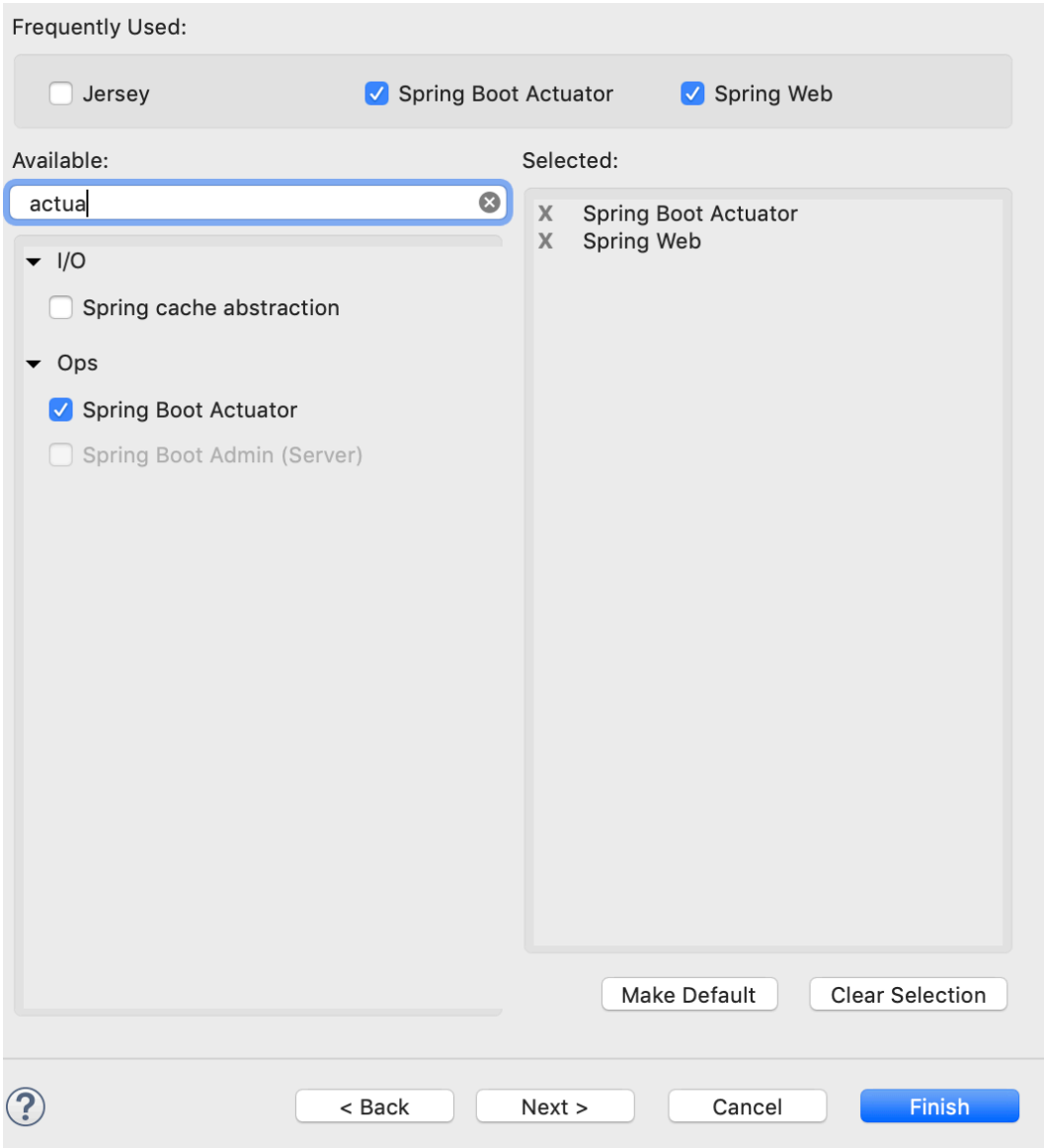
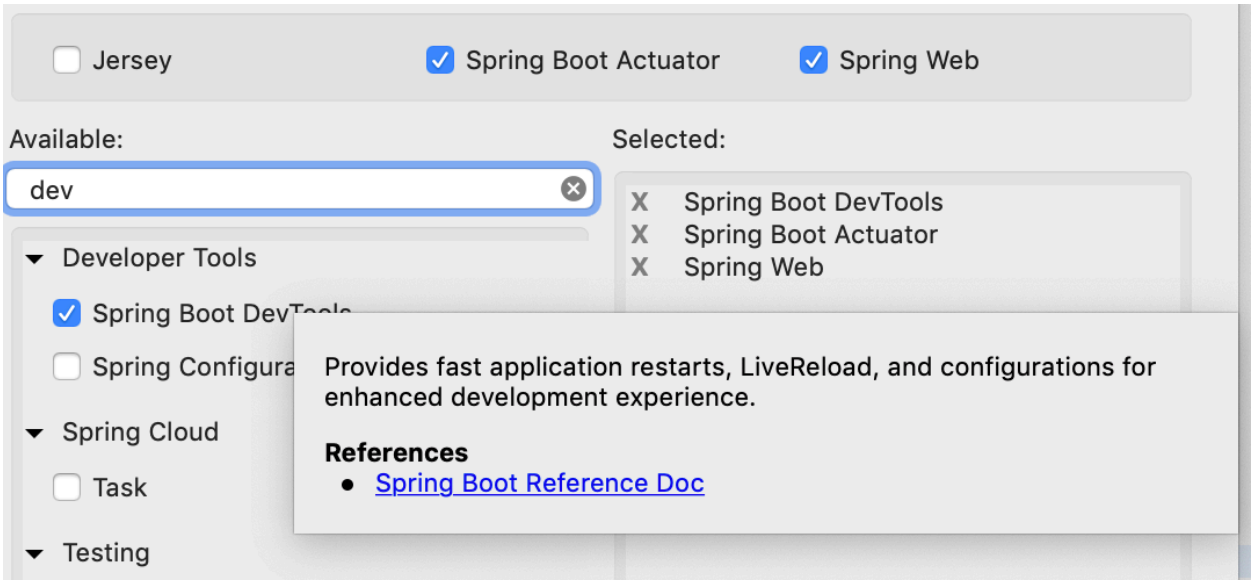
Create a ToDo REST API with CRUDS Operations

