

2.– 5. September 2013
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Ein Blick ins Aquarium

Was gibt's Neues in Bean Validation 1.1 and CDI 1.1?

Hardy Ferentschik

Red Hat

About me

- Hardy Ferentschik - hardy@hibernate.org
- Hibernate team member for 5 years
- Focus on Validator and Search
- Validator project lead
- JSR 349 Expert Group member
- +10 years experience in software development
- iDad



A Java EE Smörgåsbord



Agenda



- Java EE overview
- Bean Validation 1.1
- CDI 1.1
- Tips and tricks

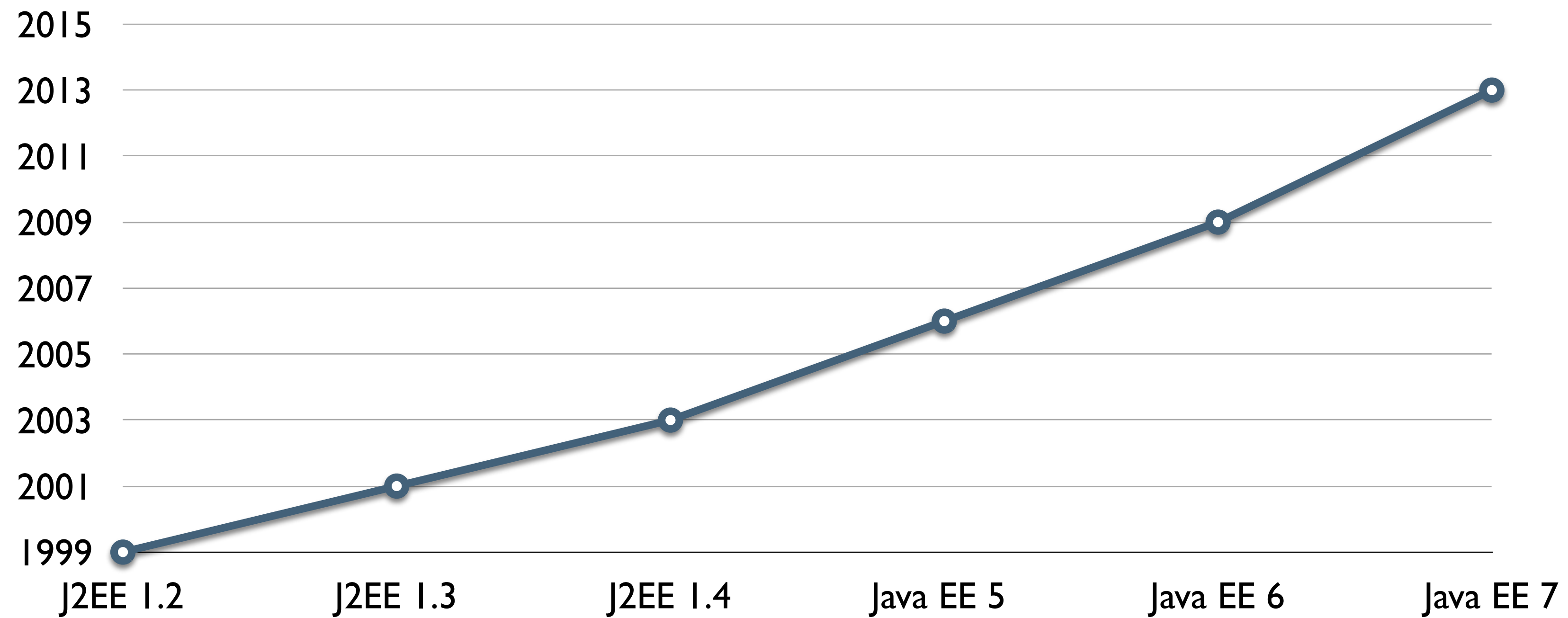
Java EE

“Java Platform, Enterprise Edition or Java EE is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications.”



