|  |
| --- |
| National Coding Week  Make a web app with Java |

# Contents

National Coding Week 1

Make a web app with Java 1

Contents 2

Welcome 3

Be social 3

Java Track 4

Stage One – Sample code to display your web app 4

Stage Two – A new page and HTML Form 5

Stage Three – Posting your HTML form 6

Stage Four – A bit of validation 6

Stage Five - Pièce de résistance 7

# Welcome

On behalf of ResponseTap and MadLab welcome to National coding week.

This guide is for the Java track during today’s event. We hope you find the coding examples and guides useful today and more importantly, **fun!!**

# Be social

Our Twitter hashtag for todays event is #NationalCodingWeek

Some of the Twitter handles from todays event:

@ResponseTap <http://responsetap.com>

@madlabuk <http://madlab.org.uk>

@eggsy84

# Java Track

The following sections outline some of the ideas you might make use of during the Java coding exercises.

Rather than the usual “lets print Hello World to the screen” exercises we’ve tried to make them a little more industry relevant. Making use of the popular Spring Framework with Java.

With that in mind the exercises will show you how a Java based web application and how to display content on web pages as well as submit forms to the server. Your ResponseTap volunteer will take you through each stage.

For the daring amongst you, we also have a bonus section at the end that will involve text to speech and phone calls ;)

Stage One – Sample code to display your web app

The initial web application.

After starting the application, open your favourite browser and try going to:

<http://localhost:8080/greeting>

You should see the Hello World message.

Now try going to:

<http://localhost:8080/greeting?name=John>

The code that generates this page is located in the GreetingController.java

At this stage we can introduce Java variables and the ‘if then’ conditional control statement.

Lets trying something like:

@RequestMapping(method= RequestMethod.***GET***, value=**"/greeting"**)  
**public** String greeting(@RequestParam(value=**"name"**, required=**false**, defaultValue=**"World"**) String name, Model model) {  
 *// Variable declaration* String messageOnScreen;

*// If conditional logic* **if**( name.equalsIgnoreCase(**"John"**)) {  
 *// variable re-assignment* messageOnScreen = **"Java Ninja"**;  
 }  
 **else** {  
 messageOnScreen = **"Java Novice"**;  
 }  
 model.addAttribute(**"name"**, messageOnScreen);  
 **return "greeting"**;  
}

After restarting your application, try going to the same page again and see what displays this time. Try changing the ‘John’ part of the page address to something different. Notice how your conditional logic is observed by what is displayed on the page.

Congratulations you just edited your first Java web application!

Stage Two – A new page and HTML Form

In stage two we introduce another page called phonecall.

We’ll need a new controller and a new HTML page.

Try using the example code for our GreetingController.java to create another Java class called PhonecallController.java.

You’ll also need a new HTML page called “phonecall.html” and anyone needing a hint, your controller should return a String called “phonecall”. This mirrors the name of your new HTML page.

Once you’ve got the new controller class organized we can start to learn a little more of the Java syntax.

Lets introduce Java loops, below is a Java loop that counts up from 1 to 10 and prints the value to the screen.

for(int i=1; i<11; i++){

System.out.println("Count is: " + i);

}

Try editing your code to introduce a loop like this. How could you change the code to make the counter go up to 20 instead of 10?

As a bonus, try getting the values printed to your new HTML page ;)

Stage Three – Posting your HTML form

Stage three builds on the HTML form created in stage 2. We’ll write code that handles submission of the form (when you click the submit button) and displays what you entered back to the screen.

We can also introduce another control statement called the ‘switch’ statement.

int number = 2;

String someString;

switch (number) {

case 1:

someString = "Number was 1";

break;

case 2:

monthString = "Number was 2";

break;

default:

monthString = "Number wasn’t 1 or 2";

break;

}

At this stage, your ResponseTap volunteer should also give you a very quick overview of the Request/Response model used by all internet applications.

Stage Four – A bit of validation

By now you’ve hopefully created a HTML form, two fields for message and telephone number and Java code to handle the submission on the form.

What would happen if someone submitted the form without filling the values out correctly?

In steps validation to save the day.

Using Spring, we can make sure that the values submitted fit the patterns we expect. For example, we can ensure that the user enters a minimum/maximum of 11 digits for their phone number.

If the user submits the form without all the required validation elements then we can make return a suitable error back to the user.

We’ll show you how this can easily be achieved.

Stage Five - Pièce de résistance

The final stage is hooking everything in to one of the ResponseTap API’s for making a phone call.

We’ll hook in your form submission and validation to an API endpoint that can generate phone calls.

Try entering your own mobile number in to the text box and an interesting message ;)