

MVC

Web Engineering
Dierk König

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")`

`<div class="cool">`

`$("a", href:"x")`

``

`$("a", text:"x")`

`<a>x`

`$("form").en`

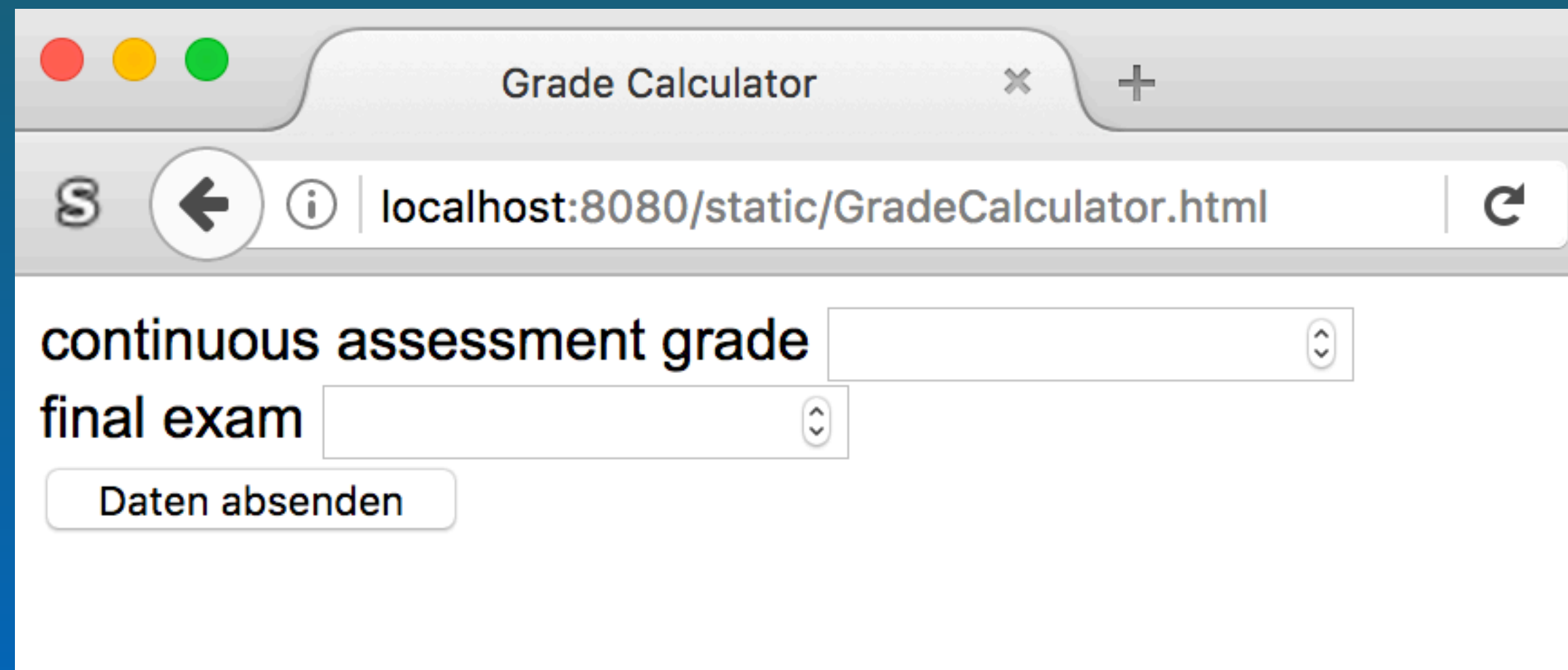
`<input name="en">`

Web View-Controller



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

The diagram illustrates the Web View-Controller pattern. A central code block for `MyController` is shown. A white arrow on the left points from the code block to a browser window representing the 'Grade Calculator' view. A white arrow on the right points from the code block to another browser window representing the 'Average' view, which is the result of an action.



