CREATING GENERIC REST INTERFACES FOR RESTEASY IN A MODEL-DRIVEN WAY

JUDCon 2013, India

Markus Gulden
<table>
<thead>
<tr>
<th>Segment</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem</strong></td>
<td>Interfaces of composed systems might not fit for external clients:</td>
</tr>
<tr>
<td></td>
<td>- Not web service-enabled</td>
</tr>
<tr>
<td></td>
<td>- Technologically heterogeneous</td>
</tr>
<tr>
<td></td>
<td>- Complex data model</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td>Use case-specific interface for each client type</td>
</tr>
<tr>
<td></td>
<td>- Additional web service layer</td>
</tr>
<tr>
<td></td>
<td>- Unified and RESTful interface</td>
</tr>
<tr>
<td></td>
<td>- Simplified data model</td>
</tr>
<tr>
<td><strong>Runtime</strong></td>
<td>RESTEasy as JAX-RS implementation for REST-interface</td>
</tr>
<tr>
<td></td>
<td>JAXB for data model</td>
</tr>
<tr>
<td></td>
<td>Reflection based transformation between data models</td>
</tr>
<tr>
<td><strong>Tooling</strong></td>
<td>XText based DSL for interface specification and code generator</td>
</tr>
<tr>
<td></td>
<td>Data model as XML schema by Eclipse XSD SDK</td>
</tr>
</tbody>
</table>
Index

Zielpuls Introduction
Trends and Motivation
Approach
Implementation
Demonstration
Summary

January 18th 2013
Creating generic REST interfaces for RESTEasy in a model-driven way
Zielpuls GmbH
- Technology consulting in automotive, aerospace, chemical and mechanical engineering industry
- Integration of technical and methodical expertise
- Interfaces (technical and organizational) are our bread and butter

Markus Gulden
- Technology consultant at Zielpuls
- Focus on Connected Car/Car IT and New Mobility
- 5 years of experience in Java EE and JBoss

January 18th 2013
Creating generic REST interfaces for RESTEasy in a model-driven way
Trends and Motivation

IT-systems become more and more the backbone of innovative enterprises

**Trends**
- Implementation of internal processes and customer services
- Composition of existing systems to reduce cost and risks
- Existing approaches (SOA) and products (ESB, message broker)

**Solution**
- Approach for use case-specific client interfaces

**Issues for external clients**
- Not web service-enabled or even legacy interfaces
- Technologically heterogeneous and oversized interfaces
- Complex data model

Creating generic REST interfaces for RESTEasy in a model-driven way

January 18th 2013
Approach

One interface per client-type is required:

- Not web service-enabled: Additional web service-layer
- Heterogeneous and oversized interfaces: REST-based interfaces
  - Unified
  - Only required services are provided
- Complex data model: Use case-specific, simplified data model

One use case-specific interface for each client-type

January 18th 2013
**Approach in Detail (1)**

**Additional layer**
- Implementation as web service
- REST-based [1]
  - HTTP as application protocol
  - Data representation as XML
- Additional layer encapsulates existing interfaces
  - Unified interface for several services

---

Simplified data model

- Employment of the *Data Transfer Object (DTO)* [2, 3] pattern
- Encapsulation of multiple objects into one DTO
- Only required attributes are mapped
- Automatic transformation between data models at runtime

Example:
Car sharing reservation list
- Time
- Station
- License number

```
RentingStation
- id : long
- stationName: String
- latitude : double
- longitude : double

Reservation
- id : long
- begin : Date
- end : Date
- status : int

Vehicle
- licenseNo : String
- type: String
- registration : Date

ReservationDTO
- stationName: String
- begin : Date
- end : Date
- licenseNo : String
```

Implementation - Runtime

- JAX-RS
- Part of Java EE 6
- RESTEasy by JBoss

- JAXB
- Marshalling as XML/JSON

- JNDI
- More technologies possible

- Reflection based implementation [3]

Interface

Data model

Call of target services

Transformation of data models
Implementation - Runtime

JAX-RS

@Path("/reservations")
public class ReservationService {

    @GET
    @Produces("application/xml")
    public List<ReservationDTO> getReservations() {
        //Call business logic
    }

    @PUT
    @Consumes("application/xml")
    public void updateReservation(ReservationDTO state) {
        //Call business logic
    }
}
JAXB

```java
@XmlType(name = "ReservationDTO")
public class ReservationDTO {

  protected String stationName;
  protected String licenseNumber;
  @XmlElement(required = true)
  protected Date begin;
  @XmlElement(required = true)
  protected Date end;

  // Getters
  // Setters
}
```

Creating generic REST interfaces for RESTEasy in a model-driven way
Implementation - Tooling

Data model design

- XML Schema Definition SDK contained in Eclipse
- Data model as XML schema
- Out-of-the-box-generation of JAXB classes

Interface design & call of target services

- Tool is based on XText framework
- Simple DSL, which
  - Models the REST structure of the new interface
  - Maps REST resources on existing services
- Belonging generator creates
  - Creates JAX-RS code
  - JNDI calls for target services
Creating generic REST interfaces for RESTEasy in a model-driven way

January 18th 2013

Implementation - Tooling

Interface DSL

- Interface
  - name: String
- RestUrl
  - relativeUrl: String
- Service
  - identifier: String
- Method
  - name: String
  - returnType
  - inputType
- DataType
  - identifier: String

Interface structure provided to client

Target services

DTOs for simplified data model
Implementation - Development Process

1. Development/Composition
   - Creation of method signatures

2. Application creation
   - JAXB
   - Interfaces

3. Configuration
   - JAXB
   - JAX-RS

4. Deployment
   - Design and generation
     - REST classes
     - DTOs

5. Execution
   - Execution once
   - Execution n times
Demonstration

Sequence

- Initial situation
- Data model and interface design
- Deployment and demonstration
- Re-design of data model
- Deployment and demonstration

1. One service with one method deployed as EJB
2. One global data model
3. Specification of DTOs as XML schema and interface structure
4. Generation of JAXB JAX-RS classes
5. Deployment and demonstration of system
6. DTO modification: Additional attribute
7. Generation of JAXB classes
8. Deployment and demonstration of system

January 18th 2013
Summary

Runtime

Solution: Use case-specific interface for each client type

- Additional web service layer
- Unified and RESTful interface
- Simplified data model

Benefits

- Less runtime resources required on client side
- Reduced effort at implementation on client side

Tooling

Solution: Model-driven approach

Benefits

- Less implementation effort on server and client side
- Ability to reuse of existing systems

January 18th 2013
Creating generic REST interfaces for RESTEasy in a model-driven way
WE HELP YOU TO SETUP AN EFFECTIVE INTERFACE MANAGEMENT.
RESTEasy

- Various frameworks to help you build RESTful Web Services and RESTful Java applications
- Fully certified and portable implementation of the JAX-RS specification
- RESTEasy can run in any Servlet container
- Embedded server implementation for junit testing
- JAXB marshalling into XML, JSON, Jackson, Fastinfoset, and Atom as well as wrappers for maps, arrays, lists, and sets of JAXB Objects