



Niklas Gustavsson  
niklas@protocol7.com  
<http://protocol7.com>  
<http://twitter.com/protocol7>

# **REST made simple with Java**

**REST?**

# HTTP 1.1



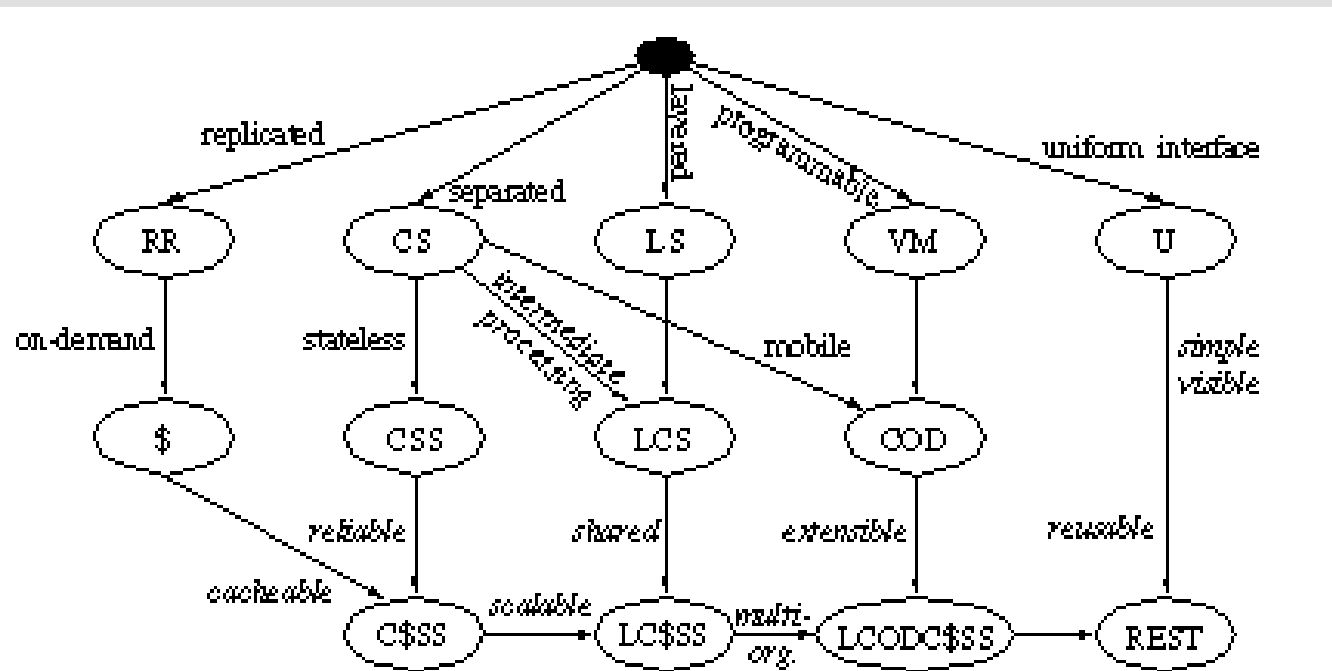
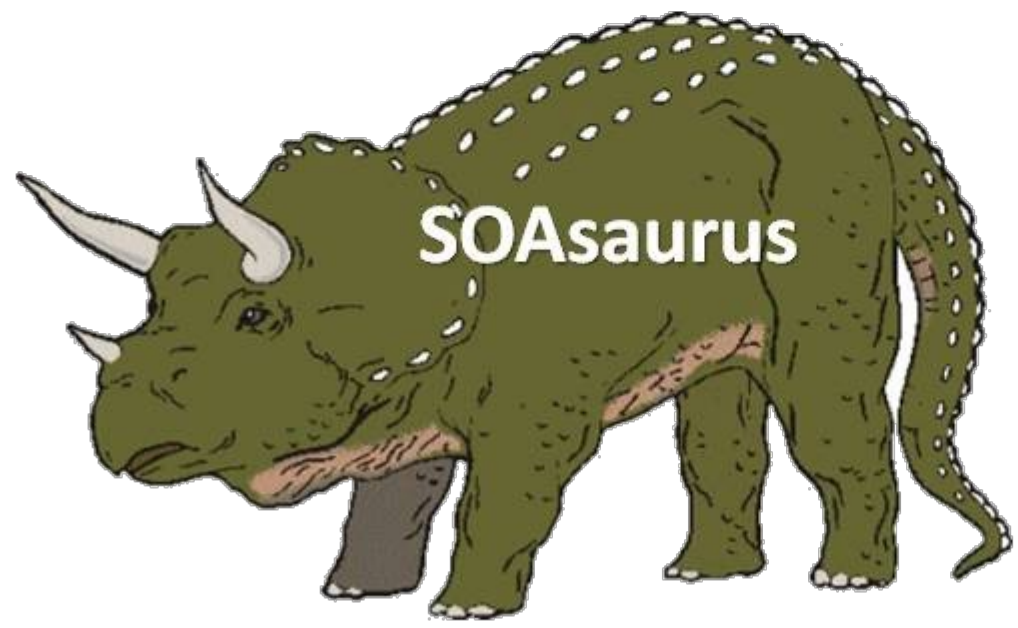


