Scala in Pattern

Martin.Eigenbrodt@innoq.com | @eigenbrodtm| bedcon 2014

Scala

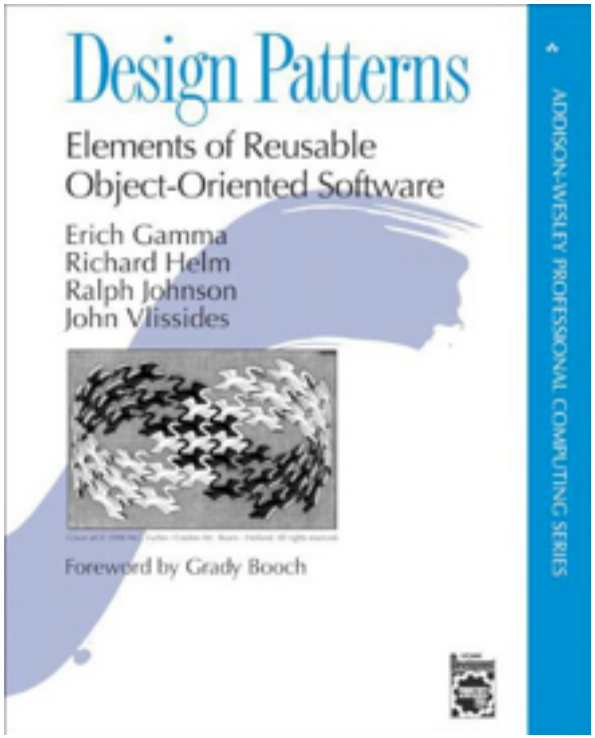
JVM

objektorientiert

funktional

statisch typisiert

Pattern



Lösungs Template für ein wiederkehrendes Problem

Andere Sprachen, andere Pattern

“Verschwundene Pattern”

innoQ

Singleton

Genau eine Instanz einer Klasse

Global adressierbar

```
public class Singleton {
    private static final Singleton
            INSTANCE = new Singleton();

    private Singleton() {}

    public static Singleton getInstance() {
        return INSTANCE;
    }
}
```

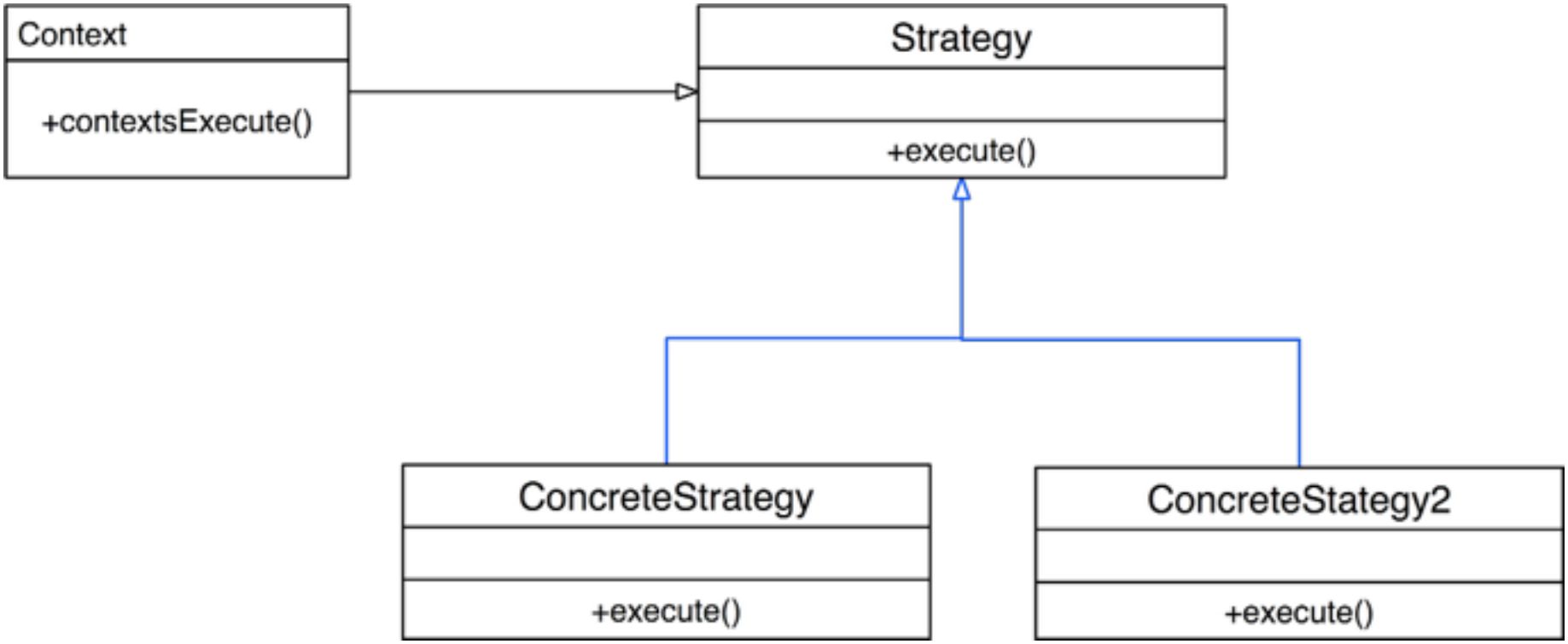
```
public class LazySingleton {
   private static volatile
      LazySingleton instance = null;

   private LazySingleton() {  }

   public static LazySingleton getInstance() {
      if (instance == null) {
         synchronized (LazySingleton.class) {
            if (instance == null) {
               instance = new SingletonDemo ();
            }
         }
      }
      return instance;
   }
```

