**Postman**

Postman is a handy web app we will be using for the next couple days to test our routes and controller actions.

The Postman app should already be installed on all a/A machines, but if you would like to install it on your personal machine, download it [here](https://www.getpostman.com/).

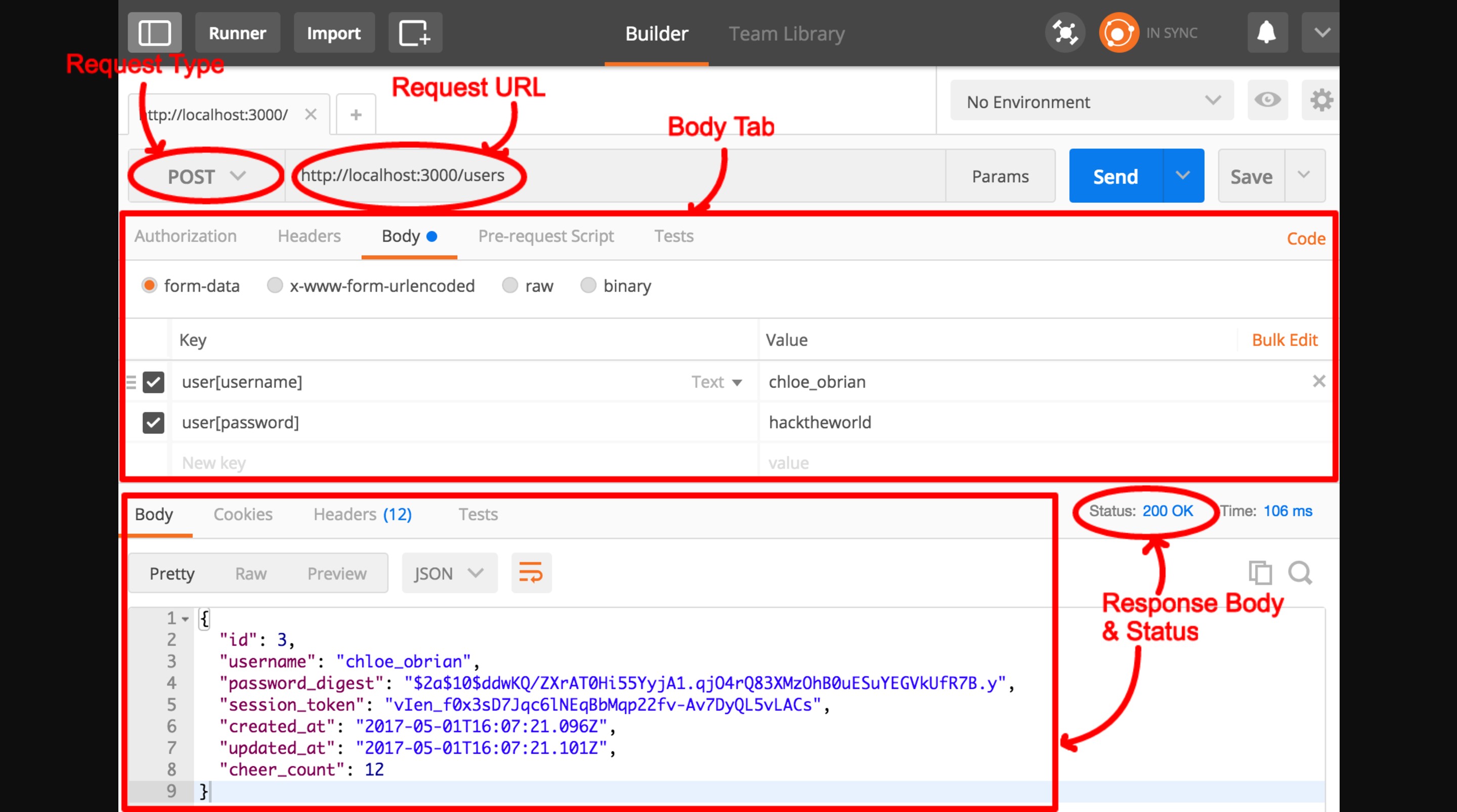
**Making requests**

We have four main components of Postman we will be using: 1. Request Type - change the dropdown to send GET, POST, PATCH, DELETE requests 2. Request URL - change the path to which our request is made 3. Body Tab - input the body/data of our request

* We recommend using the key-value pairs in the form-data sub-tab

1. Response - see the response body and HTTP status

Optionally, you can build a query string (the bit after the ?) by clicking Params to the right of the Request URL and adding key-value pairs. These are received as top-level params by the controller. For example, a request made to localhost:3000/users?username=jack\_bauer has params[:username] # => 'jack\_bauer'.



**Demo**

Let's say we have a Rails app with a route of POST /users, which is our path to create a new user.

1. Change the Request Type to POST
2. Since we are running Rails in the development environment, change the Request URL to localhost:3000/users
3. Add the request body. This will contain the key-value pairs corresponding to the attributes we would like to set for our new user

* Rails convention is to nest the attribute name (username, password) inside resource name (user).
* In this example, we will create a new User with a username of chloe\_obrian and a password of hacktheworld. To input this we will have the following key-value pairs:
* user[username]: chloe\_obrian

user[password]: hacktheworld

1. At this point, we are ready to make our request, but in order to make a request from outside of localhost:3000 we will need to turn off CSRF protection. In app/controllers/application\_controller.rb, comment out the protect from forgery line. Remember, this should only be done in development.
2. Finally, hit Send to make the request and inspect the response. Make sure to verify that you are receiving not only the correct response body, but also the correct status.

**A note regarding redirects**

When your app responds with redirects, Postman will intelligently try to follow the redirect, but not so intelligently keep the same request type. This means if we make a request to DELETE /users/1 and our app responds with a 302 Redirect to /home, Postman will make a request to DELETE /home even though it *should* be making a GET request.

For this reason, we recommend *not* testing any routes that redirect or test them without the redirect (stub a dummy response).