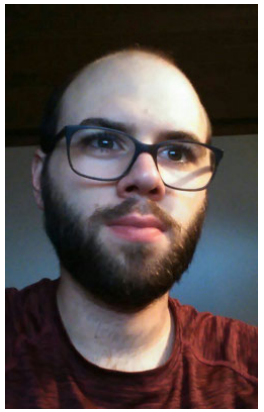
Softwareentwicklung ist kein Hexenwerk

markus.hettich@doubleslash.de

doubleSlash



doubleSlash

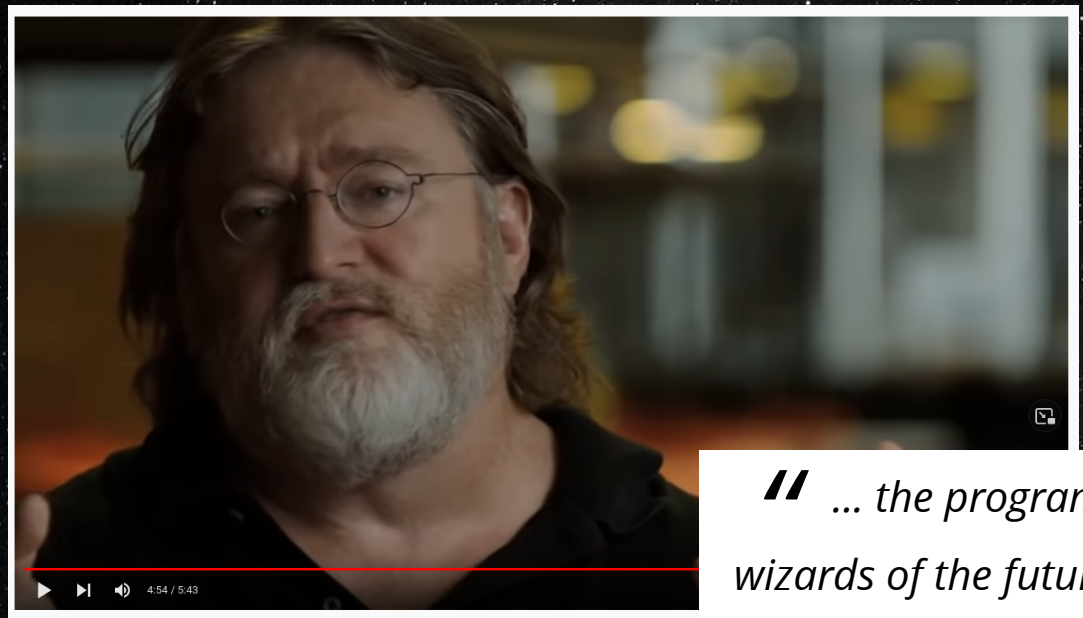
Reactive Programming

Functional Programming

Object Oriented Programming

Data Oriented Programming

markus.hettich@doubleslash.de



> *... the programmers of tomorrow are the wizards of the future you know you're going to look like you have magic powers compared to everybody else ...*

Quelle: https://www.youtube.com/watch?v=nKIu9yen5nc

Was ist Magie ?

die Faszination die von etwas bestimtem ausgeht **- Wiktionary**

vemeintliche Einflussnahme auf ..., Dinge und Ereignise auf übernatürliche Art und Weise **- Wiktionary**

geheime Kunst, die sich übersinnliche Kräfte dienstbar zu machen sucht; Zauberei

- Duden

Magie hat immer seinen Preis


JAKARTA™ EE
spring®
JPA, EJB, JAXP, CDI ...


@@
doubleSlash

```java
@Entity
@Table(name = "book")
public class BookEntity {
    @Id
    @Column(name = "isbn", nullable = false)
    private String isbn;

    @Column(name = "title", nullable = false)
    private String title;

    @ManyToOne
    @JoinColumn(name = "publisher_id")
    private PublisherEntity publisherEntity;

    @ManyToMany
    @JoinTable(name = "book_author", joinColumns = @JoinColumn(name = "isbn"),
    private List<AuthorEntity> authorEntities = new ArrayList<>();
...
}
```