WIKIPEDIA

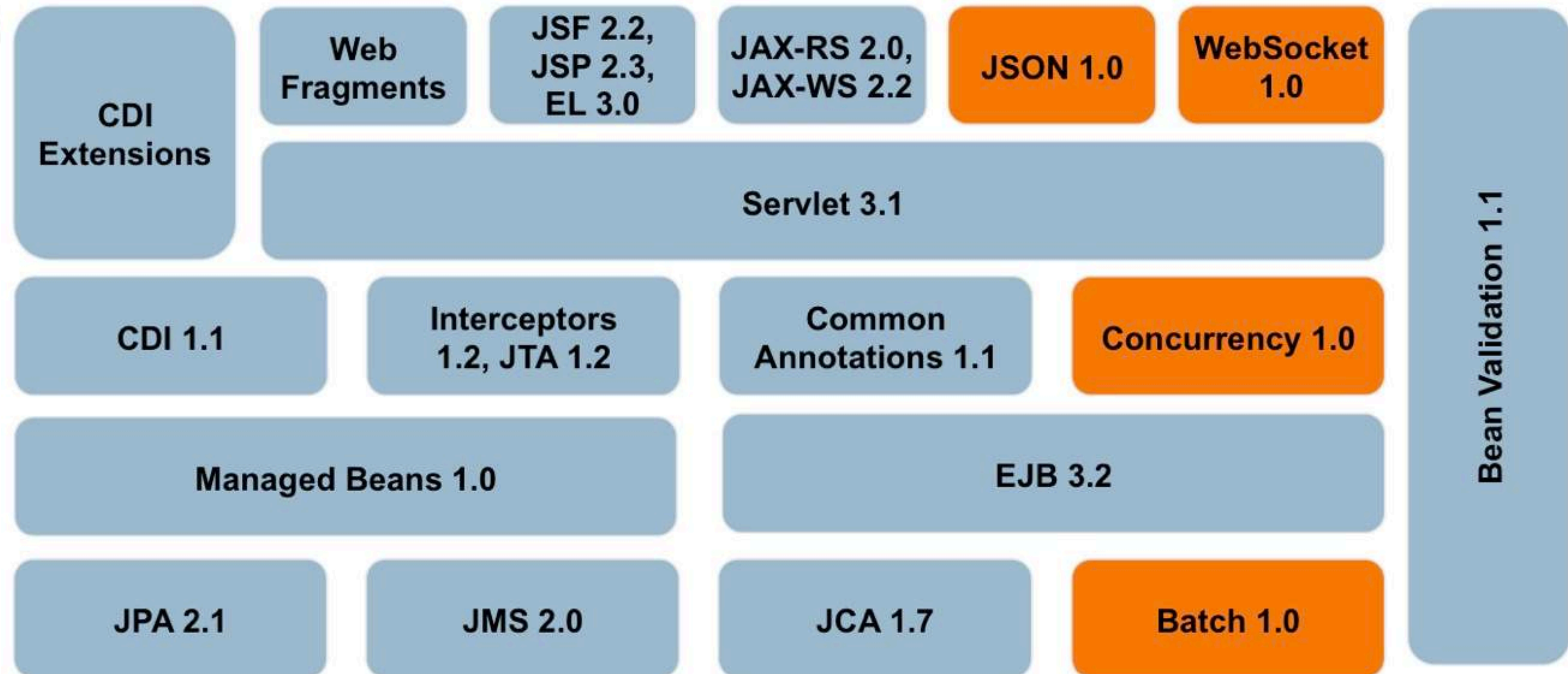
○ J2EE & Java EE Releases



Java EE 7



https://blogs.oracle.com/arungupta/entry/java_ee_7_sdk_and



https://blogs.oracle.com/arungupta/entry/java_ee_7_sdk_and

Bean Validation 1.1



BV 1.0 recap

BV 1.0 recap

@Null

@Pattern

@NotNull

@Size

@AssertTrue

@Future

@AssertFalse

@Past

@Min

@DecimalMax

@Max

@Digits

@DecimalMin

- Centered around domain model
- A uniform way to express validation rules
- A standard way to check these rules

- Constraints on bean, field and property level
- Type-safe and generic validation API
- Powerful meta data API

Bean Validation: Bean Validation

home | news | road map | contribute | forum | bug tracker | licensing

Bean Validation

Welcome to the Bean Validation specification website.

Get Bean Validation 1.1

What is Bean Validation

Bean Validation is a Java specification which

- lets you express constraints on object models via annotations
- lets you write custom constraints in an extensible way
- provides the APIs to validate objects and object graphs
- provides the APIs to validate parameters and return values of methods and constructors
- reports the set of violations (localized)
- runs in Java SE but is integrated in Java EE (6 and 7)

```
public class User {
    @NotNull @Email
    public String getEmail() { return email; }
    public void setEmail(String email) {
        this.email = email;
    }
    private String email;
}

public class UserService {
    public void createUser(@Email email,
                          @NotNull String name) {
    }
}
```

While you can run validation manually, it is more natural to let other specifications and framework validate data at the right time (user input in presentation frameworks, business service execution by CDI, entity insert or update by JPA).