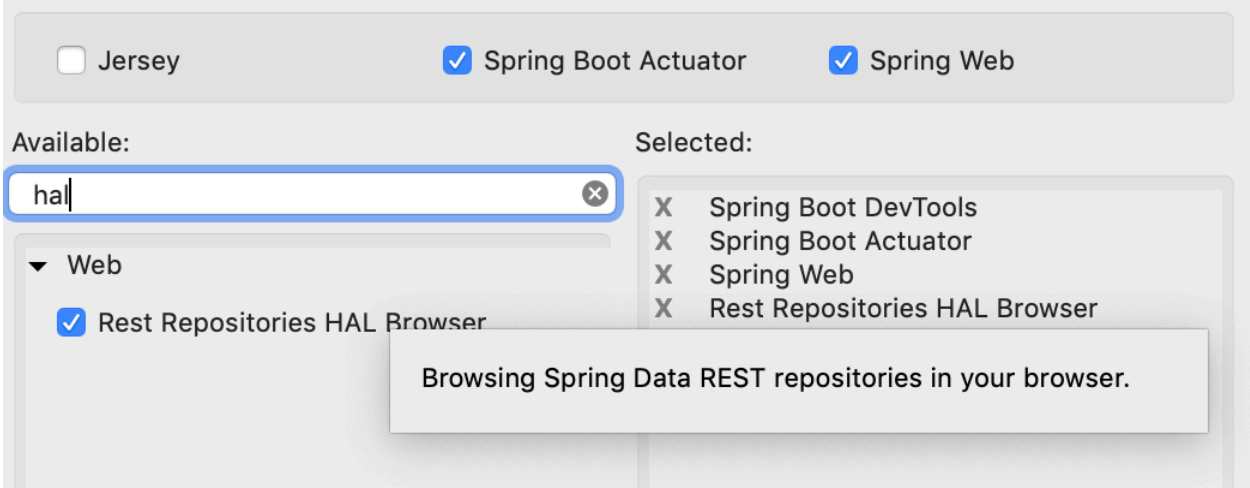
1. File -> New -> Spring Starter Project
2. Name = todo-cruds
3. Group = com.todo
4. Package = com.demo
5. Click on Next
6. Select “Spring Web”, “Actuator”, “DevTools” & “HAL Explorer” module