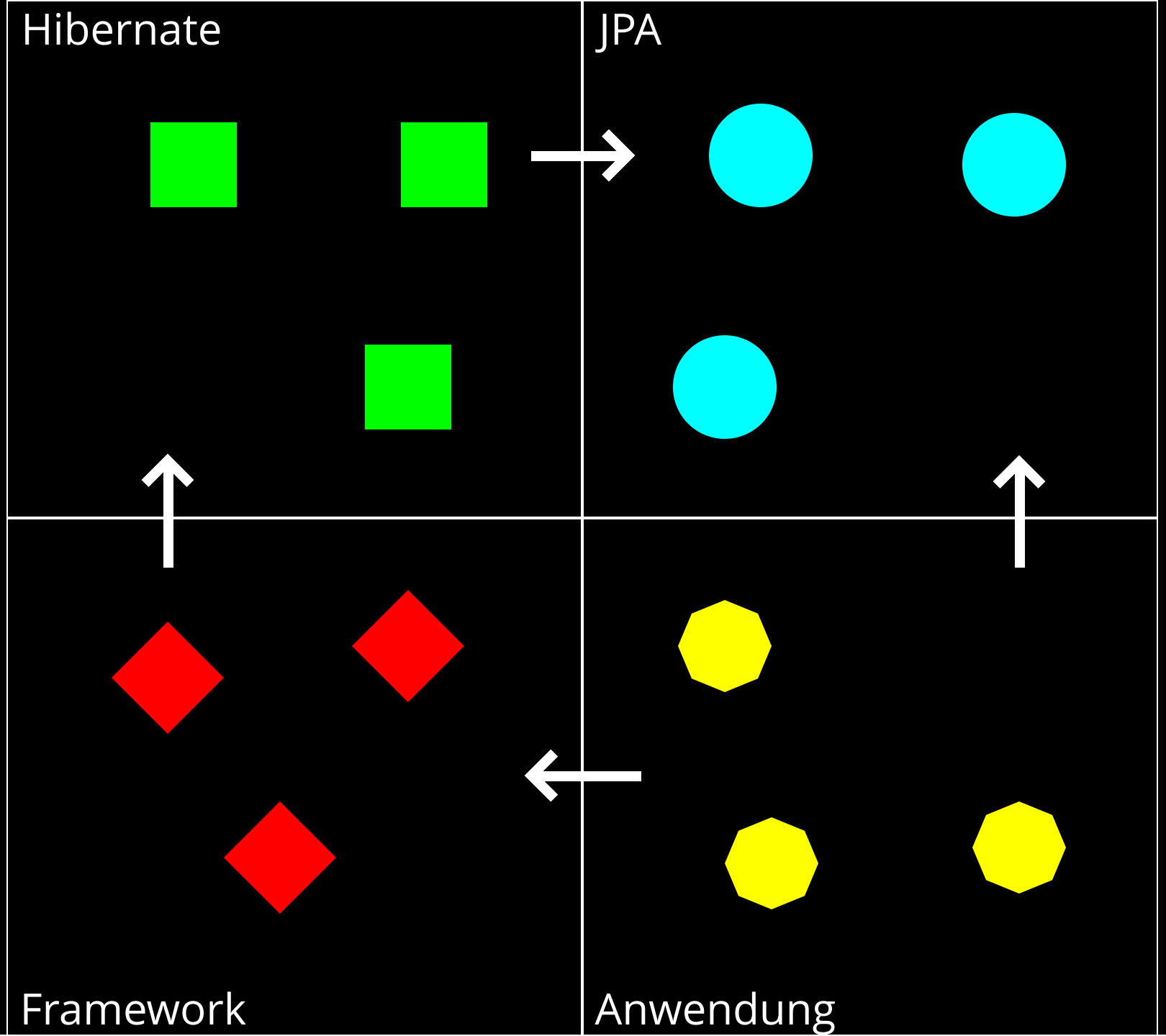
@nnotationen sind NUR Metainformationen und werden über Reflection ausgewertet