In other words, *run once, constrain anywhere*.

[Learn more and get the specification...](#)

Latest news

Stay up to date, subscribe to the [news feed](#).

XML namespace and JCP

04 June 2013

Antonio Goncalves, fellow JCP member and friend has asked me why Bean Validation XML's namespace has not moved from its original location to the jcp.org location like other Java EE 7 specifications. I don't remember being aware that such a move was orchestrated so there are two possible reasons: I was never been made aware of...

[Read more](#)

Bean Validation 1.1 is a spec

02 May 2013

It's now official, these couple of years of work have made it into an official JCP specification. Bean Validation is also part of Java EE 7 which has been approved too a few of days ago. We have already discussed the features at great length here but to do a short summary: support for method and constructor...

[Read more](#)

Bean Validation 1.1 CR3 - Final Approval Ballot

21 March 2013

Bean Validation, Hibernate Validator (its Reference Implementation) and the Test Compatibility Kit have been handed over to the JCP for what is called the Final Approval Ballot. That's when the expert committee votes for the go / no-go of the specification going final as it is. We have found a few glitches when working...

[Read more](#)

beanvalidation (Bean Validation) · GitHub

GitHub, Inc.

github.com/beanvalidation/


Reader

GitHub

Search or type a command

ExploreFeaturesEnterpriseBlog

Sign upSign in



Bean Validation

beanvalidation

<http://beanvalidation.org>

Joined on Sep 29, 2010

4

public repos

3


members

Repositories

Members

Find a Repository...

AllSourcesForksMirrors



beanvalidation.org


Bean Validation website

Last updated a month ago

Ruby

★ 5

🔗 3



beanvalidation-spec


Bean Validation specification

Last updated a month ago

XSLT

★ 8

🔗 7



beanvalidation-tck


Bean Validation TCK

Last updated 3 months ago

Java

★ 10

🔗 5



beanvalidation-api

Bean Validation API

Last updated 3 months ago


Java

★ 13

🔗 9

© 2013 GitHub, Inc.

TermsPrivacySecurityContact



StatusAPITrainingShopBlogAbout

Tuesday, September 3, 13

What's new in 1.1



- EL expressions in message interpolation
- Group conversion
- Improved CDI integration
- Method and constructor validation

EL in error messages via JSR 341

```
must be greater than  
    ${inclusive == true ? 'or equal to ' : ''}{value}
```

```
${formatter.format('Max %s, min %s', max, min)}
```

```
${validatedValue.age}
```

```
${formatter.format('%1$.2f', validatedValue)}
```


Validation Groups

```
public class User {
    @NotNull
    private String firstname;

    @NotNull(groups = Default.class)
    private String lastname;

    @Pattern(regexp = "[0-9 -]?", groups = Optional.class)
    private String phoneNumber;

    @NotNull(groups = { Billable.class, BuyInOneClick.class })
    private CreditCard defaultCreditCard;

    // ...

    public interface BuyInOneClick extends Default, Billable {
    }

    public interface Billable {
    }

    public interface Optional {
    }
}
```

Group conversion

```
public class RentalCar {  
  
    @NotNull  
    private String licensePlate;  
  
    @Valid  
    @ConvertGroup(  
        from = Default.class,  
        to = DriverChecks.class  
    )  
    private Driver driver;  
}
```


Group conversion

```
public class RentalCar {  
  
    @NotNull  
    private String licensePlate;  
  
    @Valid  
    @ConvertGroup(  
        from = Default.class,  
        to = DriverChecks.class  
    )  
    private Driver driver;  
}
```

```
public class Driver {  
  
    @NotNull  
    private String name;  
  
    @AssertTrue(  
        groups = DriverChecks.class  
    )  
    private boolean hasLicense;  
    ...  
}
```

CDI integration

Dependency injection for:

- ConstraintValidator instances
- MessageInterpolator
- TraversableResolver
- ConstraintValidatorFactory
- ParameterNameProvider

Method Validation

Design by contract

Contract Programming

Programming by contract

Design-by-contract programming

Method Validation

```
@NotNull  
@Valid  
public Order placeOrder(  
    @NotNull @Size(min=3, max=20) String customerCode,  
    @NotNull @Valid Item item) {  
    // ...  
}
```


