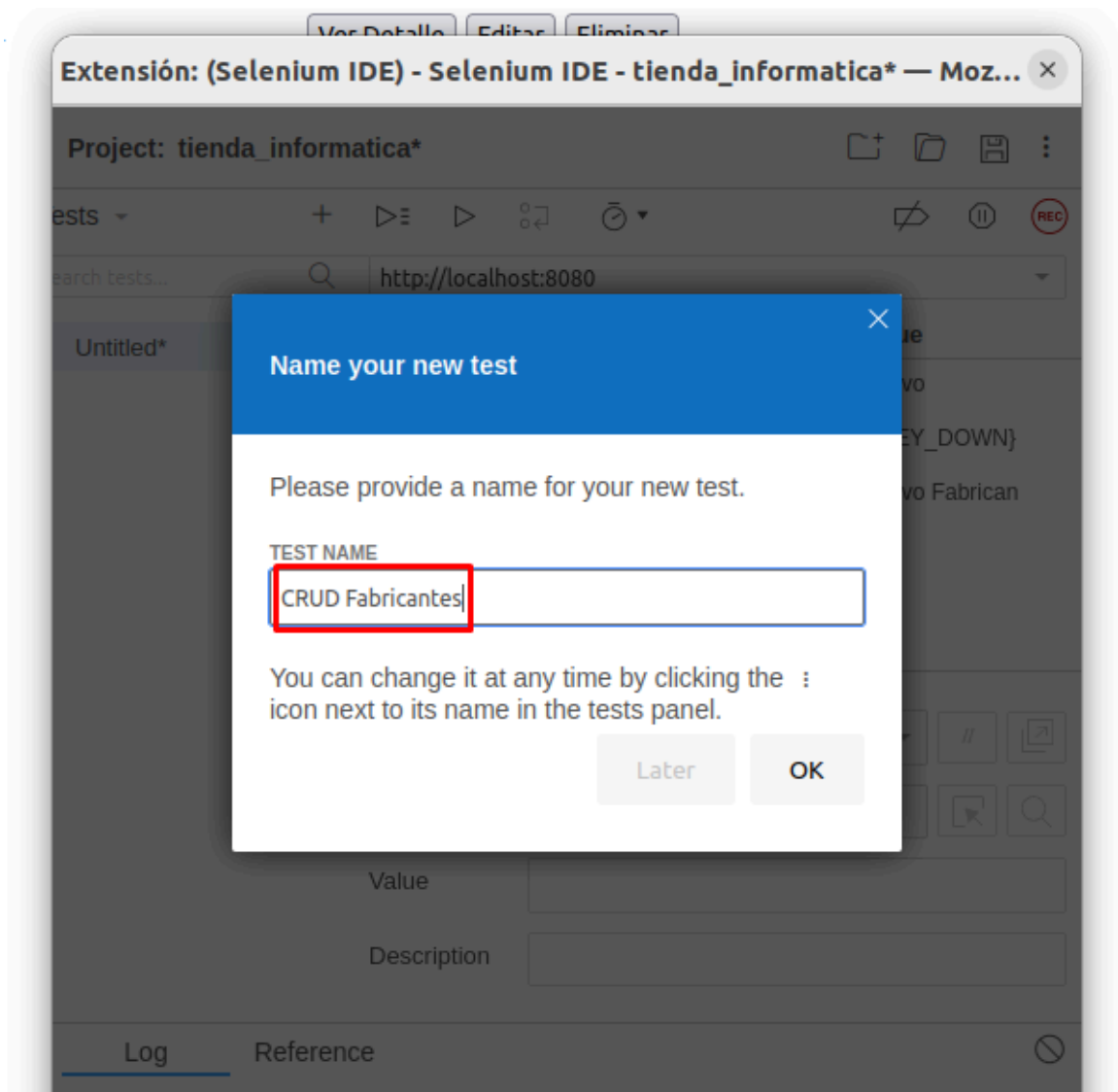


Crea un itinerario de Crear > Ver detalle > Editar > Eliminar.

