MVC

Web Engineering

Dierk König
Christian Ribeaud

Testing

Is a core tenet of engineering.

Provides a validatable specification as opposed to "just give it a try".

grails test-app

Functional integration tests to validate the behavior as seen by the user.

Unit tests to validate controllers, models, and services in isolation.

Geb and Spock

Geb: user interactions with the HTML page

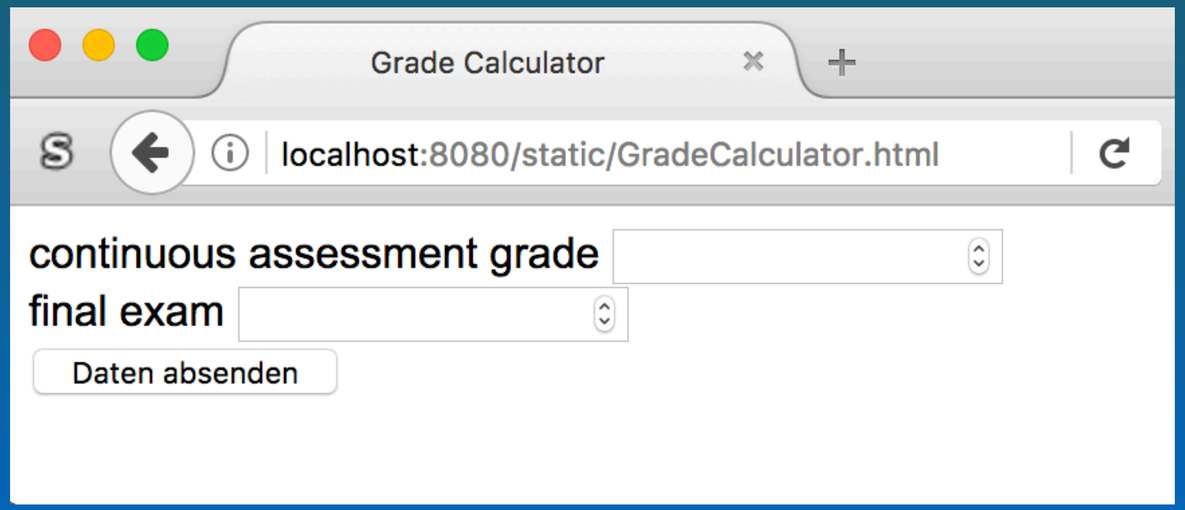
Spock: structure test cases in given - when - then - expect and allow data-driven tests

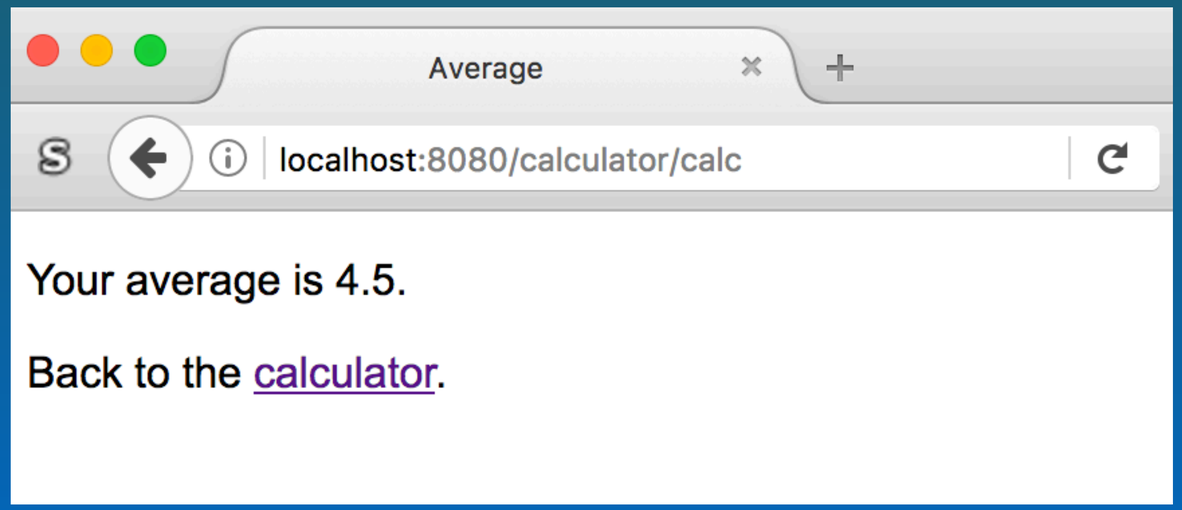
Geb selector $()