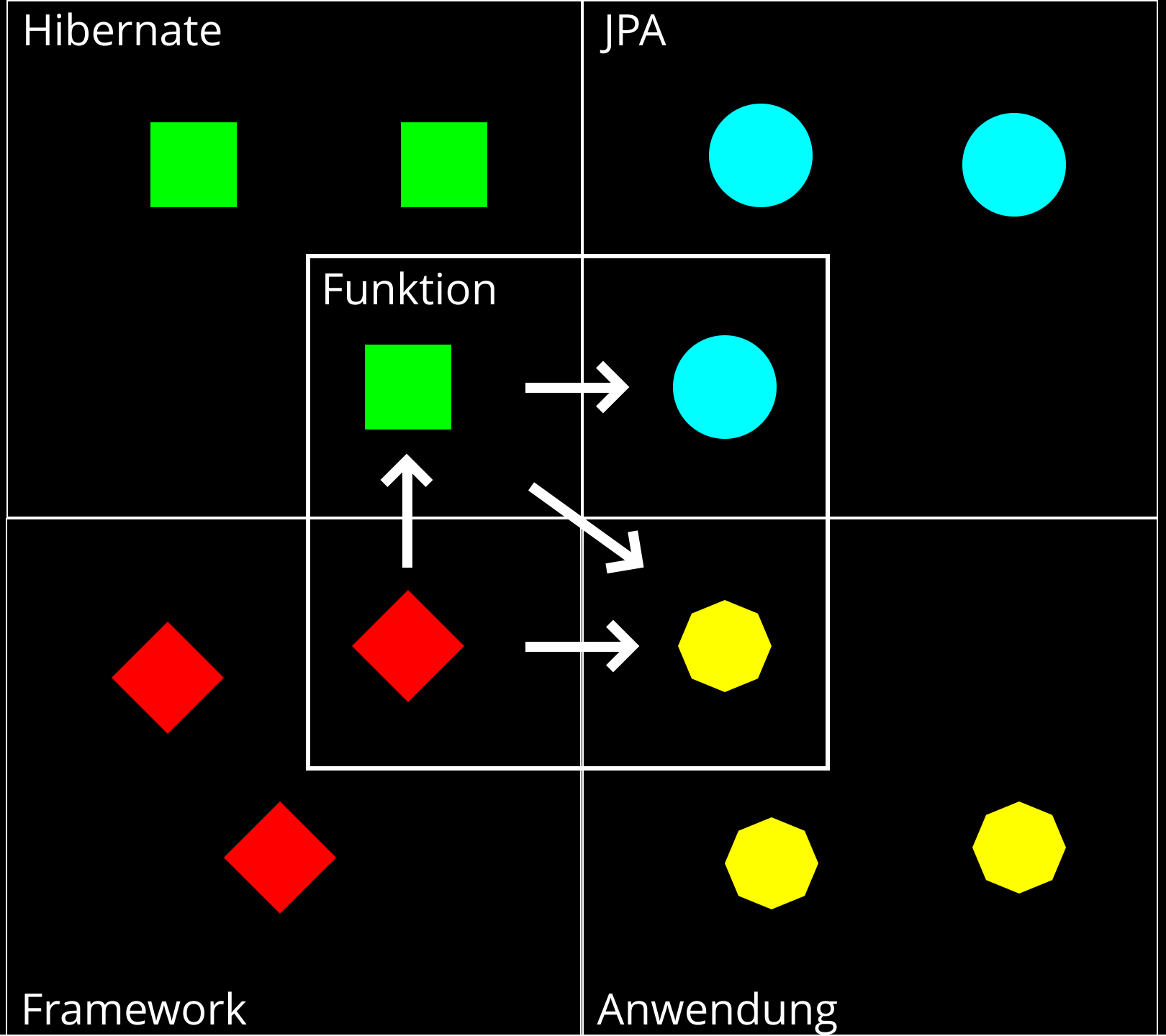
@nnotationen haben eine eigene Syntax

```java
public interface BookRepository extends CrudRepository<BookEntity, String> {
    boolean existsByTitle(String title);

    Optional<AuthorEntity> findByTitle(String title);

    List<AuthorEntity> findAllByTitleIn(List<String> titles);
}
```

Laufzeit Proxy übernimmt Implementierung

Methodennamen definieren Queries





HIBERNATE

Hibernate doesn't just offer simple mapping and querying features. Real-Life mappings and queries will obviously be much more complex than the ones you found above. In addition, Hibernate offers a ton of other *convenience* features, like cascading, lazy loading, caching and much more. It truly is a complex piece of software, that you cannot fully grasp by just copy-and-pasting some online tutorial.

This somewhat unexpectedly leads to two major issues.

1. A fair amount of developers, sooner or later, claim that "Hibernate is doing random magic, that nobody understands" or a variant thereof. Because they lack the background knowledge of what Hibernate is doing.

2. Some developers think that you do not need to understand SQL anymore, when using Hibernate. The reality is, the more complex your software gets, the *more* SQL skills you will need, to verify the SQL statements that Hibernate generates and optimize them.

To combat these two issues, you only have one choice: To use Hibernate efficiently, you need (to get) good knowledge of Hibernate **and** SQL.

Quelle: https://www.marcobehler.com/guides/java-databases

```
@Service
@ConditionalOnProperty(value = "logging.enabled", havingValue = "true", matchIfMissing = true)
class LoggingService { ... }

@Service
@ConditionalOnExpression("${logging.enabled:true} and '${logging.level}'.equals('DEBUG')")
class LoggingService { ... }

@Service
@Conditional(Java8Condition.class)
class LoggingService { ... }

@Service
@ConditionalOnClass(CustomLogger.class)
class LoggingService { ... }

@Service
@ConditionalOnBean(CustomLoggingConfiguration.class)
class LoggingService { ... }

@Service
@ConditionalOnJava(JavaVersion.EIGHT)
class LoggingService { ... }

@Service
@ConditionalOnWarDeployment
class LoggingService { ... }

...
```

```
{"id":12345,"username":"testuser","registrationDate":"2022-04-20T20:08:24.368Z"}
```

```java
@JsonInclude(JsonInclude.Include.NON_NULL)
public class JacksonUserDto {

  @JsonProperty("key")
  private long id;

  private String username;

  @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'")
  private LocalDateTime registrationDate;

  // ...
}
```

```java
JsonMapper mapper = new JsonMapper().registerModule(new JavaTimeModule()).registerModule(new Jdk8Module());
JacksonUserDto userDto = mapper.readValue("{ ... }", JacksonUserDto.class);
```

