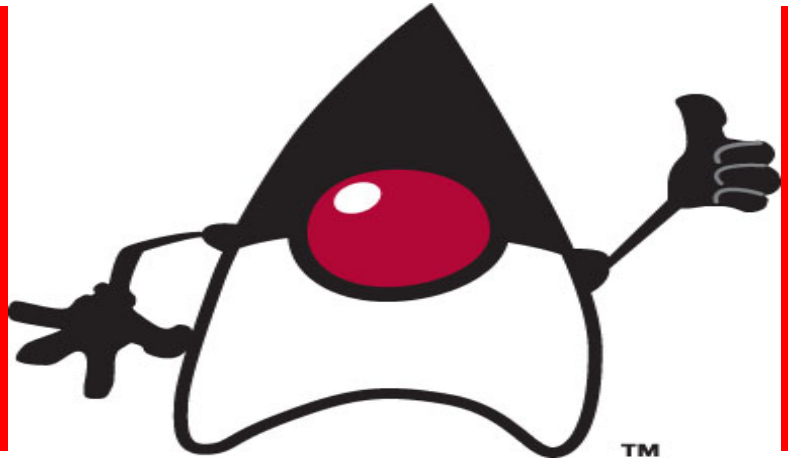


ORACLE®

RESTful Java with JAX-RS 2.0 and Jersey

Jakub Podlešák
Oracle

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Shortest REST Primer Ever

- RESTful Principles
  - Assign everything an ID
  - Link things together
  - Use common set of methods
  - Allow multiple representations
  - Stateless communications

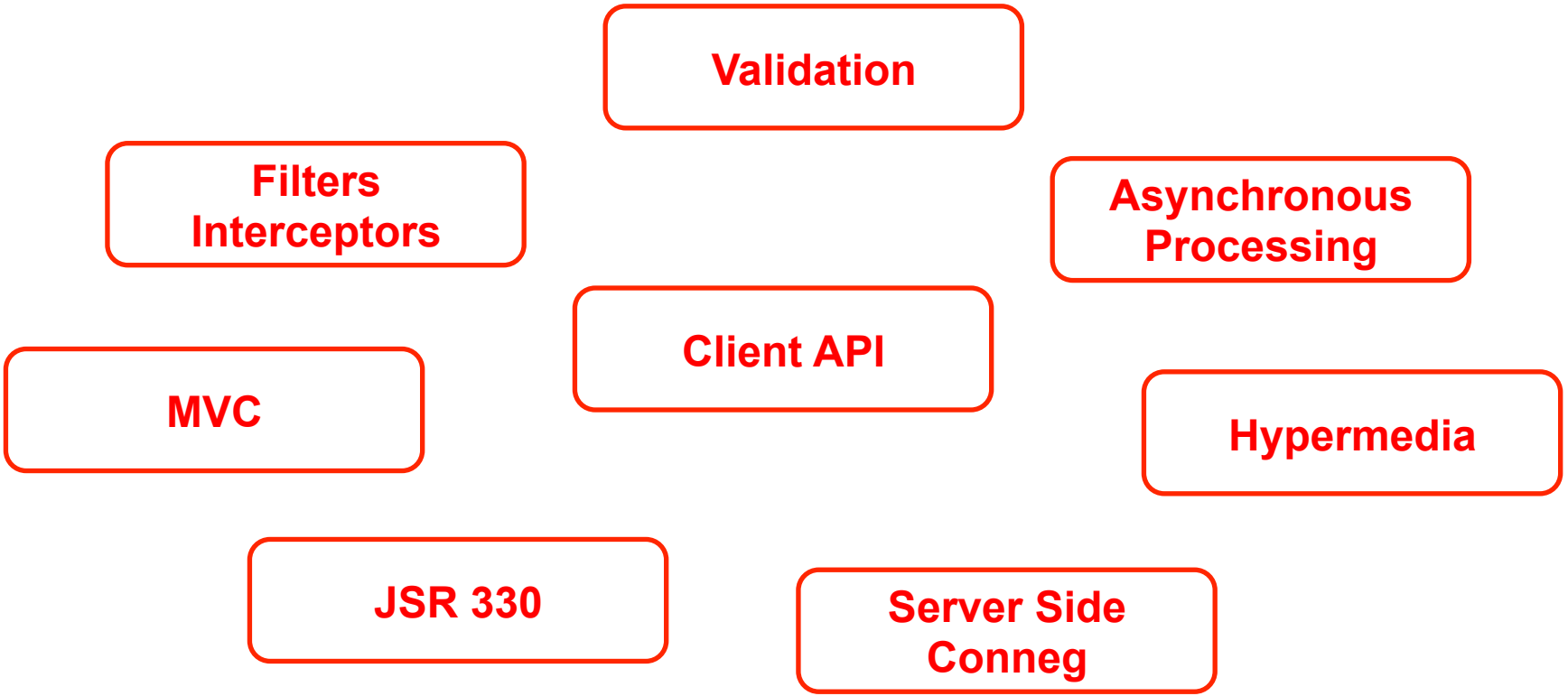
JAX-RS Introduction

- JAX-RS is Java API for RESTful Services
  - Current version: JAX-RS 1.1

- Goals
  - POJO-Based Resource API
    - Server-side
  - HTTP Centric
  - Entity Format Independence
  - Container Independence
  - Inclusion in Java EE
    - But no hard dependency on Java EE though