6.1


Jersey
Spring Boot Actuator
Spring Web
Available:
dev
Selected:
X Spring Boot DevTools
X Spring Boot Actuator
X Spring Web
Developer Tools
Spring Boot DevTools
Spring Configura
Spring Cloud
Task
Testing
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
References
Spring Boot Reference Doc

6.2


Jersey
Spring Boot Actuator
Spring Web
Available:
hal
Selected:
X Spring Boot DevTools
X Spring Boot Actuator
X Spring Web
X Rest Repositories HAL Browser
Web
Rest Repositories HAL Browser
Browsing Spring Data REST repositories in your browser.

7. Click on Finish

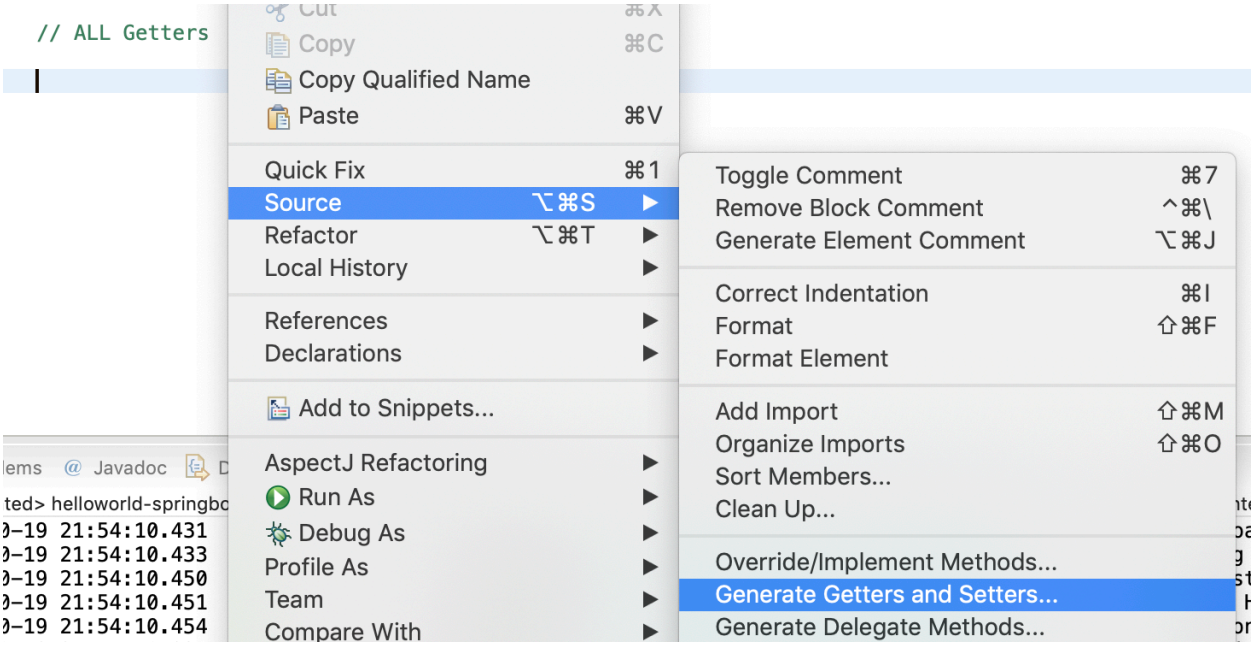
8. Open pom.xml and change dependency for HAL Browser as “explorer”:

```
<dependency>
	<groupId>org.springframework.data</groupId>
	<artifactId>spring-data-rest-hal-explorer</artifactId>
</dependency>
```

