JakartaEE Tips

Filtros e interceptores

JAKARTA ONE

LIVESTREAM

JAKARTA ONE
LIVESTREAM

Jorge Cajas

- Consultor en MangoChango, S.A.
- Lider en Guate-JUG
- Organizador de JConf Guatemala
- Certificado con +8 años experiencia
- Speaker desde hace 7 años

Contenido:

DRY - Don’t Repeat Yourself

- Filtros
  - RequestFilters
  - ResponseFilters
- Interceptores
  - ReadInterceptors
  - WriteInterceptors
- JPA Listeners

Filtros

JAKARTA ONE
LIVESTREAM

Siempre se ejecutan:
Se encuentre o no el recurso/endpoint el filtro siempre se ejecuta

Modifican propiedades:
Modifican las propiedades del request o response, como por ejemplo los HTTP Headers.

¿Donde?:
Pueden aplicarse tanto al cliente o servidor.

Tipos:
@PreMatching y @PostMatching

Filtros

JAKARTA ONE
LIVESTREAM

Siempre se ejecutan:
Se encuentre o no el recurso/endpoint el filtro siempre se ejecuta

Modifican propiedades:
Modifican las propiedades del request o response, como por ejemplo los HTTP Headers.

¿Donde?:
Pueden aplicarse tanto al cliente o servidor.

Tipos:
@PreMatching y @PostMatching

Filtros

PreMatching

Se ejecuta antes de buscar el recurso o endpoint.

Podemos por ejemplo modificar la URL del recurso a utilizar.

Podemos modificar los valores de los HTTP headers.

No podemos modificar el Body del request o response.

Se anotan con @PreMatching

Filtros

PostMaching

Se ejecuta después de buscar el recurso o endpoint, lo encuentre o no

No modificar la URL del recurso a utilizar, porque ya fué buscada.

Podemos modificar los valores de los HTTP headers.

No podemos modificar el Body del request o response.

Por default todos los filtros son PostMatching.

Filtros

```java
@Provider
@PreMatching
public class RedirectFilter implements ContainerRequestFilter {

  @Override
  public void filter(ContainerRequestContext reqContext) throws IOException {

      // redirect conditionally
      if (shouldRedirect(reqContext)) {
          reqContext.setRequestUri(URI.create("/temp"));
      }
  }

  private boolean shouldRedirect(ContainerRequestContext reqContext) {
      //todo it should be conditional return
      return true;
  }
}
```

Filtros

Ejecución condicional

```
@Secured
@Provider
@Priority(Priorities.AUTHENTICATION)
public class AuthFilter implements ContainerRequestFilter {


    @Override
    public void filter(ContainerRequestContext requestContext) throws IOException {

      String token = requestContext.getHeaderString("Authorization");
      User user = .... //Search for user
      if(user == null) trhow new Exceception("No autorizado")

    }
}
```

Filtros

Ejecución condicional

```
@NameBinding
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.TYPE})
public @interface Secured {
}
```

```
@Secured
@Path("/users")
public class UsersController {

  @GET
  public List<User> users(){
    ///magic here
  }
}
```

Interceptores

JAKARTA ONE LIVESTREAM

No Siempre se ejecutan:
Solo se ejecutan si el recurso fué encontrado y contiene un message body.

Modifican el message body:
Modifican la entrada o la respuesta del message body recibido o enviado por el servidor.

¿Donde?:
Pueden aplicarse tanto al cliente o servidor.

Tipos:
ReadInterceptor y WriteInterceptor

Interceptores

ReadInterceptor

```java
@Provider
public class RequestServerReaderInterceptor implements ReaderInterceptor {

    @Override
    public Object aroundReadFrom(ReaderInterceptorContext context) throws IOException {

      InputStream is = context.getInputStream();
      String body = new BufferedReader(new InputStreamReader(is)).lines()
                        .collect(Collectors.joining("\n"));

        context.setInputStream(new ByteArrayInputStream(
          (body + " message added in server reader interceptor").getBytes()));

        return context.proceed();
    }
}
```

Interceptores

WriteInterceptor

```java
@Provider
public class RequestClientWriterInterceptor implements WriterInterceptor {

    @Override
    public void aroundWriteTo(WriterInterceptorContext context) throws IOException {

      Object entity = context.getEntity();
      StandardResponse response = new StandardResponse();
      response.setData(entity);
      response.setSuccess(true);
      response.setTime(LocalDateTime.now())

      context.setEntity(resoponse)

        context.proceed();
    }
}
```

JPA Listeners

JAKARTA ONE
LIVESTREAM

Se ejecutan en el ciclo de vida de una entidad

@PrePersist

@PreUpdate

@PreDelete

JPA Listeners

JAKARTA ONE
LIVESTREAM

```java
@RequestScoped
public class BaseModelListener {

    @PrePersist
    private void onPersist(BaseModel model) {
        model.setCreatedAt(LocalDateTime.now());
        model.setCreatedBy(id);
    }


    @PreUpdate
    private void onUpdate(BaseModel model) {
       model.setUpdatedAt(LocalDateTime.now());
    }

    @PreRemove
    private void onRemove(BaseModel model) {
        model.setDeletedAt(LocalDateTime.now());
    }

}
```

JPA Listeners

JAKARTA ONE
LIVESTREAM

```java
@RequestScoped
public class BaseModelListener {

    @PrePersist
    private void onPersist(BaseModel model) {
        model.setCreatedAt(LocalDateTime.now());
        model.setCreatedBy(id);
    }


    @PreUpdate
    private void onUpdate(BaseModel model) {
       model.setUpdatedAt(LocalDateTime.now());
    }

    @PreRemove
    private void onRemove(BaseModel model) {
        model.setDeletedAt(LocalDateTime.now());
    }

}
```

JPA Listeners

```java
@MappedSuperclass
@EntityListeners(BaseModelListener.class)
public class BaseModel {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private LocalDateTime createdAt;

    @Column
    private LocalDateTime updatedAt;

    @Column
    private LocalDateTime deletedAt;

}
```

¿Preguntas?

JAKARTA ONE
LIVESTREAM

¡Gracias!

Jorge Cajas
@cajasmota
MangoChango, S.A.