Event - REST-Aufruf

CustomRestRule-Regeln ermöglichen den Aufruf von Java-Regeln direkt über den REST-Service.

Sie sind unabhängig von Businessobjekten und müssen deshalb nicht über den Regelmanger zugewiesen werden.

Die Berechtigung einer CustomRestRule wird über eine Zuweisung im Unterformular "REST-Regeln" in der jeweiligen Benutzergruppe durchgeführt.

Über ein Context-Objekt erhält man Zugriff auf z.B. aufrufenden Benutzer oder Sprache.

Beispiel

```
package example.rest;

import java.util.Arrays;
import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import org.nuclos.api.rule.CustomRestRule;

@Path("example")
public class CustomRestWithPathTestRule implements CustomRestRule {

	@GET
	@Path("customers")
	@Produces("application/json")
	public List<Customer> customers() {
		return Arrays.asList(
				new Customer(1, "Mustermann"),
				new Customer(2, "Maier")
		);
	}

	public static class Customer {
		private int id;
		private String name;

		public Customer(final int id, final String name) {
			this.id = id;
			this.name = name;
		}

		public int getId() {
			return id;
		}

		public String getName() {
			return name;
		}
	}
}
```

Der Aufruf des REST-Service wird über @Path Annotationen gesteuert.

CustomRestRegeln besitzen immer den URL-Prefix "/rest/custom/".

Der obige REST-Service liefert nach Aufruf von http://localhost/nuclos/rest/custom/example/customers folgendes Ergebnis:

Ergebnis

```
[
        {
                id: 1,
                name: "Mustermann"
        },
        {
                id: 2,
                name: "Maier"
        }
]
```

Über ein Context-Objekt kann man außerdem Zugriff auf z.B. den aufrufenden Benutzer oder die Sprache erhalten. Dieses wird innerhalb der Klasse wie folgt eingebunden:

Context

```
@Inject
protected Provider<CustomRestContext> context;

// Dazu müssen noch drei weitere Klassen importiert werden:

import javax.inject.Inject;
import javax.inject.Provider;
import org.nuclos.api.context.CustomRestContext;
```

Behandlung von Exception und Custom Responses, jegliche Exceptions die in der Methode nicht mit einem try/catch-Block behandelt wird, wird vom Container-Service (Tomcat/Glassfish) als 500-Fehler gemeldet.
Um dieses Verhalten anzupassen ist es notwendig die Exceptions mittels try/catch zu behandeln und korrekte eigene Responses zu liefern, als Beispiel:

Behandlung von BusinessException

```
package example.rest;


import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import org.nuclos.api.rule.CustomRestRule;
import org.nuclos.api.exception.BusinessException;

@Path("Test")
@Rule(name="HandleException", description="Demonstrates how to handle exception in CustomRest")
public class HandleException implements CustomRestRule {

        @GET
    @Path("response/{parameter}")
    @Produces("application/json")
    public Response parameterTest(@PathParam("parameter") String parameter) {
                try {
                if( this.functionToThrowException(parameter) ) {
                    return Response.status(Response.Status.FORBIDDEN).entity(new Message("Dieser User hat keine
Berechtigung für das Ausführen der Regel.")).build();
                        }
                } catch (BusinessException e) {
            // may also log error
            return Response.status(Response.Status.BAD_REQUEST).entity(new Message(e.getMessage())).build();
                }

        return Response.status(Response.Status.NO_CONTENT).build();
    }

        private boolean functionToThrowException(String parameter) throws BusinessException {
                if (parameter == null) {
                        throw BusinessException("No parameter given");
                }

                return "failure".equals(parameter);
        }

        // Any DTO class for response, will be JSON converted
    public static class Message {

        String message;

        public Message(String message) {
            this.message = message;
        }

        public String getMessage() {
            return message;
        }
    }
}
```