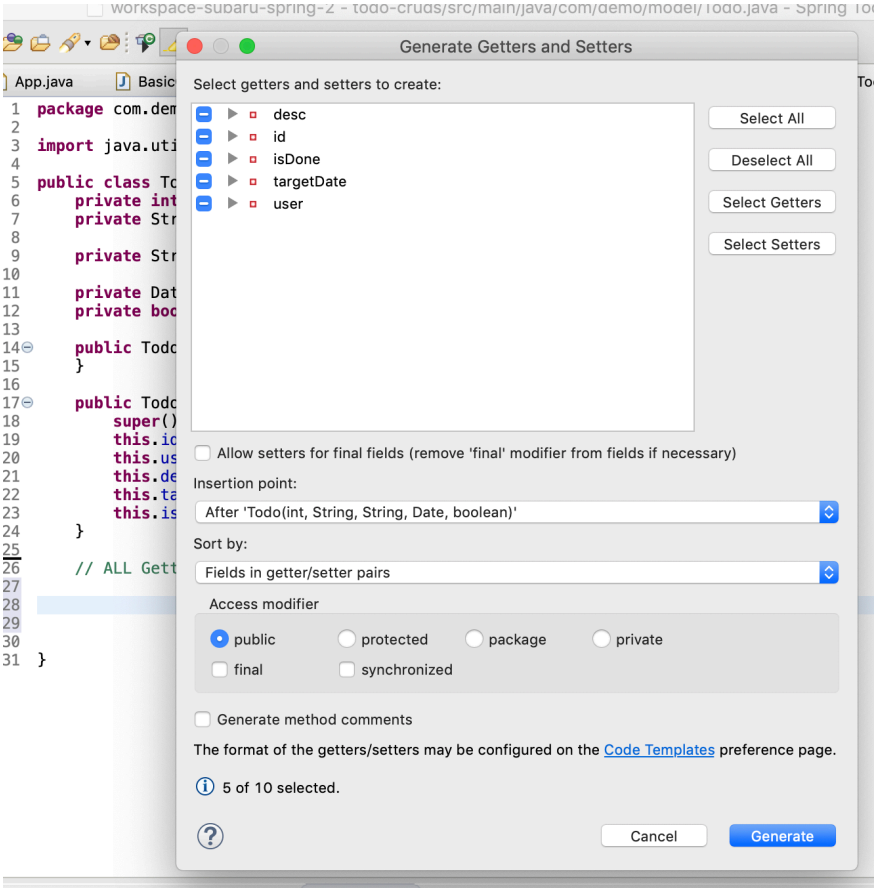
9. Create a “Todo" Model Object

```
import java.util.Date;

public class Todo {
	private int id;
	private String user;

	private String desc;

	private Date targetDate;
	private boolean isDone;

	public Todo() {
	}

	public Todo(int id, String user, String desc, Date
targetDate, boolean isDone) {
		super();
		this.id = id;
		this.user = user;
		this.desc = desc;
		this.targetDate = targetDate;
		this.isDone = isDone;
	}

	// ALL Getters


}
```