Cross Parameter Constraints

```
@PasswordMatch  
public void setPassword(  
    @NotNull String password,  
    @NotNull String passwordConfirmation) {  
    // ...  
}
```

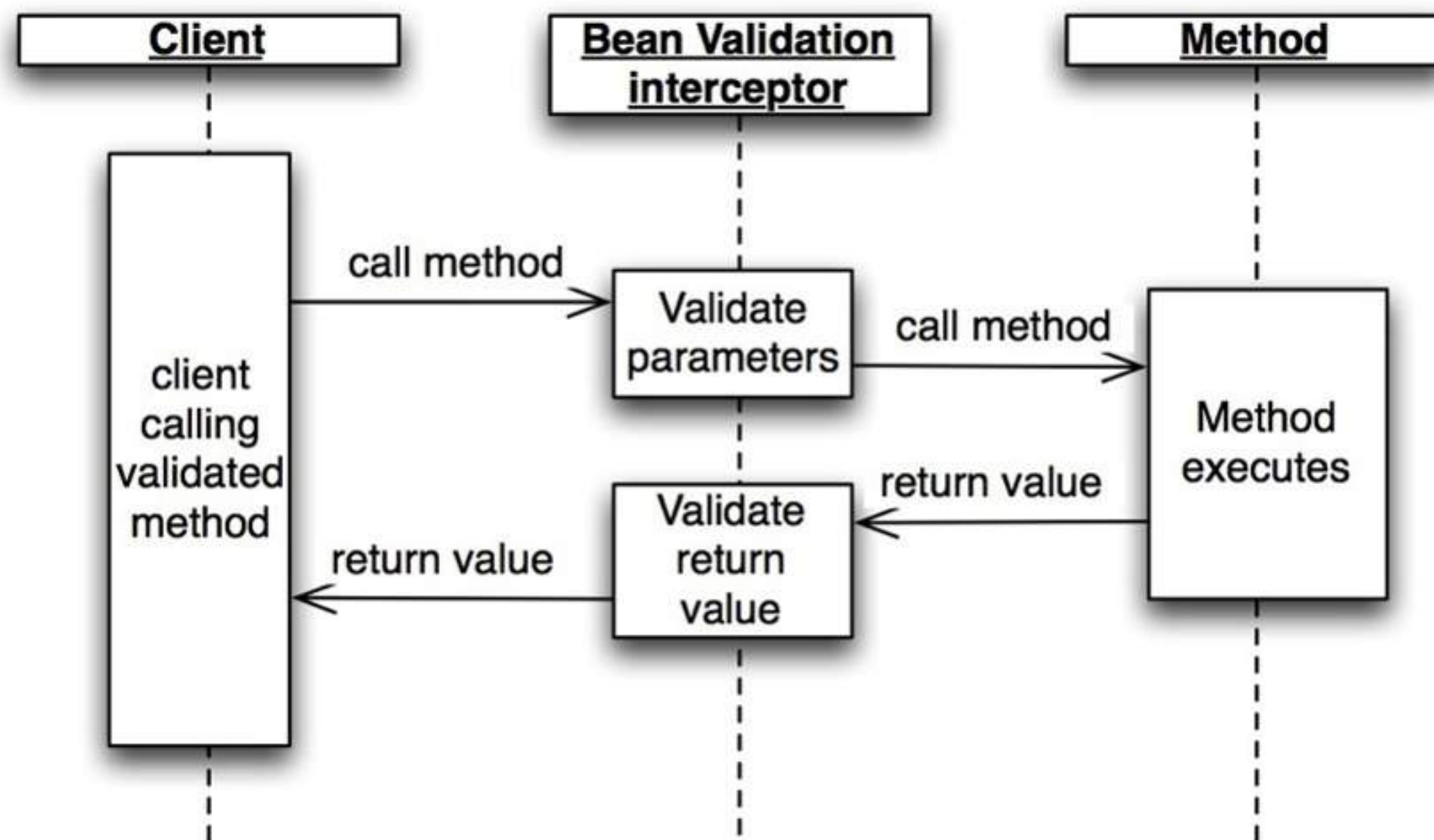
```
@SupportedValidationTarget(ValidationTarget.PARAMETERS)
public class PasswordMatchValidator
