ARS

Member of TIMETOACT GROUP SOFTWARE & CONSULTING

Einführung in JAX-RS

Java EE Grundlagen

ARS

Inhalte dieses Kapitels

Einführung

Implementierung

ARS

Einführung [1|2]

Java API for RESTful Web Services

- JSR 339 (http://jcp.org/en/jsr/detail?id=339)
- Sammlung von Interfaces und Annotationen
- Servlet API als Basis
- Vereinfachung der Entwicklung von REST Services

ARS

Einführung [2|2]

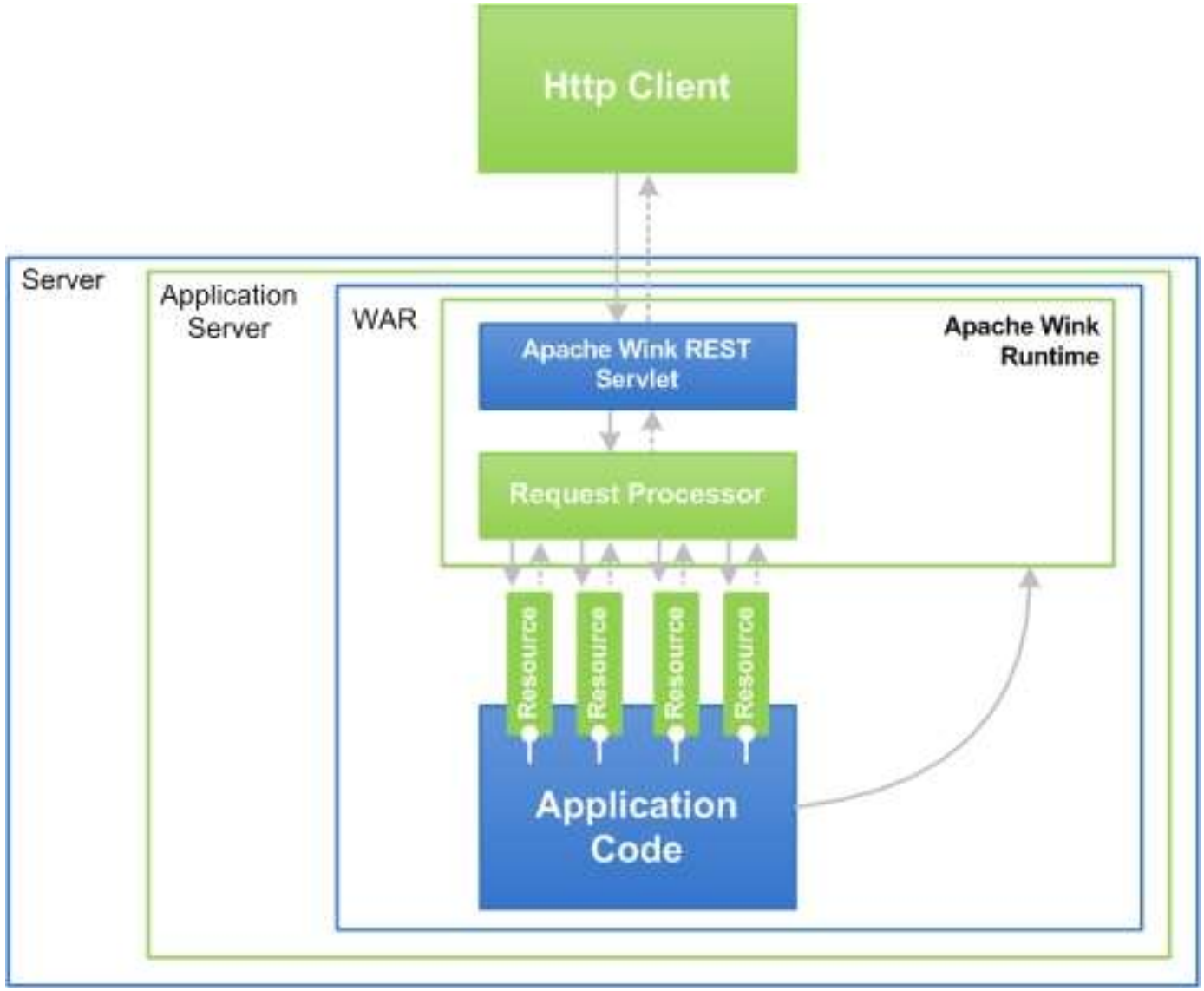
WebSphere Application Server v8.x

- Basis: Apache Wink 1.1 (JAX-RS 1.1)
  (Zukünftig: Apache CXF)
  Bestandteile:
  - Apache Wink JAX-RS 1.1 server runtime
  - Apache Wink stand-alone client API
  - JSON4J/Jackson Entity provider
  - Atom JAXB model und Apache Abdera support

- WebSphere Application Server Liberty Profile
  - JAX-RS 1.1 (Java EE 6) mit Apache Wink
  - JAX-RS 2.0 (Java EE 7) mit Apache CXF

ARS

Architektur der Apache Wink Library



ARS

Der Weg zum „Hello World“ Service

1. Application Klasse erzeugen
   - Automatisches Ressourcen-Scanning (über Annotation-Scanner)
   - oder programmatische Registrierung aller Ressourcen Klassen
2. Ressourcen Klasse (Endpoint) erzeugen

ARS

Application-Klasse erzeugen [1|2]

- Notwendige Schritte:
  - Unterklasse von `javax.ws.rs.core.Application` erstellen
  - Annotation `@ApplicationPath("/api")` hinzufügen
- Beispiel:

```
package de.ars.jaxrs.application;
import javax.ws.rs.ApplicationPath;

@ApplicationPath("/api")
public class HelloWorldApplication extends javax.ws.rs.core.Application {

}
```

- Achtung:
  Funktioniert nur bei einer Application Klassen pro Webmodul (WAR)!

ARS

Application-Klasse erzeugen [2|2]

Programmatische Registrierung

- Falls mehrere Basispfade („api“ und „xxx“) pro Webmodul
- Überschreiben von “`getClasses()`” und “`getSingletons()`”

```
@ApplicationPath("/api")
public class HelloWorldApplication extends javax.ws.rs.core.Application {
    private final Set<Object> singletons = new HashSet<Object>();
    private final Set<Class<?>> classes = new HashSet<Class<?>>();
    public HelloWorldApplication() {
        this.singletons.add(new RessourcenKlasse());
        this.classes.add(HelloWorldResource.class);
    }
    @Override
    public Set<Class<?>> getClasses() {
        return this.classes;
    }
    @Override
    public Set<Object> getSingletons() {
        return this.singletons;
    }
}
```

ARS

Der Weg zum „Hello World“ Service

1. Application Klasse erzeugen
   - Automatisches Ressourcen-Scanning (über Annotation-Scanner)
   - oder programmatische Registrierung aller Ressourcen Klassen
2. Ressourcen Klasse (Endpoint) erzeugen

ARS

Erstellung der Ressourcenklase (Endpoint)

Beispiel

- Ressource über Pfad `/hello` oder `/hello/<name>` erreichbar
- Ressource soll per HTTP-GET angesprochen werden
- MIME-Format der Antwort als `text/plain`

```
@Path("/hello")
public class HelloWorldResource {
    @GET
    @Produces("text/plain")
    public String sayHello() {
        // ...
    }
    @GET
    @Path("{name}")
    @Produces("text/plain")
    public String sayHello2(@PathParam("name") final String param) {
        // ...
    }
}
```