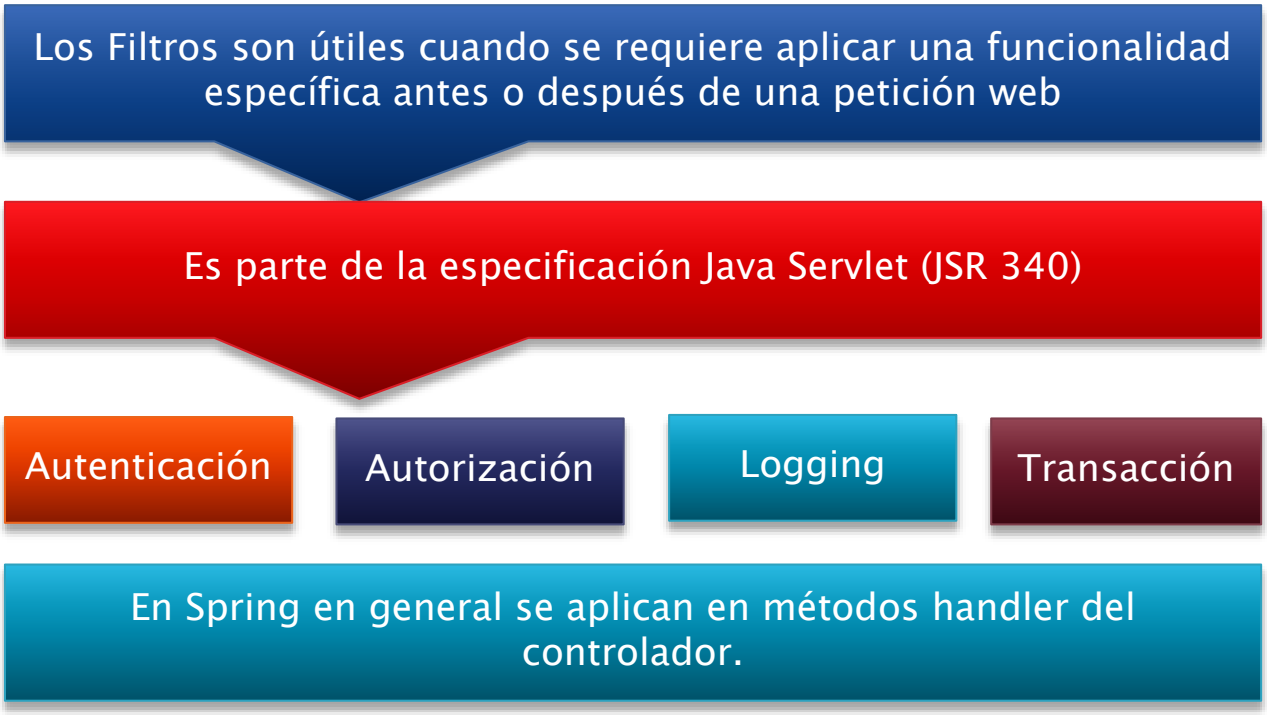
Filtros en Spring

¿Qué hace un Filtro?

Los filtros son útiles cuando se requiere aplicar funcionalidades específicas para ciertas peticiones, es decir son fragmentos de códigos reutilizables que interceptan una petición web (HTTP REQUEST) para agregar funcionalidad extra y transversal

- La idea es poder ejecutar una tarea o proceso antes o después de que se ejecute la petición (el controlador), modificando algún comportamiento de la petición o respuesta, como por ejemplo un sistema de autorización de usuario.
- Ejemplo: Autenticación, autorización, control de transacciones, logging, validaciones, etc...


Los Filtros son útiles cuando se requiere aplicar una funcionalidad específica antes o después de una petición web
Es parte de la especificación Java Servlet (JSR 340)
Autenticación
Autorización
Logging
Transacción
En Spring en general se aplican en métodos handler del controlador.

¿Qué hace un Filtro?

Los Filtros deben implementar la interfaz Filter o extender de la clase abstracta GenericFilterBean

- Método dofilter(): es un método de una clase Filter (API servlets) que nos permite implementar alguna tarea que necesitemos invocar antes o después de cada request.
- Cuando se invoca al chain.doFilter(...), continúa con la ejecución del controlador y si tiene más filtros asociados continúa con la ejecución en cadena.

Ejemplo Filtro

Algún filtro importante

```
public class FilterBean implements Filter {

   private static final Logger logger = LoggerFactory.getLogger(FilterBean.class);

   public void init(FilterConfig filterConfig) throws ServletException{}

   public void doFilter(ServletRequest request, ServletResponse response,
                        FilterChain chain) throws IOException, ServletException {

      logger.info("ALGÚN PROCESO ANTES");
      chain.doFilter(request, response);
      logger.info("ALGÚN PROCESO DESPUÉS");
   }

   public void destroy(){}
}
```

Filter → doFilter

```java
@RestController
@RequestMapping("/api/clientes")
public class ClienteRestController {

    @Autowired
    private IClienteService clienteService;

    @GetMapping(value = "/listar")
    public ClienteList listar() {
        return new ClienteList(clienteService.findAll());
    }

}
```

Ejecutar justo antes del request o método handler

Ejecutar después del request o método handler