object Singleton

Strategy

Definiert eine Familie von Algorithmen mit untereinander austauschbaren Implementierungen

Strategy



```
public interface FormatStrategy {
   public String format(int data);
}

public class PlainFormatStrategy implements
FormatStrategy {
   public String format (int data) {
      return "This is your int: " +  data;
   }
}

public class XMLFormatStrategy implements
FormatStrategy {
   public String format(int data) {
      return "<data>" + data + "</data>";
   }
```

```
class Context {
    private FormatStrategy strategy;

    public Context(FormatStrategy strategy) {
        this.strategy = strategy;
    }

    public String executeStrategy(int data) {
        return this.strategy.format(data);
    }
}
```

Eine *Strategie* ist *Verhalten*

Funktion

```
class Context (strategy : Int => String) {
  def executeStrategy(data : Int) = {
    strategy (data)
  }
}

val plainFormatStrategy =
   { i: Int =>  "This is your int:" + i }

val xmlFormatStrategy =
   { i: Int =>  "<data>" + i + "</data>"; }
```

Beispiel

```
  def Authenticated[A](
    userinfo: RequestHeader => Option[A],
    onUnauthorized: RequestHeader => Result)
(action: A => EssentialAction):
EssentialAction = {
…
    }

  }
```

Beispiel

```
userinfo: RequestHeader => Option[A]
```

“ExtractUserFromHttpRequestStrategy”