> JAX-RS 1.1 API

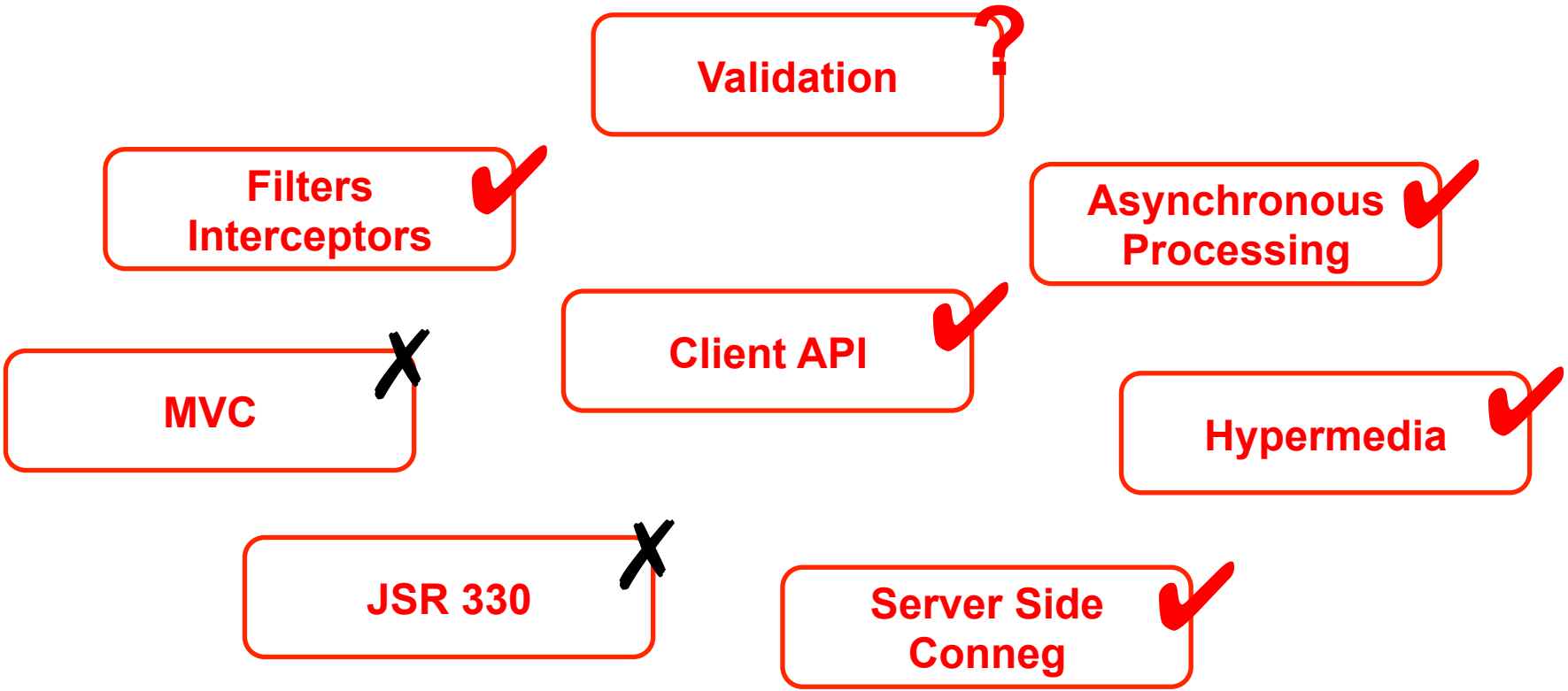
JAX-RS Most Wanted



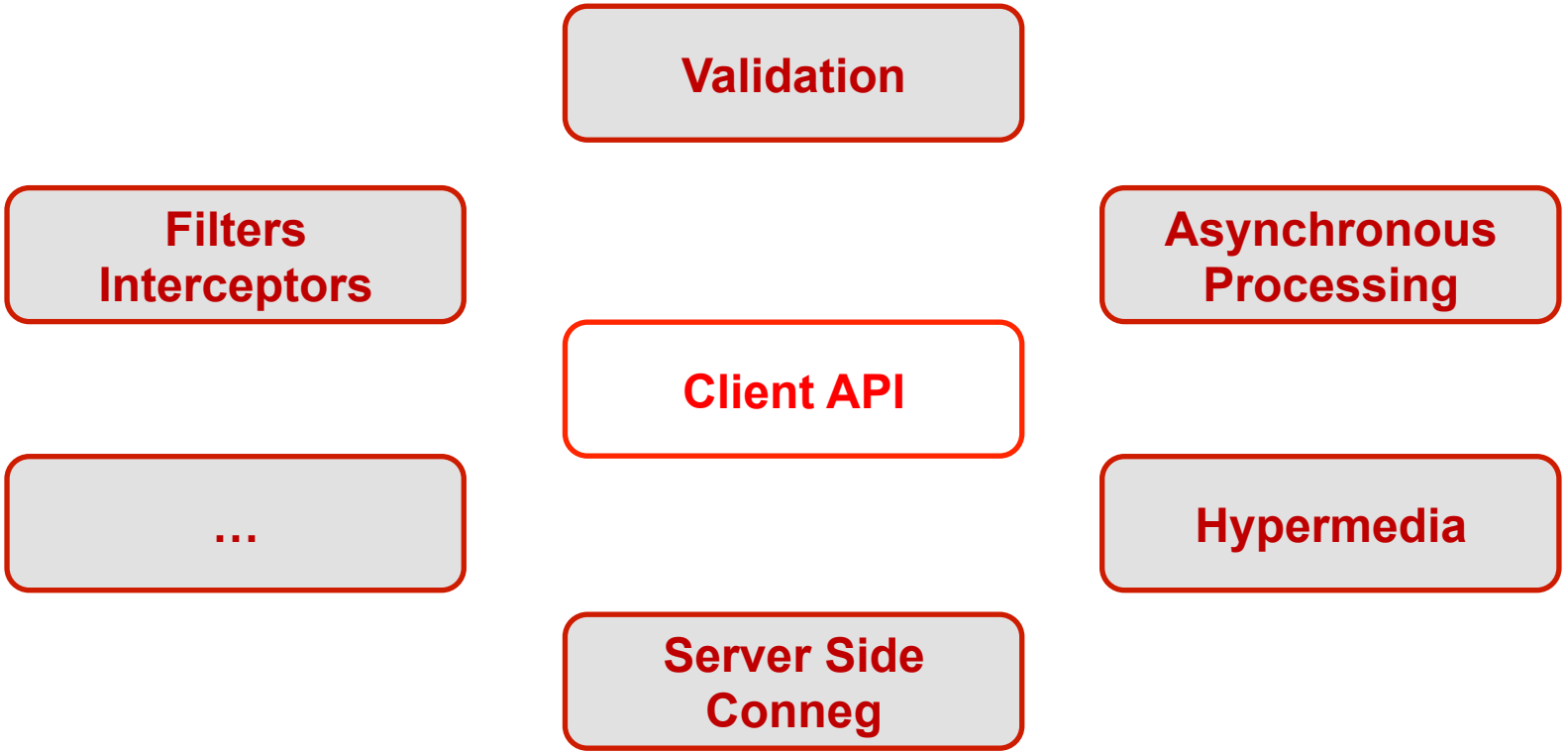
JSR 339: JAX-RS 2.0

- EG Formed in February 2011
  - Oracle Leads
    - Marek Potociar / Santiago Pericas-Geertsen
  - Expert Group:
    - Jan Algermissen, Florent Benoit (OW2), Sergey Beryozkin (Talend), Adam Bien, Bill Burke (RedHat), Clinton Combs, Bill De Hora, Markus Karg, Sastri Malladi (Ebay), Julian Reschke, Guilherme Silveira, Dionysios Synodinos
- Public Review ends on Nov 12th, 2012