Figure 5-9. REST Derivation by Style Constraints

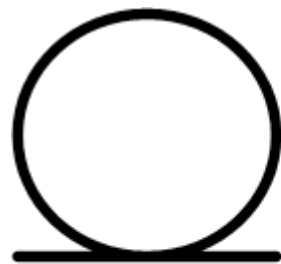
HTTP done right



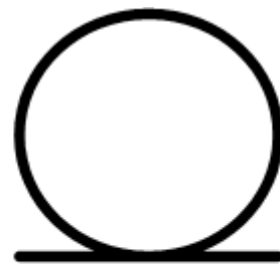
# Principles

# Principles

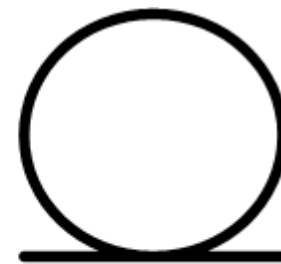
Everything is a resource



**Order**



**Customer**



**Customer  
with orders**

# Principles

A resource has an identifier

<http://example.com/customers/1453>

# Principles

We transfer representations

# Principles

All resources expose a uniform interface

**GET, POST, PUT, DELETE**

# Principles

Hypermedia as the engine of application state

# Principles

Client-server, Stateless, Cacheable, Layered

Why?

Why?

It's easy! Well, it's not

Why?

It's what the cool kids use

Why?

Web has been successful, copy!



Why?

It's what others use

Why?

Interoperability on the right level

Frameworks, yeay!

# JAX-RS

(aka JSR-311)

**Jersey** <http://jersey.dev.java.net>

**Restlets** <http://www.restlet.org>

**RESTeasy** <http://www.jboss.org/resteasy>

**CXF** <http://cxf.apache.org>

POJO based  
Annotation heavy

# Resources

Code! Show me the code!

```
public class TimeReportService {  
    private TimeReportDao reportDao;  
  
    public TimeReport getReport(String username) {  
        return reportDao.forUser(username);  
    }  
  
    public void saveReport(TimeReport report) {  
        reportDao.update(report);  
    }  
  
    ...  
}
```

# Request mapping

```
@Path("report")
public class TimeReportService {

    private TimeReportDao reportDao;

    @GET
    public TimeReport getReport(String username) {
        return reportDao.forUser(username);
    }

    @PUT
    public void saveReport(TimeReport report) {
        reportDao.update(report);
    }

    ...
}
```