$("div.cool")	* *
$("a", href:"x")	*<a href="x">*
$("a", text:"x")	*<a>x</a>*
$("form").en	*<input name="en">*

Web View-Controller

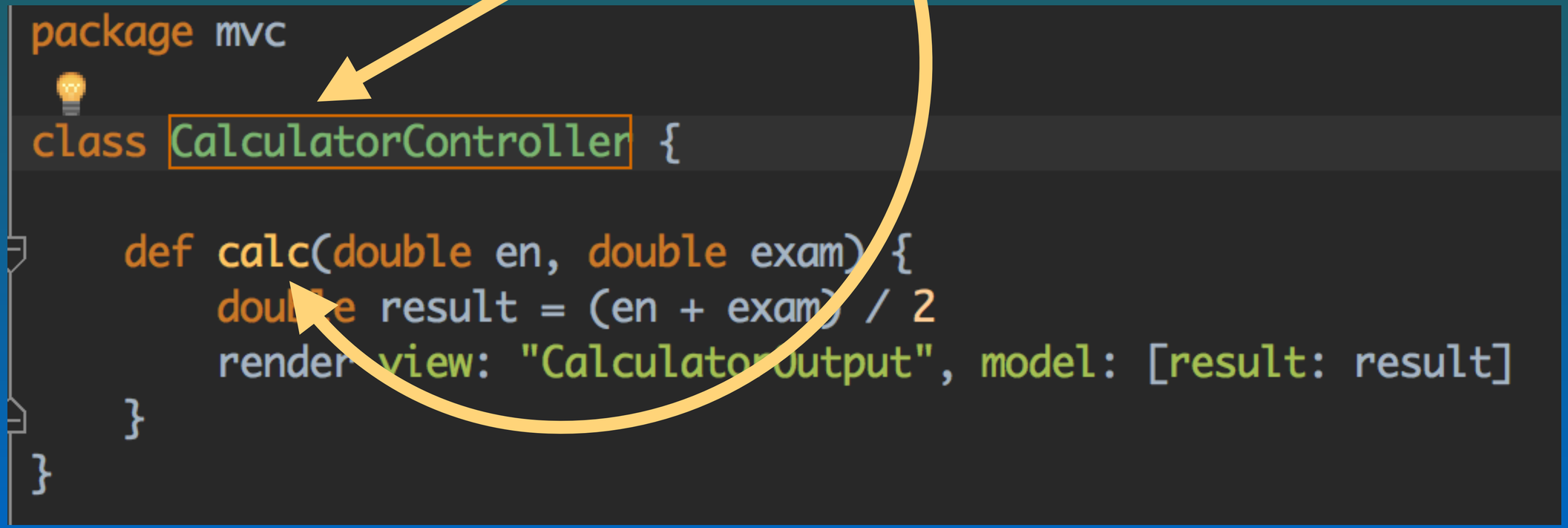
```
class MyController
  def myAction(en, exam)
```





Dispatch

/calculator/calc



View Selection

```
package mvc

class CalculatorController {

    def calc(double en, double exam) {
        double result = (en + exam) / 2
        render view: "CalculatorOutput", model: [result: result]
    }
}
```

views/calculator/
CalculatorOutput.gsp

View Binding

```groovy
package mvc

class CalculatorController {

    def calc(double en, double exam) {
        double result = (en + exam) / 2
        render view: "CalculatorOutput", model: [result: result]
    }
}
```

views/calculator/
CalculatorOutput.gsp

GSP View

use of view binding

```
<p> Your average is <output>${ result }</output>.</p>

<p> Back to the <a href="/static/GradeCalculator.html">calculator</a>.</p>
```