Annotationen haben das Potenzial, langfristig die Wartbarkeit von Software empfindlich zu beeinträchtigen. Hinter ihrer einfachen Form kann sich eine faktisch unbeschränkte Funktionalität verbergen, die es sogar ermöglicht, grundlegende Prinzipien der objektorientierten Programmierung außer Kraft zu setzen. Im Beitrag werden einige Aspekte dieser Gefahr genauer dargestellt. Ange-

Schlussfolgerungen

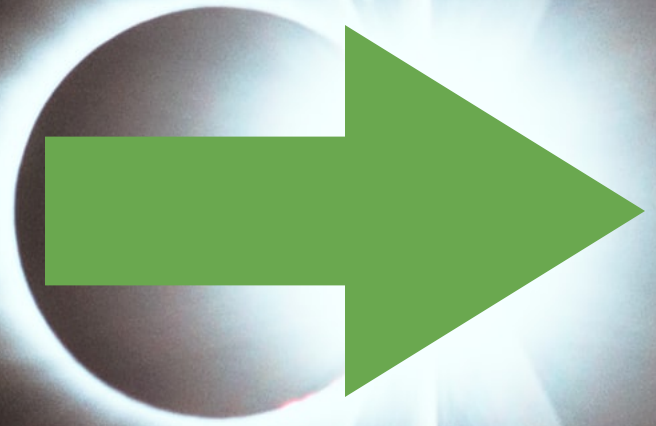
Annotationen und das, wozu sie verwendet werden können, sollen verteufelt werden. Der Erfolg zeigt, dass es einen breiten Bedarf an ergänzenden Technologien gibt, der durch die origi-