# Application

## Resource

GET /report HTTP/1.1  
Accept: text/calendar

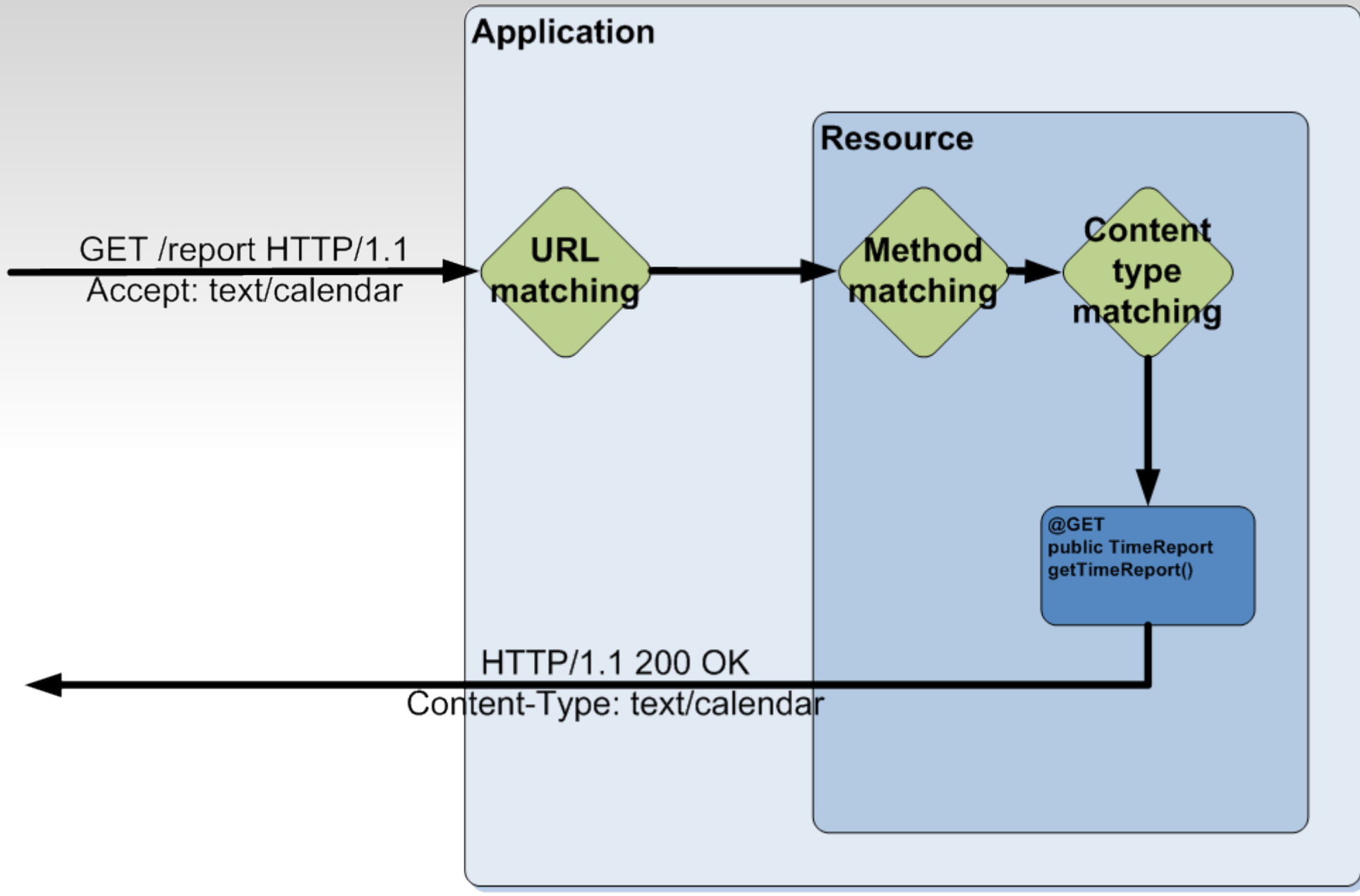
URL  
matching

Method  
matching

Content  
type  
matching

```
@GET  
public TimeReport  
getTimeReport()
```

HTTP/1.1 200 OK  
Content-Type: text/calendar



```
public class TimeReportApplication extends Application {  
  
    @Override  
    public Set<Class<?>> getClasses() {  
        Set<Class<?>> resources = new HashSet<Class<?>>();  
  
        resources.add(TimeReportService.class);  
  
        return resources;  
    }  
}
```