New in JAX-RS 2.0



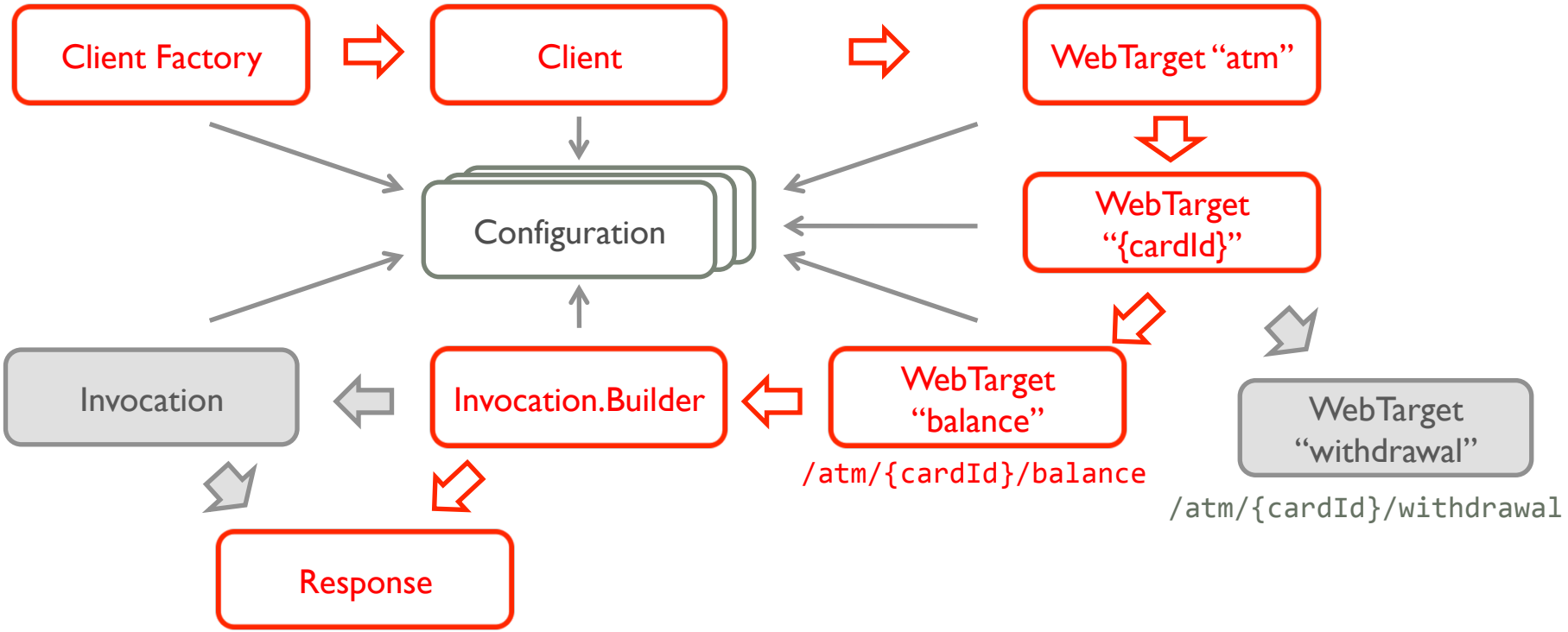
Client API


Validation
Filters
Interceptors
Asynchronous
Processing
Client API
…
Hypermedia
Server Side
Conneg

Client API

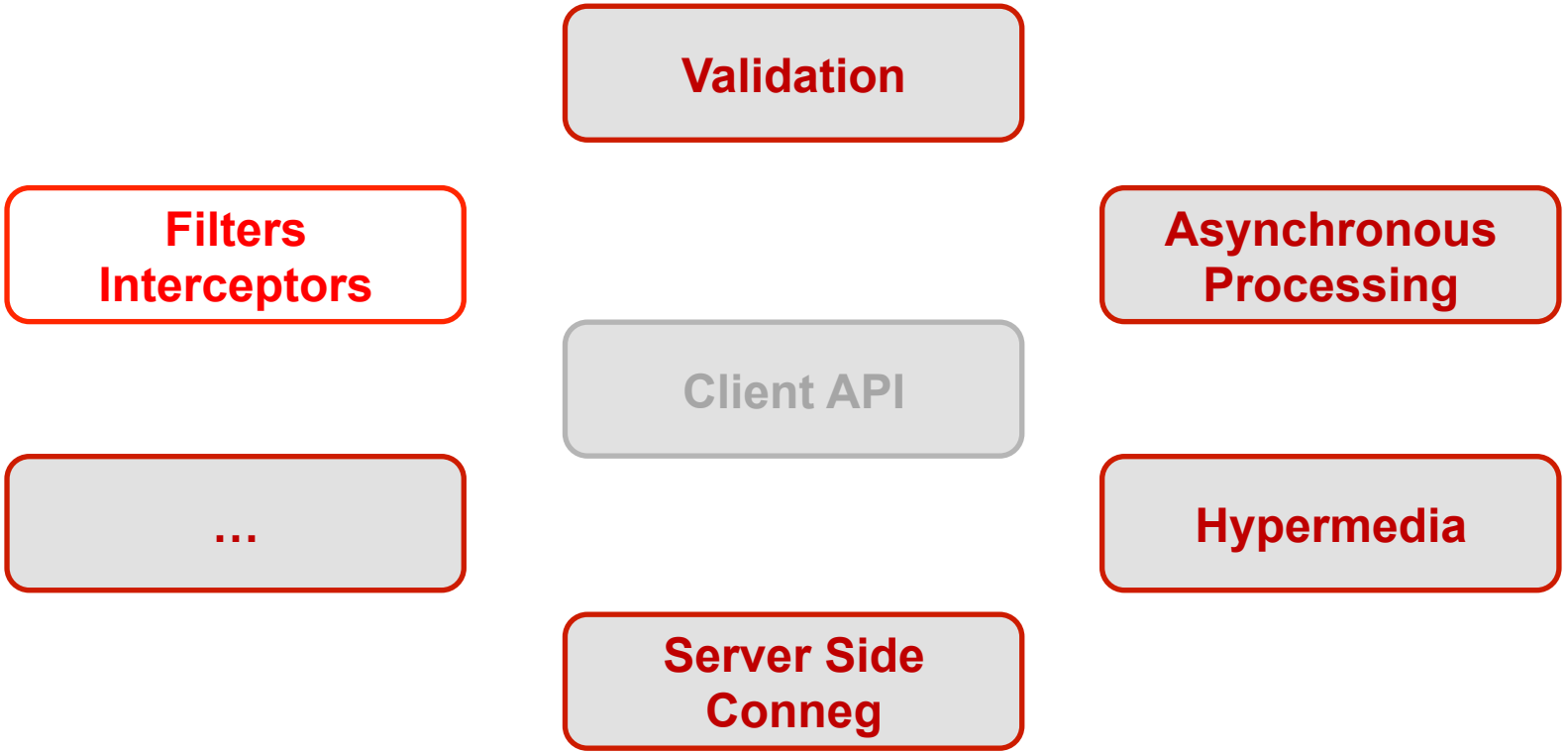
- HTTP client libraries too low level
- Sharing features with JAX-RS server API
  - E.g., MBRs and MBWs
- Supported by some JAX-RS 1.x implementations
  - Need for a standard