Quelle: https://entwickler.de/java/wolf-im-schafspelz

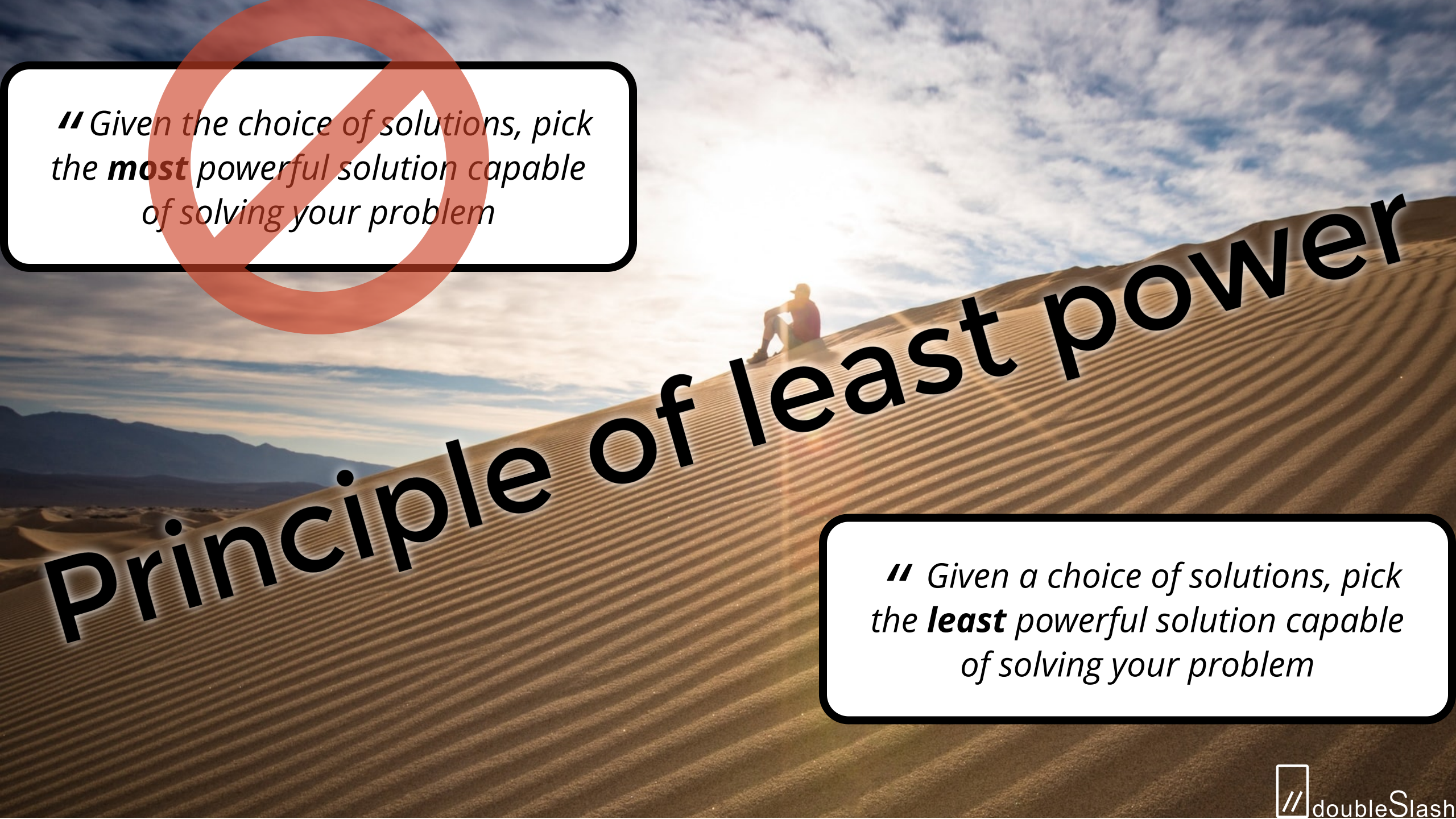
Die Lösung

doubleSlash

Problem



Lösung


" Given the choice of solutions, pick the most powerful solution capable of solving your problem
Principle of least power
" Given a choice of solutions, pick the least powerful solution capable of solving your problem
doubleSlash


Principle of least suprise
“If a **necessary** feature has a **high astonishment factor**, it may be necessary to **redesign the feature**.”
doubleSlash