# Path parameters and contexts

```
@Path("{username}/report")
public class TimeReportService {

    @GET
    public TimeReport getReport(@PathParam("username")
        String username) {
        return reportDao.forUser(username);
    }

    ...
}
```

<http://example.com/niklas/report>

```
@GET
public TimeReport getReport(@Context SecurityContext sc) {
    if(sc.isSecure()) {
        Principal user = sc.getUserPrincipal();
        return reportDao.forUser(user.getName());
    } else {
        ... throw error, redirect to login
    }
}
```

```
@GET
public TimeReport getReport(@CookieParam("username")
    String username) {
    return reportDao.forUser(username);
}
```

# Application

## Resource

GET /report HTTP/1.1  
Accept: text/calendar

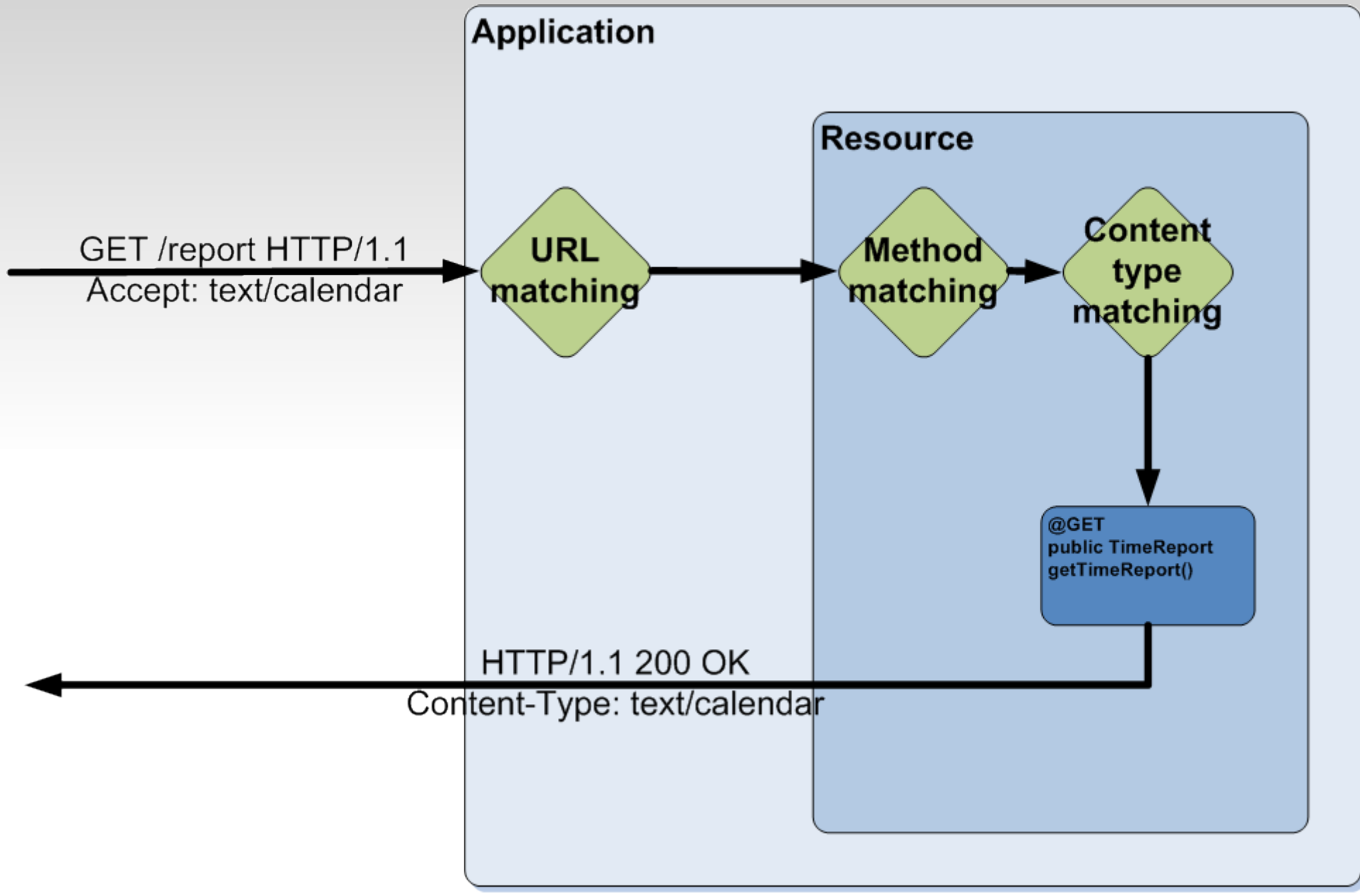
URL  
matching

Method  
matching

Content  
type  
matching

```
@GET  
public TimeReport  
getTimeReport()
```

HTTP/1.1 200 OK  
Content-Type: text/calendar



Any media type is allowed  
XML, JSON, text/plain, Binary files, ...

