**Studio: Bonjour, Spring!**

In the [prep work](https://education.launchcode.org/skills-back-end-java/class-prep/3/) for this class, you create a basic Hello, World application using Spring Boot. Open that project up in IntelliJ, and get ready to add some features!

Modify your HelloController class to display a form on GET request that asks the user for both their name and the language they would like to be greeted in. It should look something like this:



The resulting form submission should return and display the message, "Bonjour Chris".

Note that the language is presented in a dropdown, more formally known as a select element. If the syntax of selects is fuzzy, [quickly brush up](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select).

When the user submits the form (a POST request), they should be greeted in the selected language. Your new feature should:

* Include at least 5 languages, with English being the default. If you don't speak 5 languages yourself, ask your friend [the Internet](http://pocketcultures.com/2008/10/30/say-hello-in-20-languages/).
* Include a new public static method, createMessage, in the HelloController that takes a name as well as a language string. Based on the language string, you'll display the proper greeting.

**Bonus Missions**

* Instead of returning the greeting as plain text, add a bit of HTML to the response string so that the displayed message looks a bit nicer.
* Restructure your code so that the controller class doesn't know anything about the specific languages available. In other words, it asks the model for available languages to present to the user.
* Add some additional output that displays the number of times the user has been greeted. *Hint:* Use a static property to keep track of the count.
* The bonus mission above doesn't discriminate between requests made by you or somebody else. In other words, it counts *total* greetings rather than greetings to a specific user. Fix this by using cookies. When a user is greeted for the first time, set a cookie that has the visit count 1. On subsequent visits, check for this cookie and update its value. Here are a few tips to get you going:
  + You'll need to access both the request and response objects for the given request, to get and set the cookie, respectively. You can add these as parameters to your controller method by included them in the signature:

public String hello(HttpServletResponse response, HttpServletRequest request, /\* other params \*/)

* + With these params in place, request.getCookies() will return an array of Cookie objects, and response.addCookie(myCookie) will set a cookie in the response.
  + To find a given cookie, you'll need to loop through your cookie array, using getName on the cookie objects, and getValue when you find the cookie you're looking for.
  + Don't forget to display your cookie value to test that the code is working! (Though, you can also see cookies in your browser dev tools.)
  + You might find the [documentation for the Cookie](http://docs.oracle.com/javaee/6/api/javax/servlet/http/Cookie.html) class helpful.