Client API


Client Factory
Client
WebTarget "atm"
Configuration
WebTarget "{cardId}"
Invocation
Invocation.Builder
WebTarget "balance"
/atm/{cardId}/balance
WebTarget "withdrawal"
/atm/{cardId}/withdrawal
Response

> JAX-RS 2.0 CLIENT API

Filters & Interceptors



Filters & Interceptors

- Customize JAX-RS request/response processing
  - Use Cases: Logging, Compression, Security, Etc.
- Introduced for client and server APIs
- Replace existing proprietary support
  - Provided by most JAX-RS 1.x implementations
    - All using slightly different types or semantics

Filters & Interceptors

- Invoked for each request or response
  - Non-wrapping filter chain managed by the JAX-RS runtime
  - Each filter decides to proceed or break the chain
- Request ➔ Request
  - `ContainerRequestFilter, ClientRequestFilter`
- Response ➔ Response
  - `ContainerResponseFilter, ClientResponseFilter`
- Server-side specialties
  - `@PostMatching, DynamicBinder`

Filters & Interceptors

- Intercept entity providers
  - Invoked ONLY when entity processing occurs
    - Performance boost
  - Wrapping interceptor chain
    - Each interceptor may decide to proceed or break the chain
      - By calling `context.proceed()`
- `MessageBodyReader` interceptor
  - `ReaderInterceptor` interface
- `MessageBodyWriter` interceptor
  - `WriterInterceptor` interface

Filters & Interceptors Bindings & Priorities

- Binding
  - Associating filters and interceptors with resource methods
  - Server-side concept
- Priority
  - Declaring relative position in the execution chain
  - `@BindingPriority(int priority)`
- Shared concept by filters and interceptors

	Scoped Binding	Global Binding
Static	`@NameBinding` (`@Qualifier`?)	*Default* `@PostMatching`
Dynamic	`DynamicBinder` interface	N/A

Filters & Interceptors Bindings

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

@Provider
@Logged
@BindingPriority(USER)
public class LoggingFilter
    implements ContainerRequestFilter, ContainerResponseFilter { … }
```

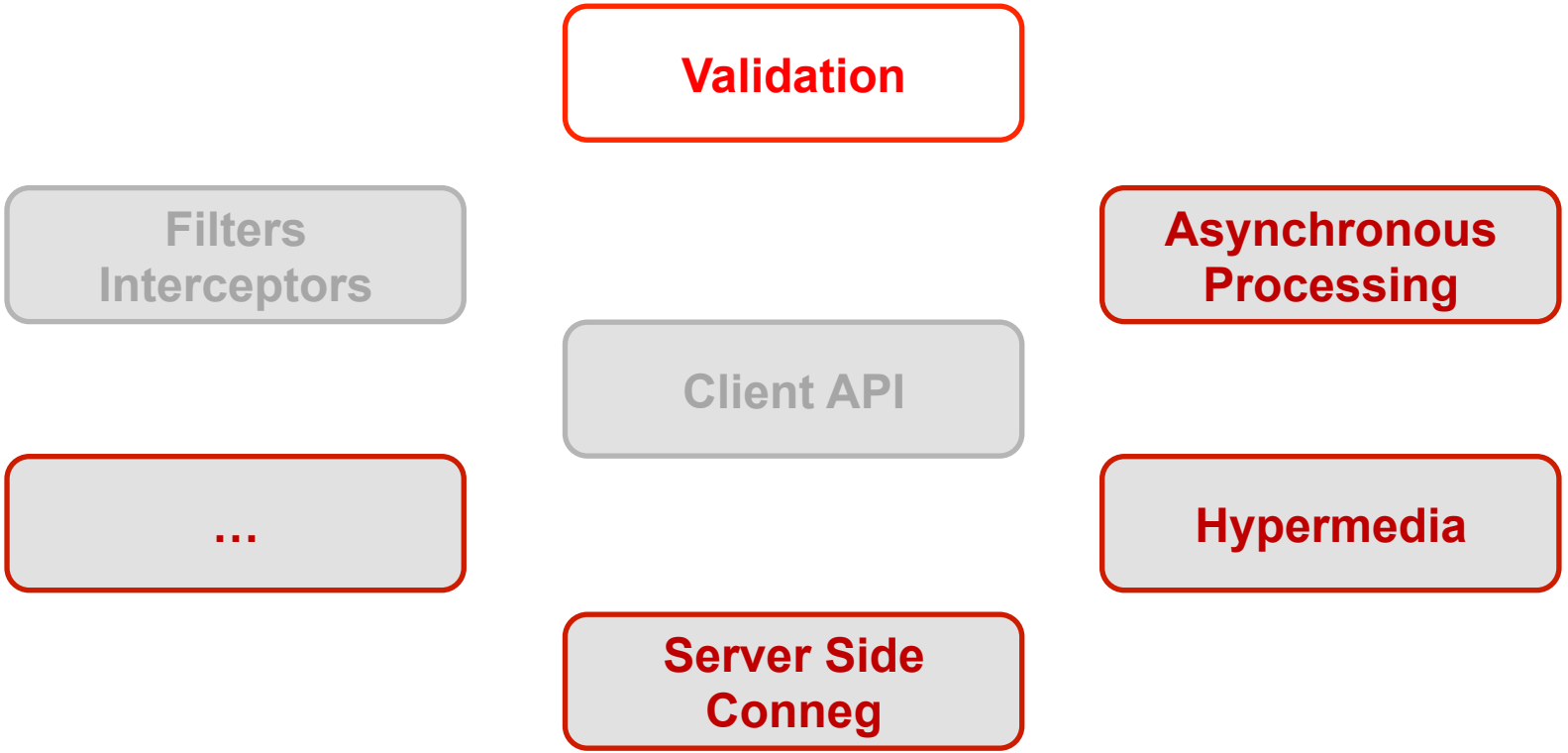
Filters & Interceptors Bindings