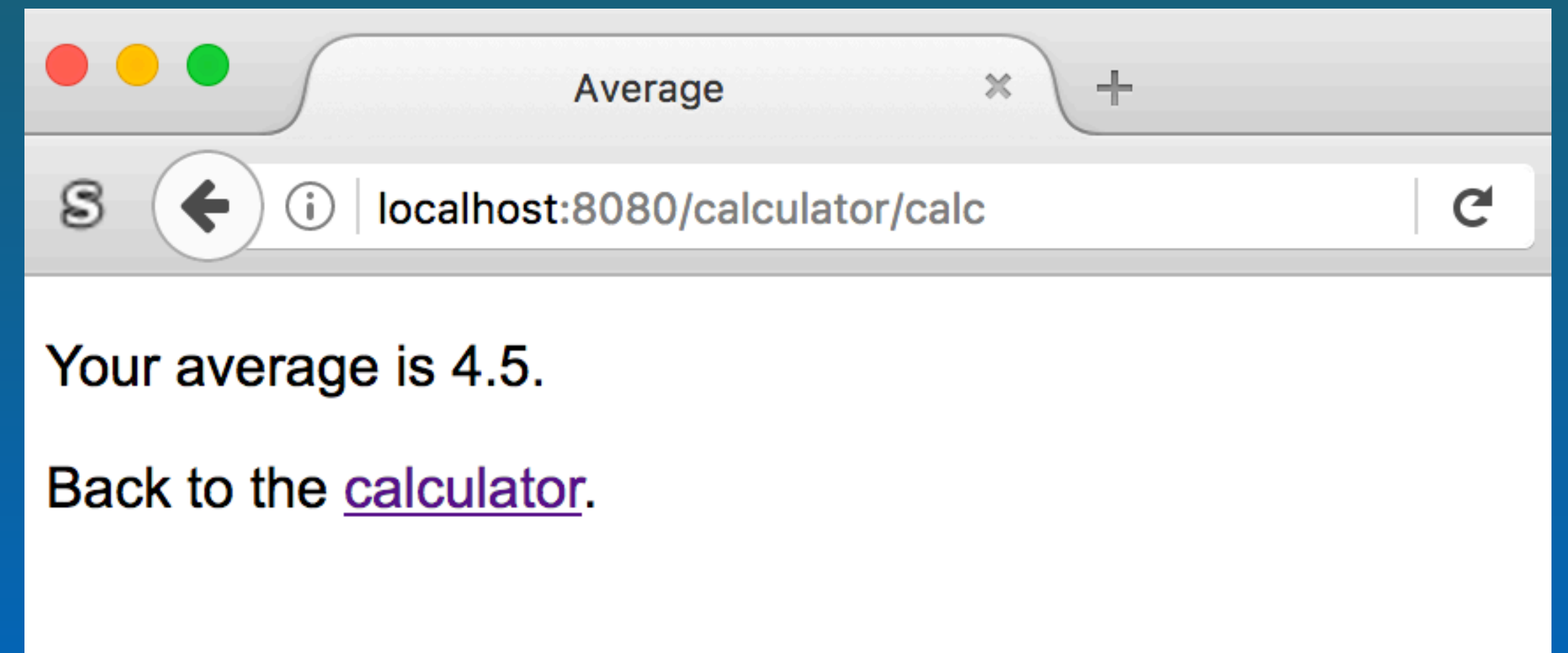
Grade Calculator

localhost:8080/static/GradeCalculator.html

continuous assessment grade

final exam

Daten absenden



Average

localhost:8080/calculator/calc

Your average is 4.5.

Back to the [calculator](#).