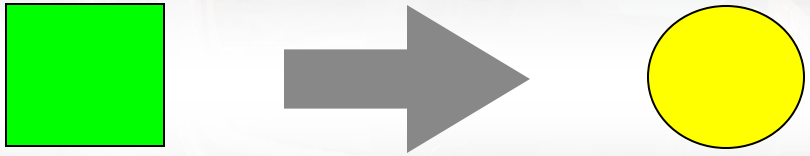
```java
@Service
@ConditionalOnExpression("${logging.enabled:true} and '${logging.level}'.equals('DEBUG')")
class LoggingService {
    // ...
}
```

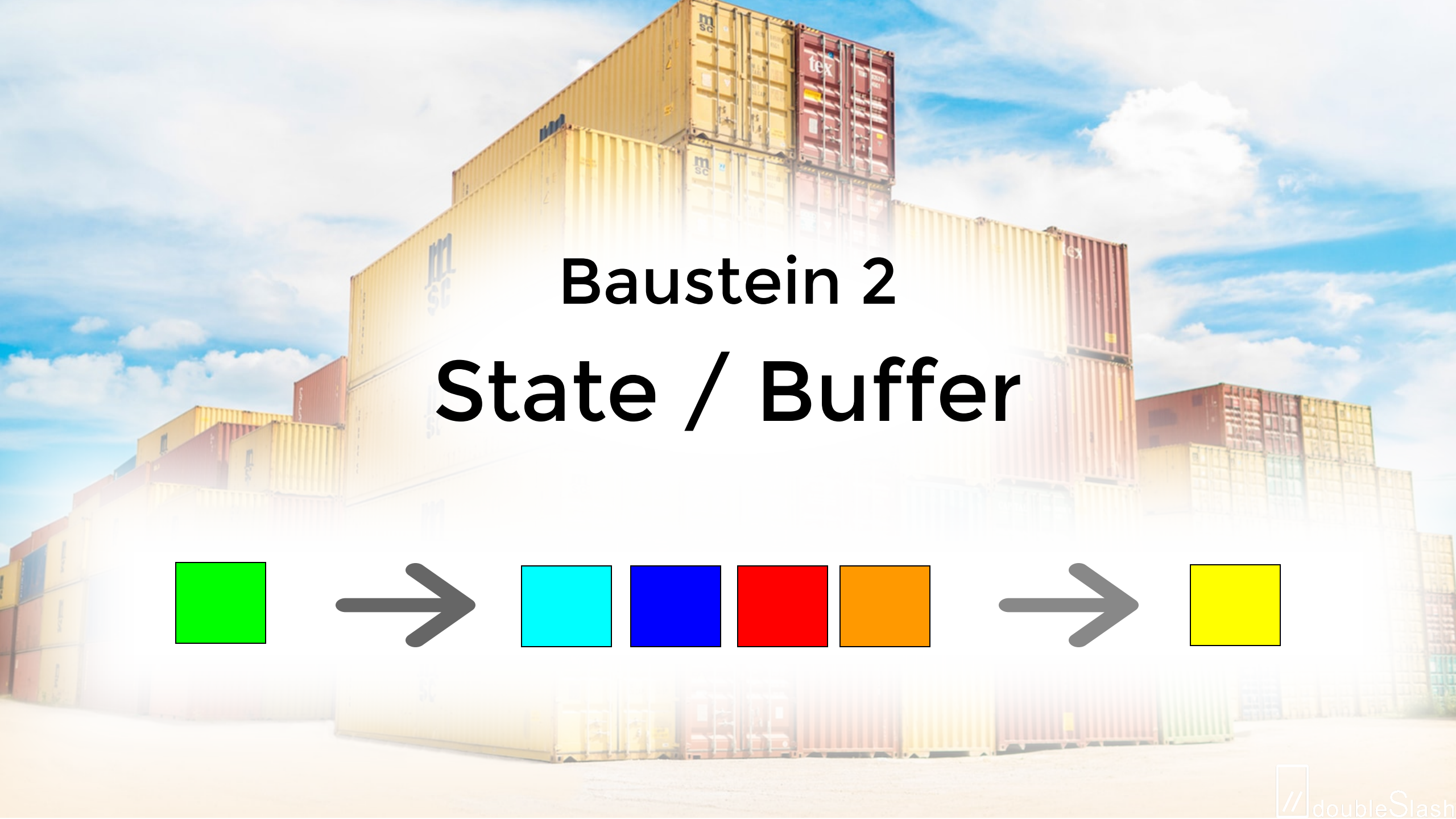
```java
final LoggingService loggingService;
if(environment.getProperty("logging.enabled", Boolean.class, true) &&
   environment.getProperty("logging.level", String.class, "INFO").equals("DEBUG")) {
  loggingService = new LoggingService(...);
} else {
  loggingService = ...;
}
```