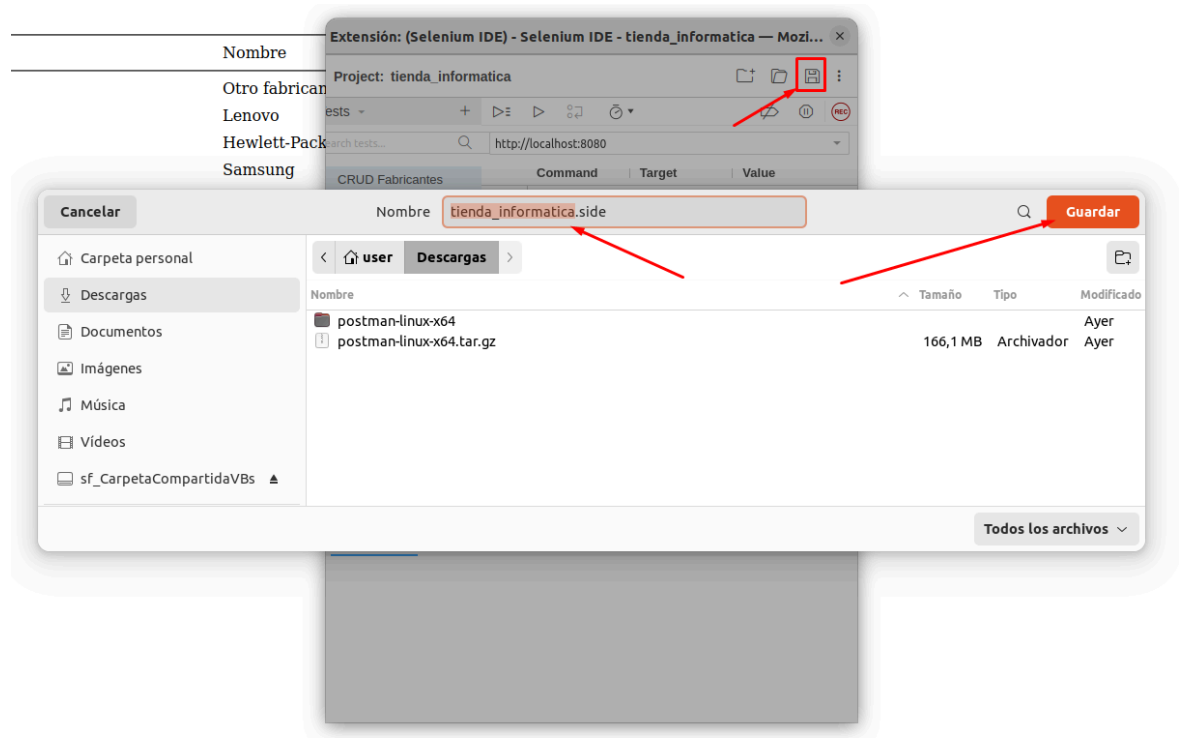
Cuando termines de realizar la ruta pulsa en el botón de parar.