mehre “verlorene” Pattern

- Command
- Chain of Responsibility
- Visitor

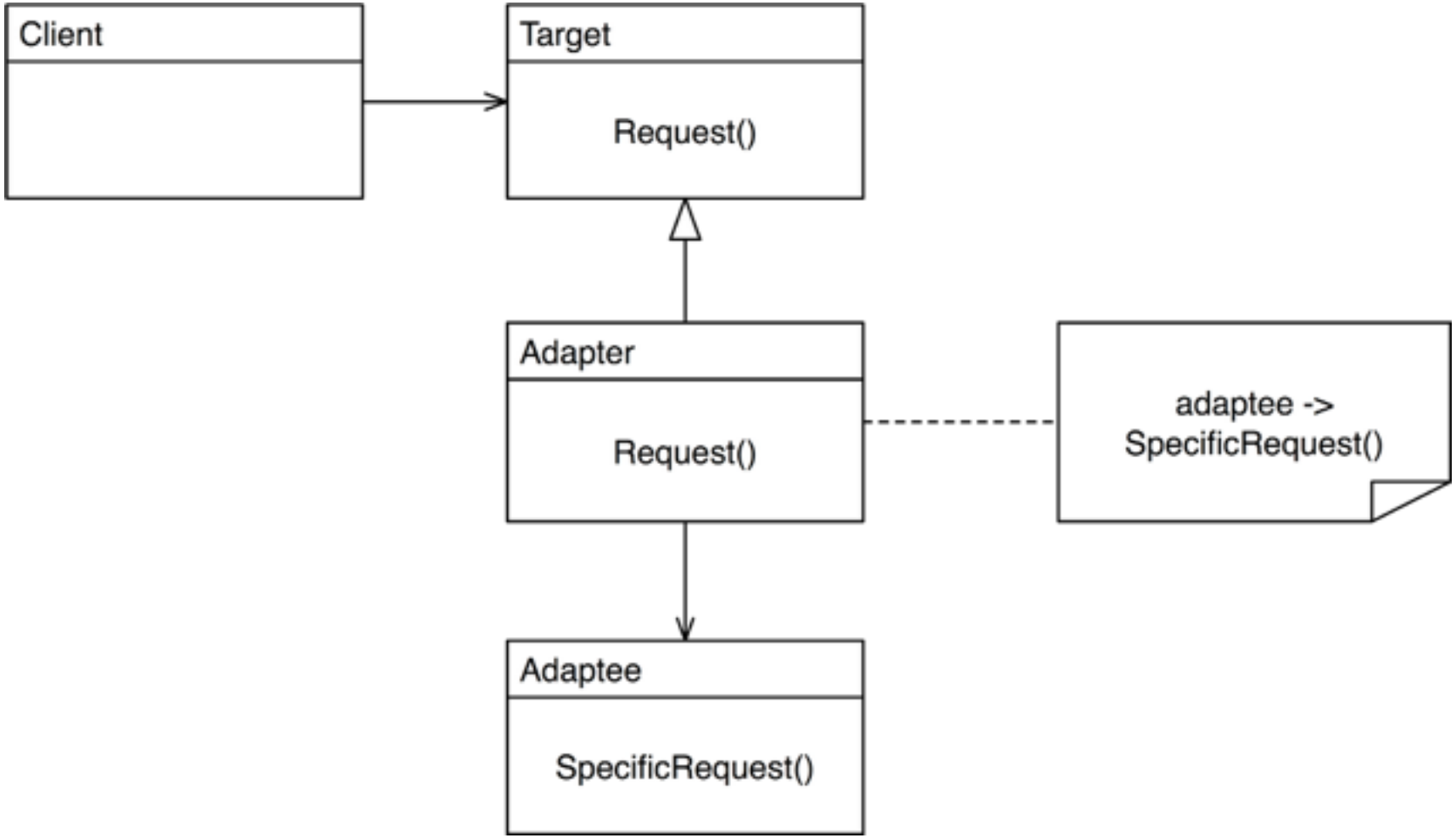
..