```
@Path("/greet/{name}")
@Produces("text/plain")
public class MyResourceClass {

    @Logged
    @GET
    public String hello(@PathParam("name") String name) {
        return "Hello " + name;
    }
}
```

> JAX-RS 2.0 FILTERS

Validation


Validation
Filters
Interceptors
Asynchronous
Processing
Client API
…
Hypermedia
Server Side
Conneg

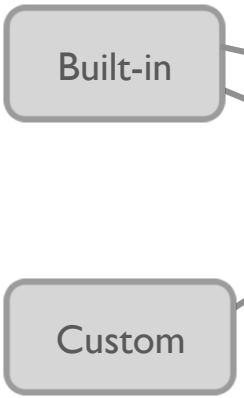
Validation

- Services must validate data
- Bean Validation already provides the mechanism
  - Integration into JAX-RS
- Support for constraint annotations in:
  - Fields and properties
  - Parameters (including request entity)
  - Methods (response entities)
  - Resource classes

Validation

```
@Path("/")
class MyResourceClass {

        @POST
        @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
        public void registerUser(
            @NotNull @FormParam("firstName") String firstName,
            @NotNull @FormParam("lastName") String lastName,
            @Email @FormParam("email") String email) {
            ... }
```



Validation

```
@Target({ METHOD, FIELD, PARAMETER })
@Retention(RUNTIME)
@Constraint(validatedBy = EmailValidator.class)
public @interface Email { ... }

class EmailValidator implements ConstraintValidator<Email, String> {
    public void initialize(Email email) { … }
    public boolean isValid(String value,
        ConstraintValidatorContext context) {
        // Check 'value' is e-mail address
        …
    }
}
```

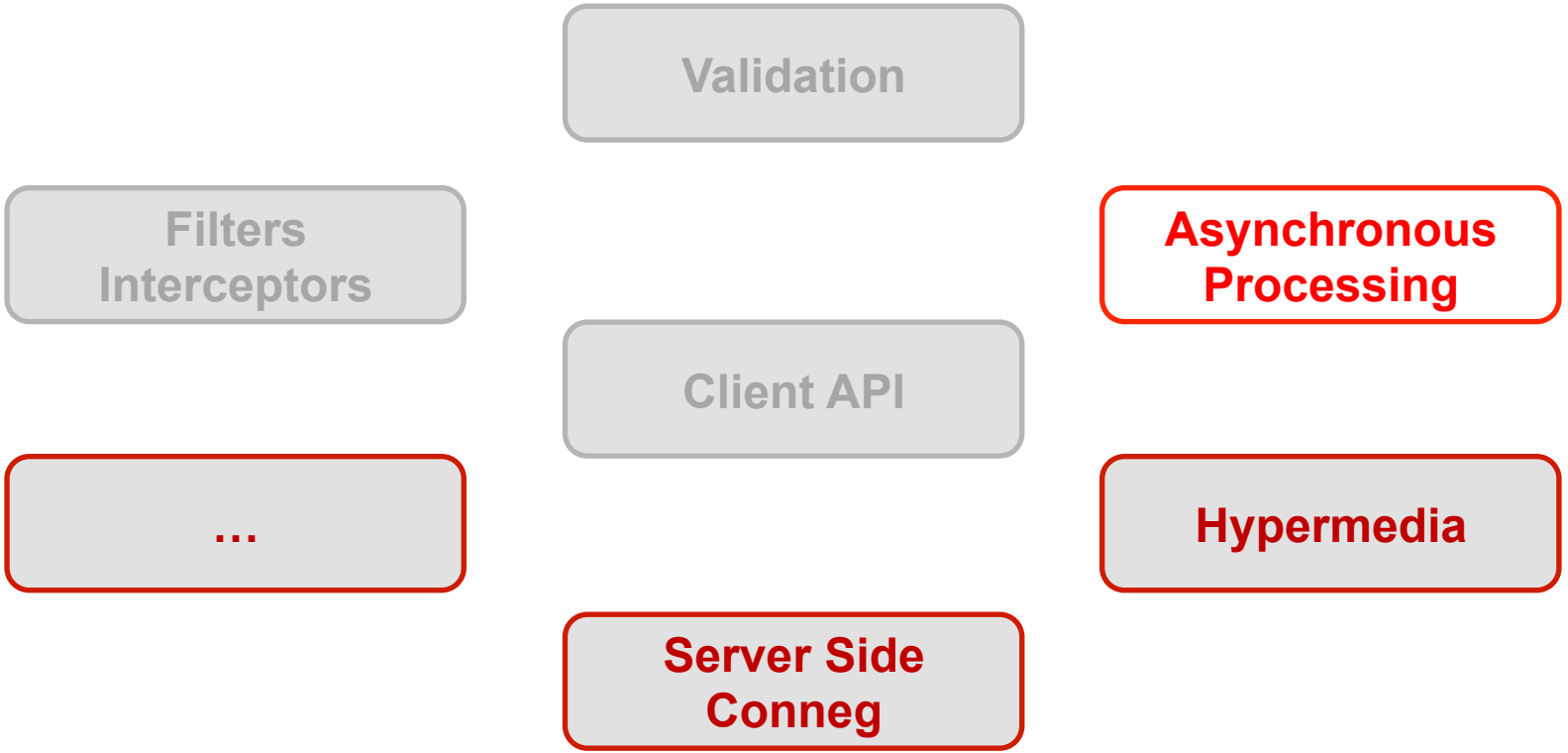
Validation