Web MVC

data binding

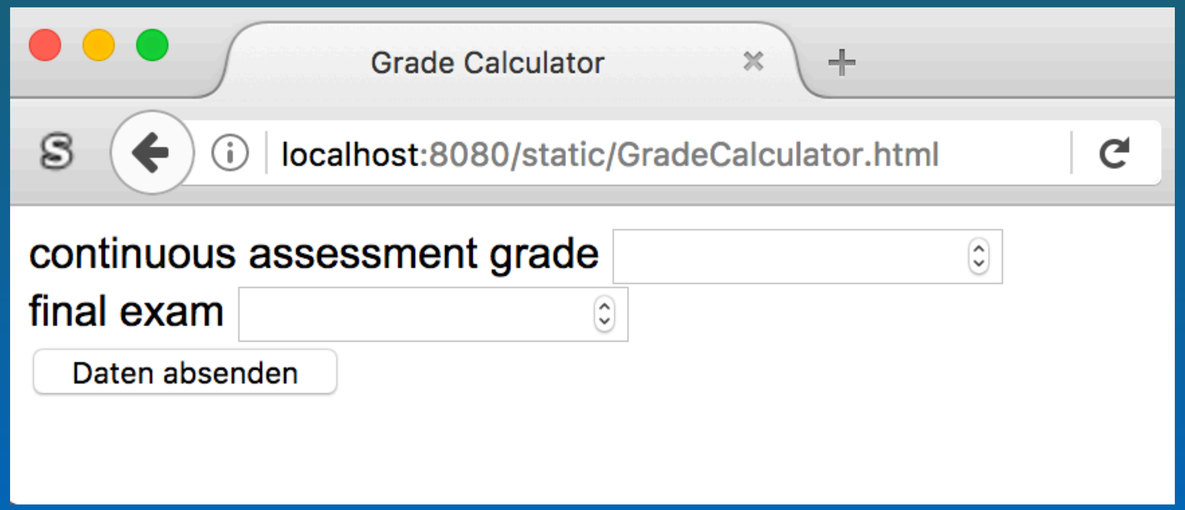
dispatch

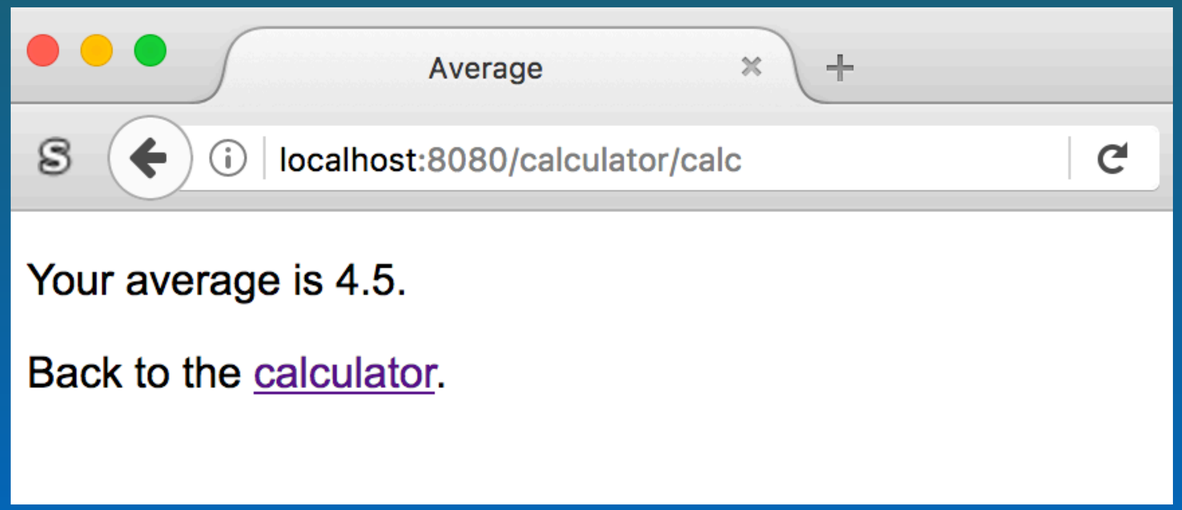
class MyModel
def en, exam, result

class MyController
def myAction(model)

view binding

view selection





See Examples

views/inPlaceCalculator/calc.gsp

controllers/mvc/
InPlaceCalculationController.groovy

(same file) class CalculatorModel

Validation

In the view (trust: never!)

In the controller (imperative)

In the data binding (declarative)

Place of declaration: model constraints