Grundbausteine

Baustein 1:

IO - Input/Output



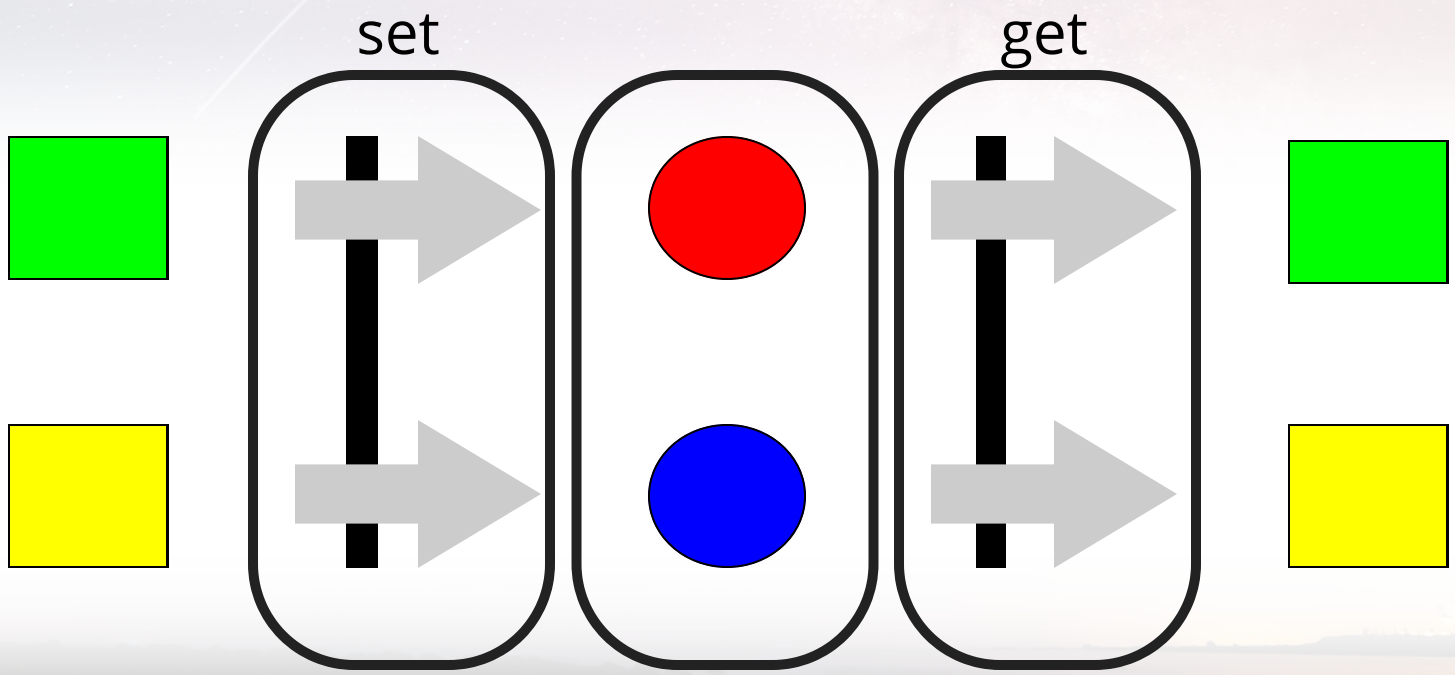


Baustein 2

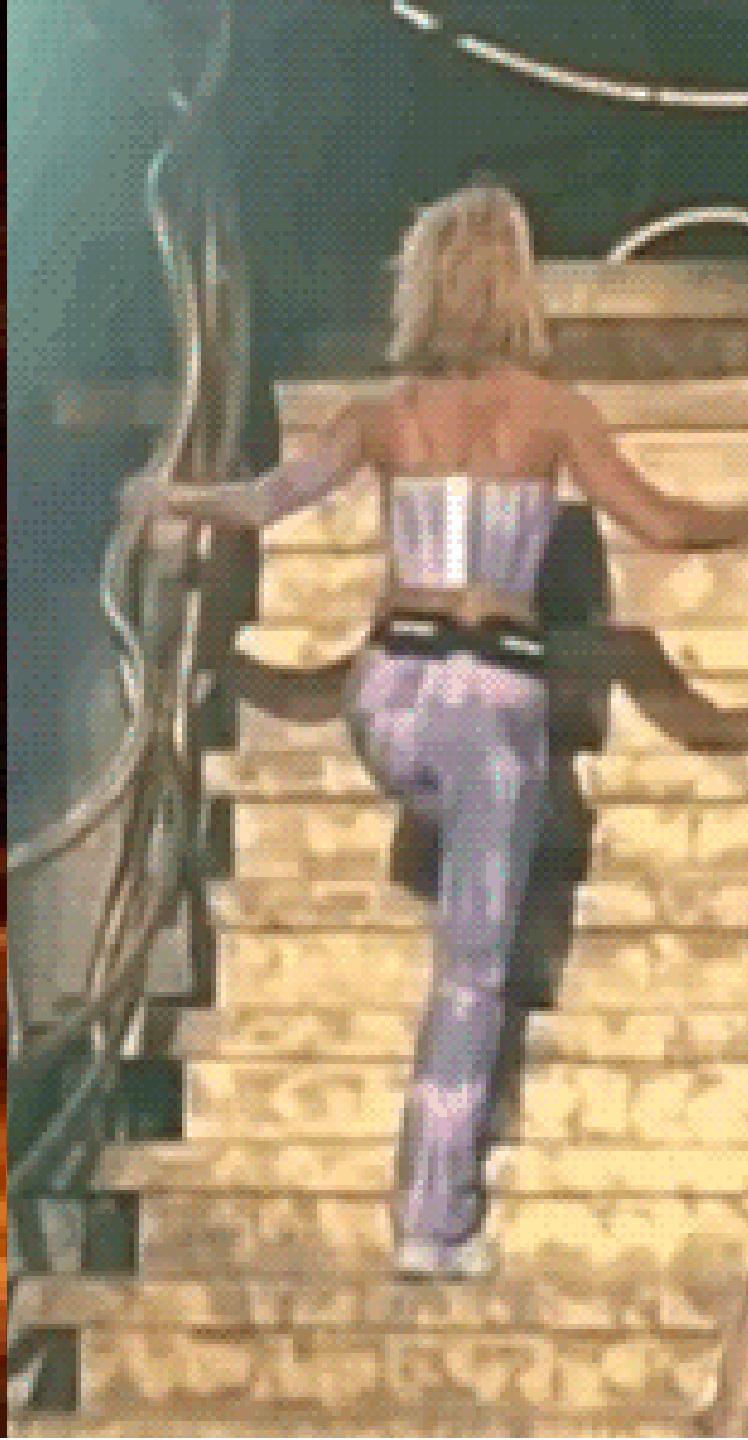
State / Buffer

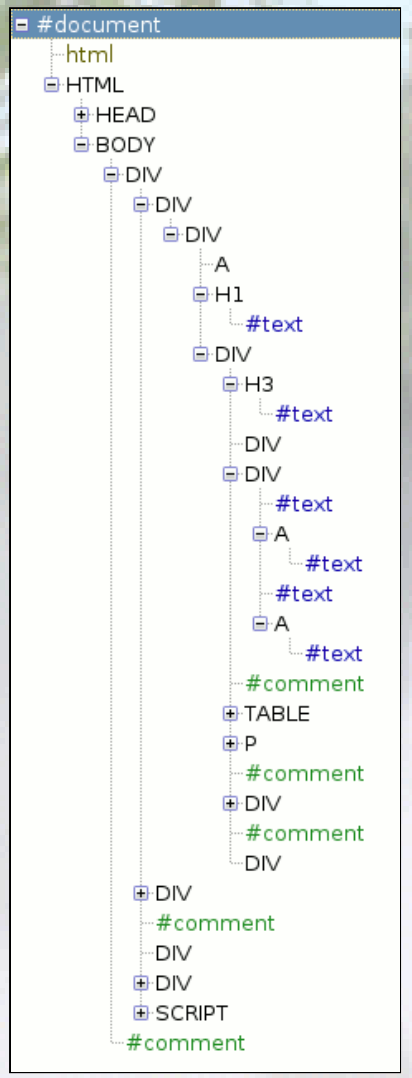
Baustein 3 = 1 + 2

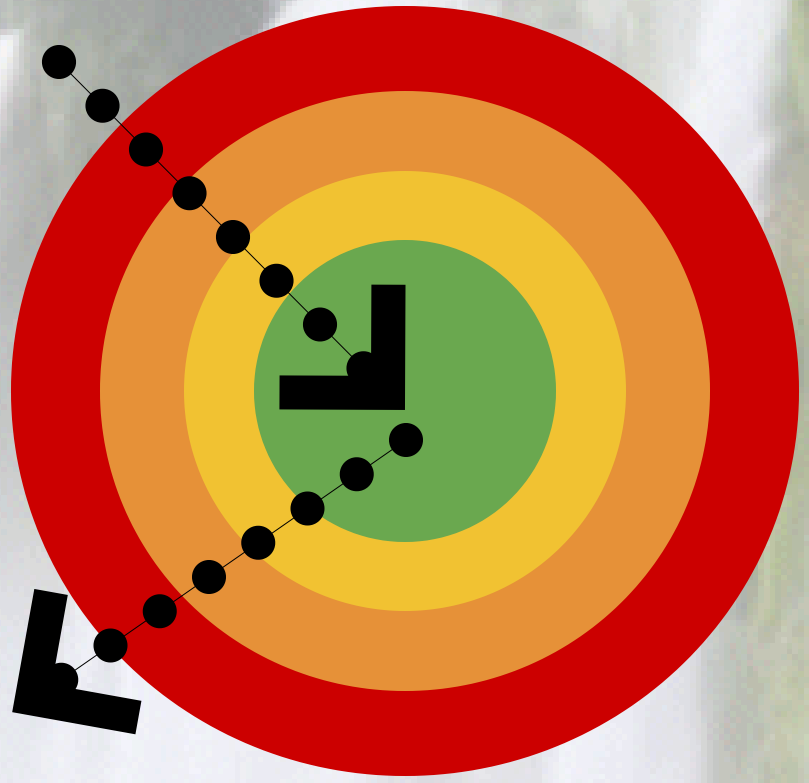
Wrappen / Kappseln / Verbergen / Abstrahieren

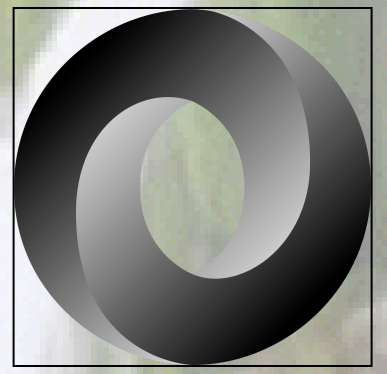


OOPs!... I Did It Again




#document
html
HTML
HEAD
BODY
DIV
DIV
DIV
A
H1
#text
DIV
H3
#text
DIV
DIV
#text
A
#text
#text
A
#text
#comment
TABLE
P
#comment
DIV
#comment
DIV
DIV
#comment
DIV
DIV
DIV
SCRIPT
#comment





```java
public final class SimpleUserDto {

  private static final DateTimeFormatter DATE_TIME_FORMATTER = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");

  private final JsonObject jsonObject;

  public SimpleUserDto(JsonObject jsonObject) {
    this.jsonObject = jsonObject;
  }

  public JsonObject getJsonObject() {
    return this.jsonObject;
  }

  public long getId() {
    return this.jsonObject
  }

  public void setId(long i
    this.jsonObject.set("i
  }

  public String getUsernam
    return this.jsonObject
  }

  public void setUsername(
    this.jsonObject.set("u