```
@ValidationA
class User { ... }

@Path("/")
class MyResourceClass {
    @POST
    @Consumes("application/xml")
    public void registerUser1(@Valid User u) { … }

    @POST
    @Consumes("application/json")
    public void registerUser12(@ValidationB @Valid User u) { … }
}
```

Async Processing



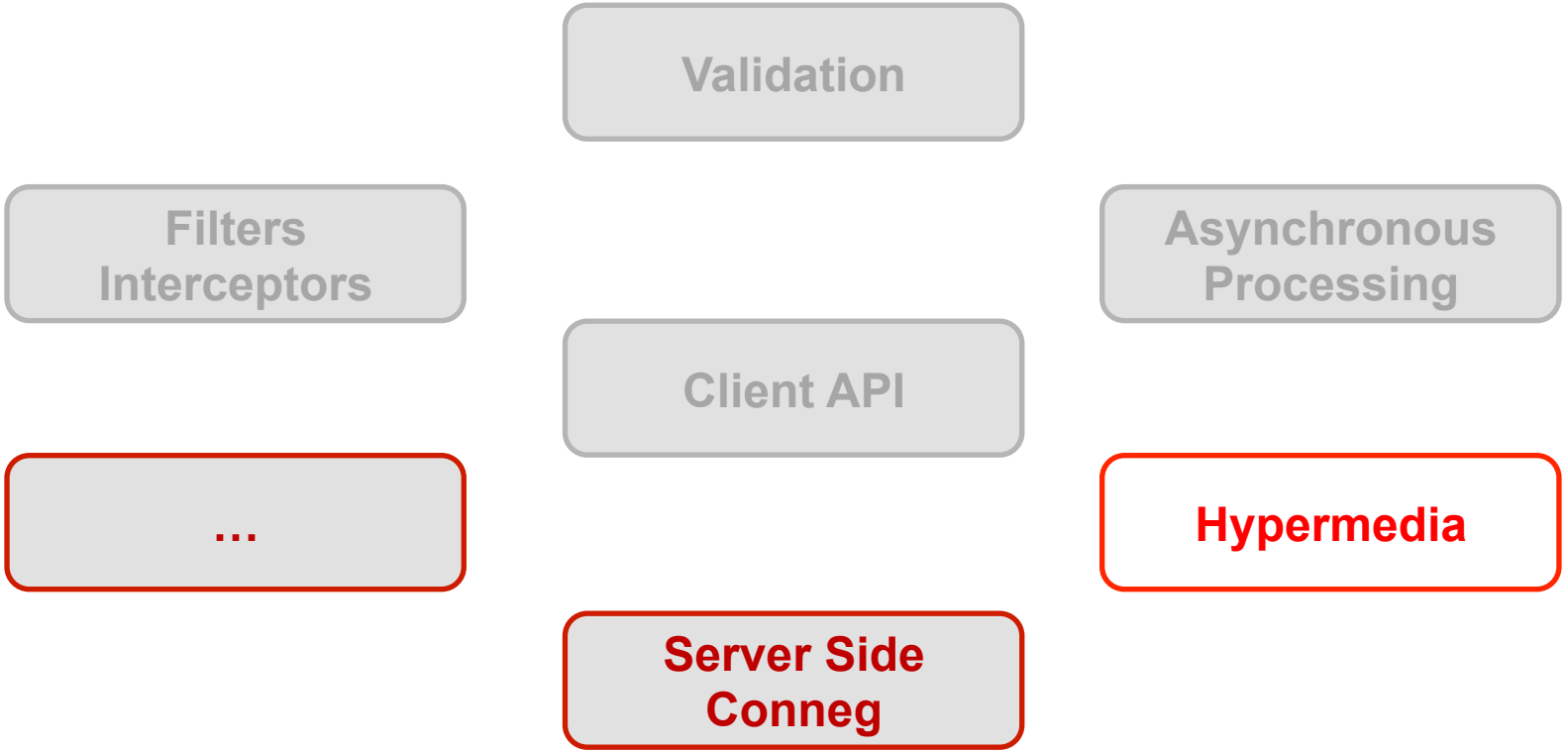
Async Processing

- Server API support
  - Off-load container threads
    - Long-running operations
  - Efficient asynchronous event processing
    - Suspend while waiting for an event
    - Resume when event arrives
  - Leverage Servlet 3.x async support (if available)

- Client API support
  - Asynchronous request invocation API
    - `Future<`*`RESPONSE`*`>`, `InvocationCallback<`*`RESPONSE`*`>`

> JAX-RS 2.0 SERVER ASYNC

Hypermedia

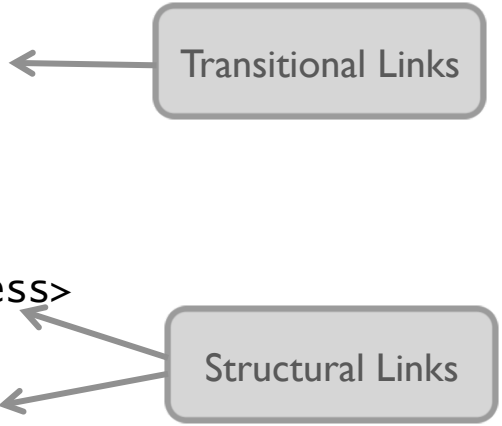

Validation
Filters
Interceptors
Asynchronous
Processing
Client API
…
Hypermedia
Server Side
Conneg

Hypermedia

- REST principles
  - Identifiers and Links
  - HATEOAS (Hypermedia As The Engine Of App State)
- Link types:
  - Structural Links
  - Transitional Links

Hypermedia