# Entity providers

MessageBodyReader, MessageBodyWriter

Standard Entity providers

# Application

## Resource

GET /niklas/report HTTP/1.1  
Accept: text/calendar

URL  
matching

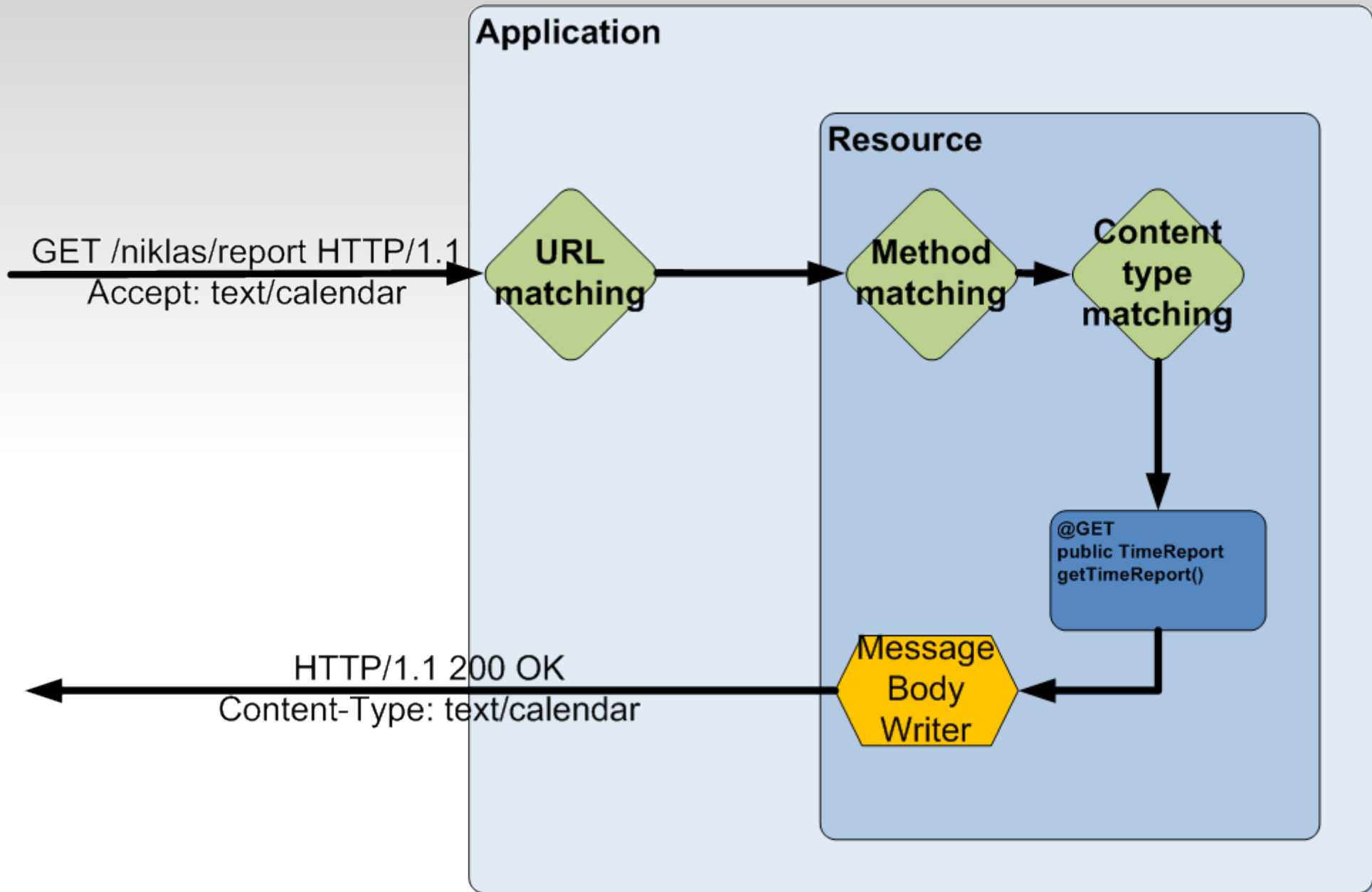
Method  
matching

Content  
type  
matching

```
@GET  
public TimeReport  
getTimeReport()
```

Message  
Body  
Writer

HTTP/1.1 200 OK  
Content-Type: text/calendar



Writing your own Entity provider

BEGIN:VFREEBUSY

DTSTART:20090324T080000Z

DTEND:20090324T170000Z

DTSTAMP:20090316T123136Z

UID:ca668b14ddf589ea8650b0b992eb64a656cdb7e@google.com

SUMMARY:SDC2009

END:VFREEBUSY

```
@Provider @Produces("text/calendar")
public class TimeReportICalWriter implements
    MessageBodyWriter<TimeReport> {

    public void writeTo(TimeReport t, Class<?> type, Type
        genericType, Annotation[] annotations, MediaType
        mediaType, MultivaluedMap<String, Object> httpHeaders,
        OutputStream entityStream) {

        PrintWriter wr = new PrintWriter(entityStream);
        ...

        for(TimeRange range : t.getRanges()) {
            wr.println("BEGIN:VFREEBUSY");
            wr.println("DTSTART:" + DF.format(range.getStartTime()));
            wr.println("DTEND:" + DF.format(range.getEndTime()));
            wr.println("SUMMARY:" + range.getDescription());
            wr.println("END:VFREEBUSY");
        }

        ...
    }
}
```

Response, ResponseBuilder

# Exception Mapping



# Deployment

# JAX-RS limitations



# JAX-RS limitations

## Lifecycle support

# JAX-RS limitations

Weak support for links, caching, method tunneling

# JAX-RS limitations

Accept based content negotiation only

<http://example.com/report>

<http://example.com/report.xml>

<http://example.com/report.cal>

<http://example.com/report.json>

**JAX-RS limitations**  
Limited security support

# JAX-RS limitations

## Leaky abstraction

Try it out!  
And have a look at the alternatives

# Want more?



Questions?

[niklas@protocol7.com](mailto:niklas@protocol7.com)

# Attributions

<http://www.flickr.com/photos/psd/421186578/>

<http://www.flickr.com/photos/sineout/2491569707/>

<http://www.flickr.com/photos/apelad/sets/72157594388426362/>

<http://www.flickr.com/photos/goopymart/289959670>