Pimp my Library

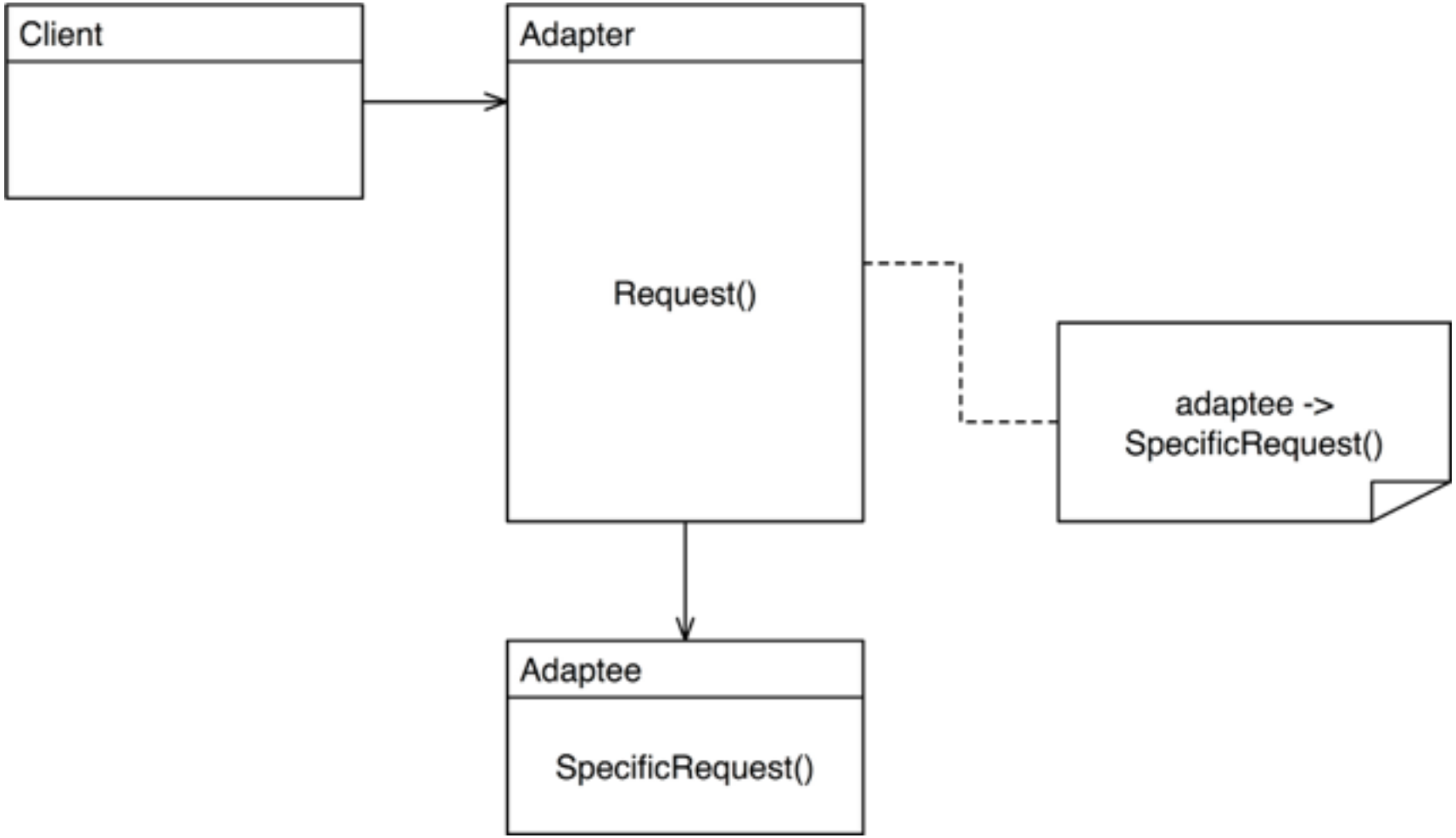
Fügt einer existierenden (geschlossenen)

Klasse neue Methoden hinzu

Adapter



Adapter



```
class StringAdapter (val adaptee : String) {
 def replaceLit (literal : String, replacement : String)={
    val quotedLiteral  = Pattern.quote(literal)
    val quotedRep = Matcher.quoteReplacement(replacement)

    adaptee.replaceAll(quotedLiteral, quotedRep)
  }
}

object StringAdapter {
   implicit def sToAdapter (adaptee : String) = new
      StringAdapter(adaptee)
}

// Client Usage
  import StringAdapter._
  "Hallo World".replaceLit ("a", "e")
```

“repaceLit” gibt’s schon:

```
"Hallo World".replaceAllLiterally ("a", "e")
```

Type Class

MyClass ∈ Druckbar

innoQ

```
   // Wie kann ich ein Objekt vom Typ T
drucken?

  trait Printable [T] {
    def print (a : T)
  }
```

```
implicit object StringPrintable
   extends Printable[String] {
     def print (a  : String) = {
       println (a)
     }
}
```

```
//Methode die ein A verlangt, das printable ist
  def printIt [A : Printable] (a : A) {
    val printable = implicitly[Printable[A]]
    printable.print(a)
  }


// Benutzung:
  printIt("Hello World")
```

In the wild:

```
// Eine Controller Methode in einer Play App
def index = Action {
    Ok ("Hello World")
  }

def json = Action {
    Ok (jsonObject)
  }
//Signature of "Ok"

Ok [C : Writeable] (content: C) …
```

Thank you!
Questions?
Comments?

Martin Eigenbrodt, @eigenbrodtm
martin.eigenbrodt@innoq.com
Phone: +49 0171 3333 085

innoQ