```
Link: <http://.../orders/1/ship>; rel=ship,
      <http://.../orders/1/cancel>; rel=cancel
      ...
<order id="1">
    <customer>http://.../customers/11</customer>
    <address>http://.../customers/11/address/1</address>
    <items>
        <item>
            <product>http://.../products/111</product>
            <quantity>2</quantity>
        </item>
    <items>
    ...
</order>
```



Hypermedia

- `Link` and `LinkBuilder` classes
  - RFC 5988: Web Linking
- Support for `Link` in `ResponseBuilder` and filters
  - Transitional links (headers)
- Support for manual structural links
  - Via `Link.JaxbAdapter` & `Link.JaxbLink`
- Create a resource target from a Link in Client API

Hypermedia

```
// Producer API (server-side)
Link self = Link.fromResourceMethod(MyResource.class, ”handleGet”)
    .build();
Link update = Link
    .fromResourceMethod(MyResource.class, “handlePost”)
    .rel(”update”).build();

...

Response res = Response.ok(order)
    .link("http://.../orders/1/ship", "ship")
    .links(self, update)
    .build();
```

Hypermedia

```
Response order =
    client.target(…).request("application/xml").get();

// Consumer API (client-side)
if (order.getLink(“ship”) != null) {
    Response shippedOrder = client.target(order.getLink("ship"))
        .request("application/xml").post(…);
    …
}
```

Other Topics

Validation

Filters
Interceptors

Asynchronous
Processing

Client API

Hypermedia

…

Server Side
Conneg

Server Side Conneg

```
GET http://.../widgets2
Accept: text/*; q=1
…
```

```
Path("widgets2")
public class WidgetsResource2 {
    @GET
    @Produces("text/plain", "text/html")
    public Widgets getWidget() {...}
}
```

Server Side Conneg

```
GET http://.../widgets2
Accept: text/*; q=1
…
```

```
Path("widgets2")
public class WidgetsResource2 {
    @GET
    @Produces("text/plain; qs=0.5", "text/html; qs=0.75")
    public Widgets getWidget() {...}
}
```

ORACLE KRAKOW IS HIRING

- Who do we need:
  - Java developers
  - .NET developers
  - QA
- Oracle Office address: ul.Bora-Komorowskiego 25B
- See our booth for more details!

Thank You!

Links

http://jcp.org/en/jsr/detail?id=339 (JSR @ jcp.org)
http://java.net/projects/jax-rs-spec (JAX-RS Project)
http://jersey.java.net/ (Jersey Project)

Mailing lists

users@jax-rs-spec.java.net (JAX-RS users alias)
users@jersey.java.net (Jersey users alias)

JSR-330 Integration

- Support Java Dependency Injection API (JSR-330)
  - Support `@Inject` and `@Qualifier` ?
  - `@Qualifier` as a replacement for `@NamedBinding` ?
  - `Provider` vs. `ContextResolver` ?
  - Support DI (JSR-330) or CDI (JSR-299)?
- Issues
  - Interference with CDI providers
  - EG does not see enough added value for DI
- DI-style injection support deferred

Other Topics

- Improved Java EE security support?
  - `@RolesAllowed`, …
- Pluggable Views?
  - Completes the MVC pattern
- High-level client API ?
  - Hard to design it to be RESTful
  - Jersey 2 provides an experimental support
  - Don’t want to end-up with an RPC-style API