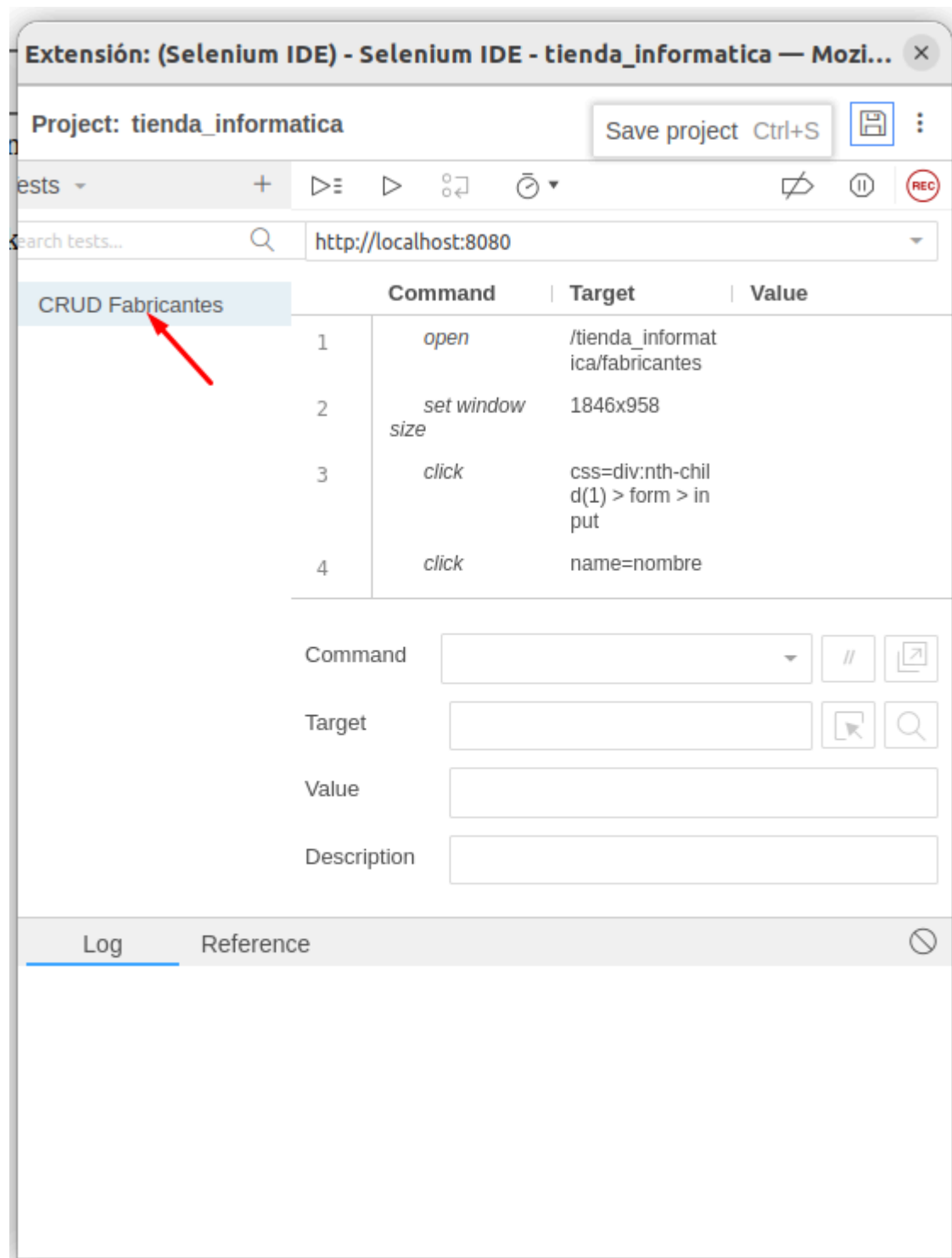
Dale un nombre al test de CRUD Fabricantes:



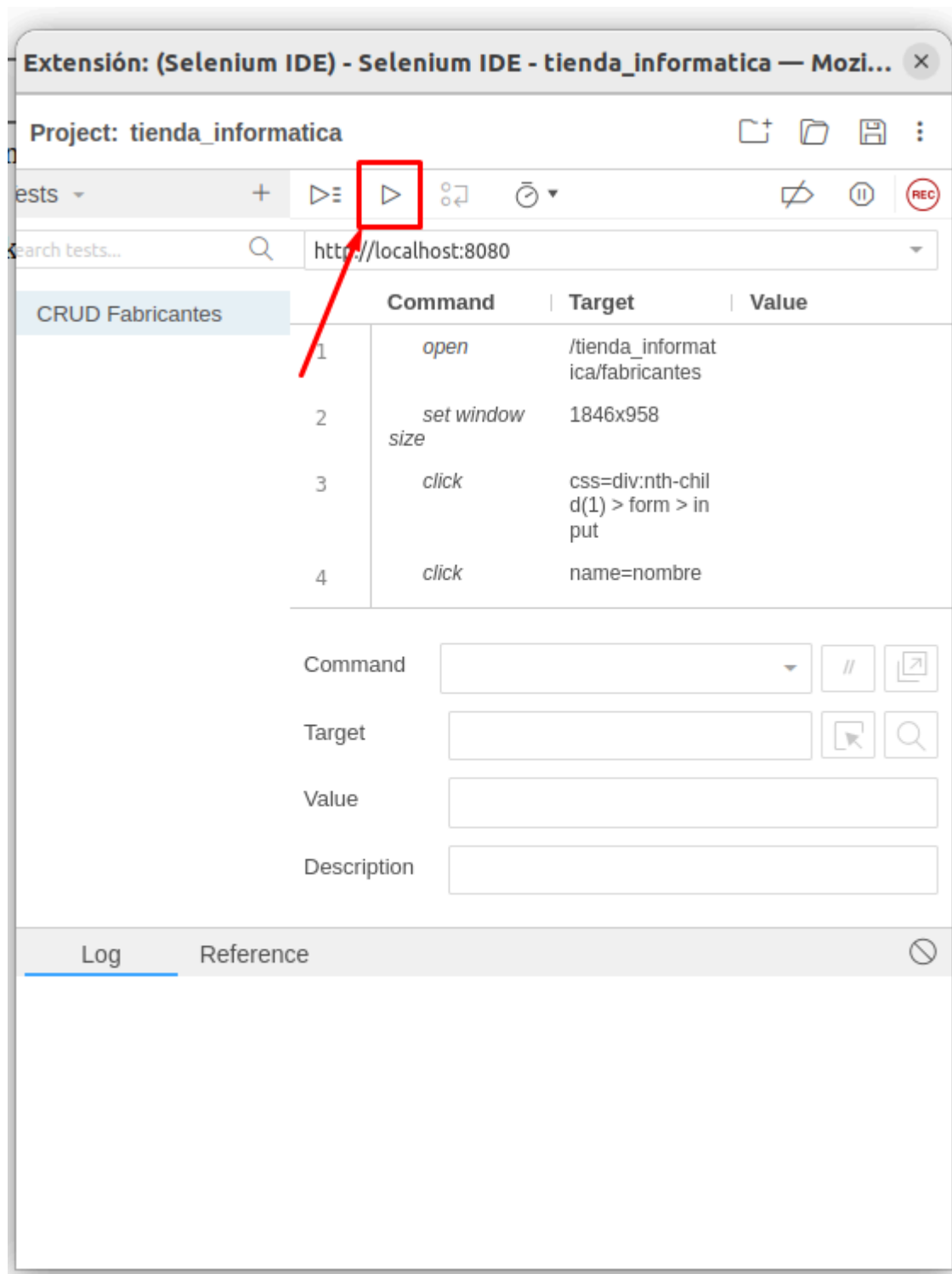
Pulsa en el disco para guardar el proyecto



Pinchando en el nombre del Test CRUD Fabricantes:



Y seguidamente en el botón de play para ejecutar el test:



Y observa si la aplicación pasa el test de CRUD Fabricantes:

Extensión: (Selenium IDE) - Selenium IDE - tienda_informatica — Moz...

Project: tienda_informatica

ests + ⏮ ⏪ ⏩ ⏭ ⌚ Test execution speed ⏴ ⏵ REC

Search tests... 🔍 http://loc Test execution speed

✓ CRUD Fabricantes

	Command	Target	Value
12	✓ send keys	name=nombre	\${KEY_DOWN}
13	✓ type	name=nombre	Nuevo Fabricante
14	✓ click	css=div:nth-child(1) > input	
15	✓ click	css=.clearfix:nth-child(14) input:nth-child(3)	

Command

Target

Value

Description

Log

Reference

⌵

10. click on name=nombre OK 19:41:09

11. type on name=nombre with value Nuevo OK 19:41:09

12. sendKeys on name=nombre with value \${KEY_DOWN} OK 19:41:10

13. type on name=nombre with value Nuevo Fabricante OK 19:41:10

14. click on css=div:nth-child(1) > input OK 19:41:10

15. click on css=.clearfix:nth-child(14) input:nth-child(3) OK 19:41:10

'CRUD Fabricantes' completed successfully 19:41:11

Observa cómo se pasa el test