Dispatch

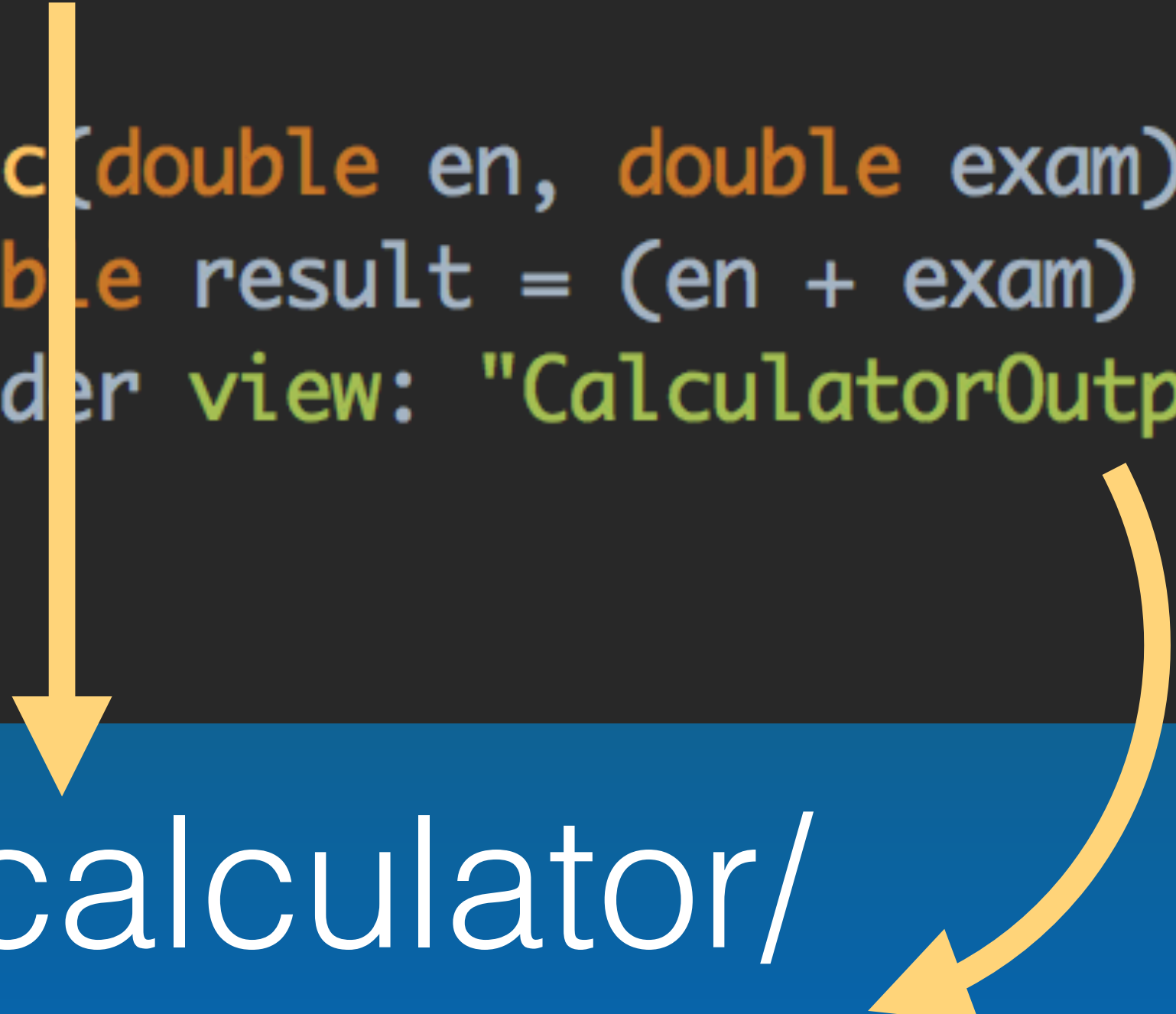
/calculator/calc

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

The diagram illustrates the dispatch of a request to a controller method. A yellow arrow points from the path `/calculator/calc` to the `CalculatorController` class. Another yellow arrow points from the `calc` method within the class to the `calc` part of the path, indicating the mapping between the request and the method.


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

```
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>
```

```
<💡> 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

Grade Calculator

localhost:8080/static/GradeCalculator.html

continuous assessment grade

final exam

Daten absenden

Average

localhost:8080/calculator/calc

Your average is 4.5.

Back to the [calculator](#).

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