    implements ConstraintValidator<PasswordsMatch, Object[]> {

    @Override
    public void initialize(PasswordsMatch constraintAnnotation) {
        // ...
    }

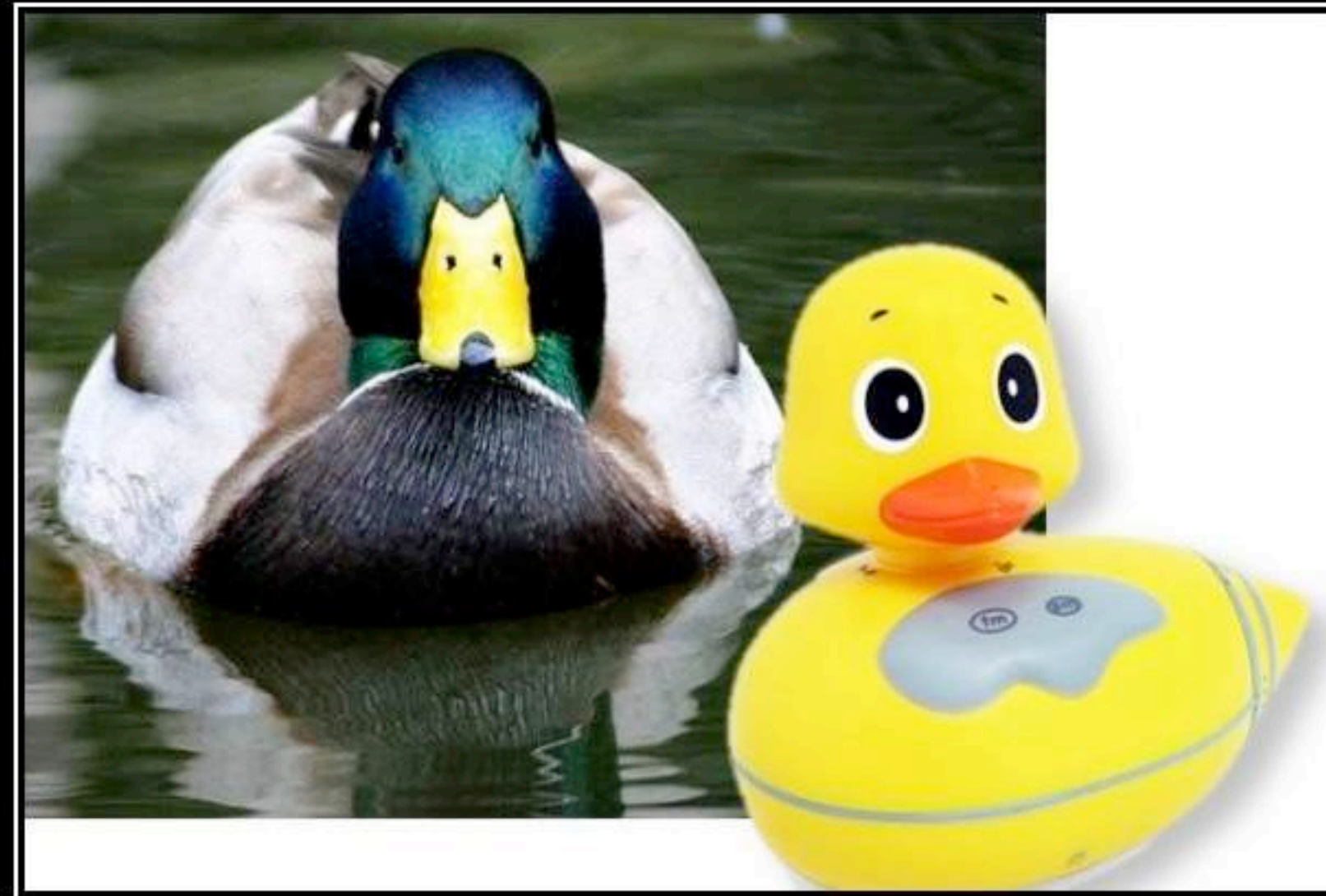
    @Override
    public boolean isValid(Object[] value,
                          ConstraintValidatorContext context) {
        // ...
    }
}
```