  }

  public LocalDateTime get
    final String dateStrin
    return dateString != nu
  }

}
```

```java
public final class SimpleUserRepository {

  private final DbClient client;

  public SimpleUserDto(DbClient client) {
    this.client = client;
  }

  public List<SimpleUserDto> getUsers() {
    return this.client.query("SELECT * FROM users").stream().map(SimpleUserDto::new).toList();
  }

  public void putUser(SimpleUserDto userDto) {
        this.client.save("users", userDto);
  }

  // ...

}
```

FAZIT

markus.hettich@doubleslash.de

markus.hettich@doubleslash.de

Magie lauert in der Softwareentwicklung überall

- Reflection (z.B. Annotationen in Java)
- Datenbindung (z.B. Angular)
- Condext Dependency Injection (z.B. Spring Framework)
- DSLs (z.B. Gatling)
- Skripte und integrierte Sprachen (z.B. SPeL)
- Implicit (z.B. Scala 2)
- Codegeneratoren (z.B. OpenAPI)
- Metaprogramming (z.B. Rust Macros)
- Vererbung (z.B. OOP)

...

Senior

Junior

Frameworks

Libraries

Pattern

Programmiersprache

markus.hettich@doubleslash.de

Explizit > Implizit

markus.hettich@doubleslash.de

markus.hettich@doubleslash.de

doubleSlash