ExecutableValidator

- Access via `Validator.forExecutables()`
- Offers:
 - `validateParameters`
 - `validateReturnValue`
 - `validateConstructorParameters`
- For example:
 - `validateParameters(T object, Method m, Object[] parameters, Class<?>... groups);`



Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.



LISKOV SUBSTITUTION PRINCIPLE

If It Looks Like A Duck, Quacks Like A Duck, But Needs Batteries - You
Probably Have The Wrong Abstraction

LSP and BV

When defining method constraints within inheritance hierarchies (that is, class inheritance by extending base classes and interface inheritance by implementing or extending interfaces) one has to obey the LSP which mandates that:

- a method's preconditions (as represented by parameter constraints) must not be strengthened in sub types
- a method's postconditions (as represented by return value constraints) must not be weakened in sub types

LSP example

```
public interface Foo {  
    public void doStuff(@NotNull String s);  
}  
  
public class Fubar implements Foo {  
    public void doStuff(@Size(min=3) String s) {}  
}
```

LSP example

```
public interface Foo {  
    public void doStuff(@NotNull String s);  
}  
  
public class Fubar implements Foo {  
    public void doStuff(@Size(min=3) String s) {}  
}
```



LSP example

```
public interface Foo {  
    public void doStuff(@NotNull String s);  
}
```

```
public class Fubar implements Foo {  
    public void doStuff(@Size(min=3) String s) {}  
}
```

```
public interface Foo {  
    public void doStuff(@NotNull @Size(min=3) String s);  
}
```

```
public class Fubar implements Foo {  
    public void doStuff(String s) {}  
}
```


A word cloud centered on the text "CDI". The words are arranged in a circular pattern around the central "CDI". The words include: "Inject", "Dependency", "standard", "context", "Produce", "Decorator", "Alternative", "Enterprise", "portable", "extensible", "Qualifier", "Type-safe", "Java", and "Interceptor". The words are in various colors (red, orange, yellow, green, blue) and sizes, with "CDI" being the largest and most prominent.

Interceptor
Inject
Dependency
CDI
standard
context
Produce
Decorator
Alternative
Enterprise
portable
extensible
Qualifier
Type-safe
Java

Contexts and Dependency Injection

www.cdi-spec.org

Reader

Home

Download

FAQ

News

Major issues

Mailing List

Bug tracker

Contribute

Interceptor

Inject

Dependency

standard

context

Produce

Decorator

Alternative

portable

Enterprise

extensible

Qualifier

Type-safe

Java

CDI

Highlights

JCP pages

Servers

Overview

What is CDI?

Contexts and Dependency Injection for Java EE (CDI) 1.0 was introduced as part of the Java EE 6 platform, and has quickly become one of the most important and popular components of the platform.

CDI defines a powerful set of complementary services that help improve the structure of application code.

- A well-defined lifecycle for stateful objects bound to lifecycle contexts, where the set of contexts is extensible
- A sophisticated, typesafe dependency injection mechanism, including the ability to select dependencies at either development or deployment time, without verbose configuration
- Support for Java EE modularity and the Java EE component architecture?the modular structure of a Java EE application is taken into account when resolving dependencies between Java EE components
- Integration with the Unified Expression Language (EL), allowing any contextual object to be used directly within a JSF or JSP page
- The ability to decorate injected objects
- The ability to associate interceptors to objects via typesafe interceptor bindings
- An event notification model
- A web conversation context in addition to the three standard web contexts defined by the Java Servlets specification
- An SPI allowing portable extensions to integrate cleanly with the container

Latest Release

The latest release of CDI is 1.1. You can [download](#) the spec or [browse the javadoc](#).

Tuesday, September 3, 13



- Aligning the JSF component model/scoping to CDI
- JTA 1.2 @Transactional CDI interceptor
- Modernizing the JMS 2 API utilizing CDI
- Better support for CDI in Bean Validation 1.1



- Programmatic bean disablement via `@Vetoed`
- `@WithAnnotations` as part of `ProcessAnnotatedType`

JSR 318 - Interceptors

- Common base of EJB and CDI interceptors
- `@Priority` for interceptor ordering
- `@AroundConstruct` for constructor interception

Finally some code!



PortableExtension

- Dynamically adding interceptor binding annotation
- Dynamically creating beans
- Dynamically registering interceptors

Links

- http://www.youtube.com/watch?v=oooumQO8b3_8
- http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- https://blogs.oracle.com/theaquarium/entry/fifteen_javaee_7_apis_featured
- <http://beanvalidation.org/news/2013/02/20/bean-validation-1-1-proposed-final-draft/>
- <http://jcp.org/en/procedures/overview>
- http://en.wikipedia.org/wiki/Design_by_contract
- <http://stackoverflow.com/questions/56860/what-is-the-liskov-substitution-principle>
- <http://in.relation.to/Bloggers/CDI11Available>

Q + A