10. Generate Getter methods ONLY for all the properties as shown here:



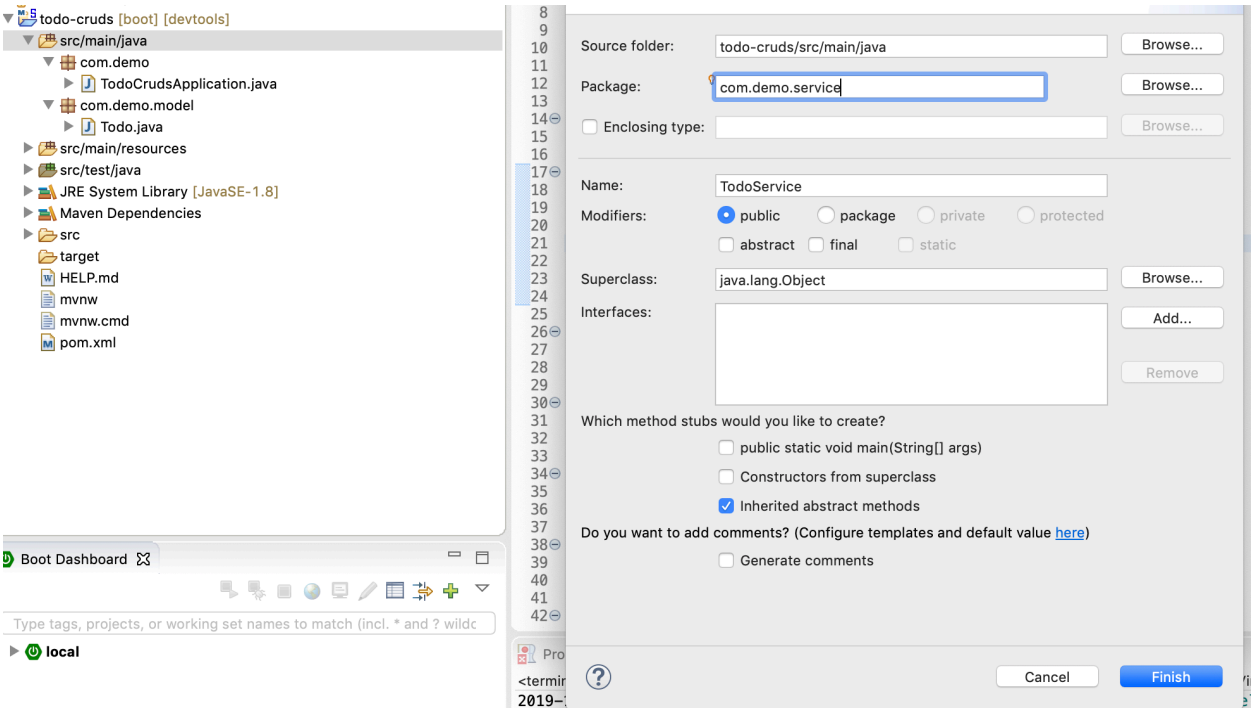
10.1 Click on “Select Getters” Button



10.2 Click on Generate button

11. Generate a Service class to handle all the data operations

So, lets create a TodoService.java



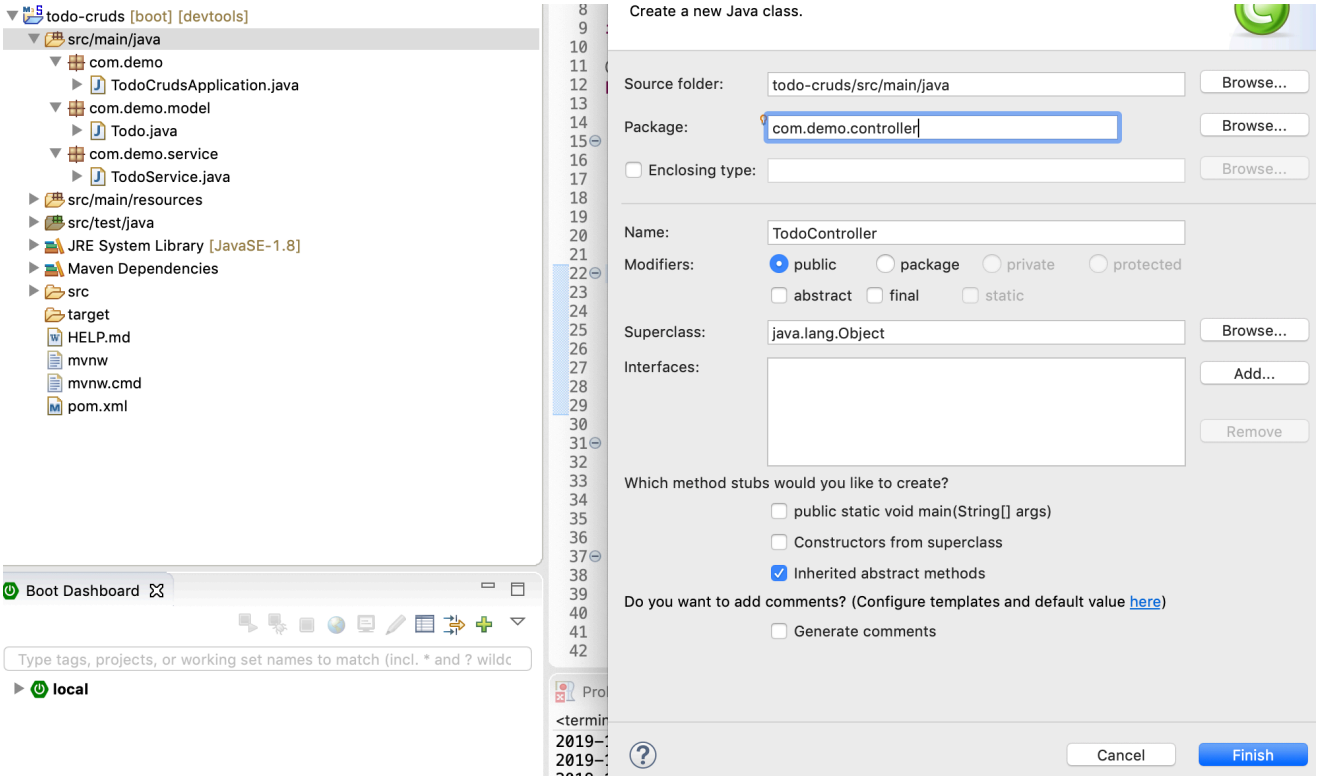
12. TodoService deals with in-memory data and performs all the operations which are connected to appropriate API resource.

So, lets add the in-memory data here:

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import org.springframework.stereotype.Service;

import com.demo.model.Todo;

@Service
public class TodoService {
	private static List<Todo> todos = new ArrayList<Todo>();
	private static int todoCount = 4;
	static {
		todos.add(new Todo(1, "Jack", "Learn Spring MVC", new Date(), false));
		todos.add(new Todo(2, "Jack", "Learn Struts", new Date(), false));
		todos.add(new Todo(3, "Jill", "Learn Hibernate", new Date(), false));
		todos.add(new Todo(4, "Jill", "Learn Spring MVC", new Date(), false));
	}


}
```

13. Retrieving a Todo List

We will create a new **RestController** annotation called **TodoController.java.**



14. Add a service method to return all the todos for the **/todos** API

So, add a method in **TodoService.java**

```
public List<Todo> retrieveTodos(String user) {
     List<Todo> filteredTodos = new ArrayList<Todo>();
     for (Todo todo : todos) {
          if (todo.getUser().equals(user))
               filteredTodos.add(todo);
     }
     return filteredTodos;
}
```

15. Add an API resource method in the **TodoController.java** for retrieving all the todos

The code for the **retrieveTodos()** method is as follows:

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.demo.model.Todo;
import com.demo.service.TodoService;

@RestController
public class TodoController {

 @Autowired
 private TodoService todoService;

 @GetMapping("/users/{name}/todos")
 public List<Todo> retrieveTodos(@PathVariable String name) {
   return todoService.retrieveTodos(name);
 }

}
```

16. Retrieving details for a specific Todo

16.1 **TodoService.java**

Add a method to find a todo

```
public Todo retrieveTodo(int id) {
	for (Todo todo : todos) {
		if (todo.getId() == id)
			return todo;
	}
	return null;
}
```

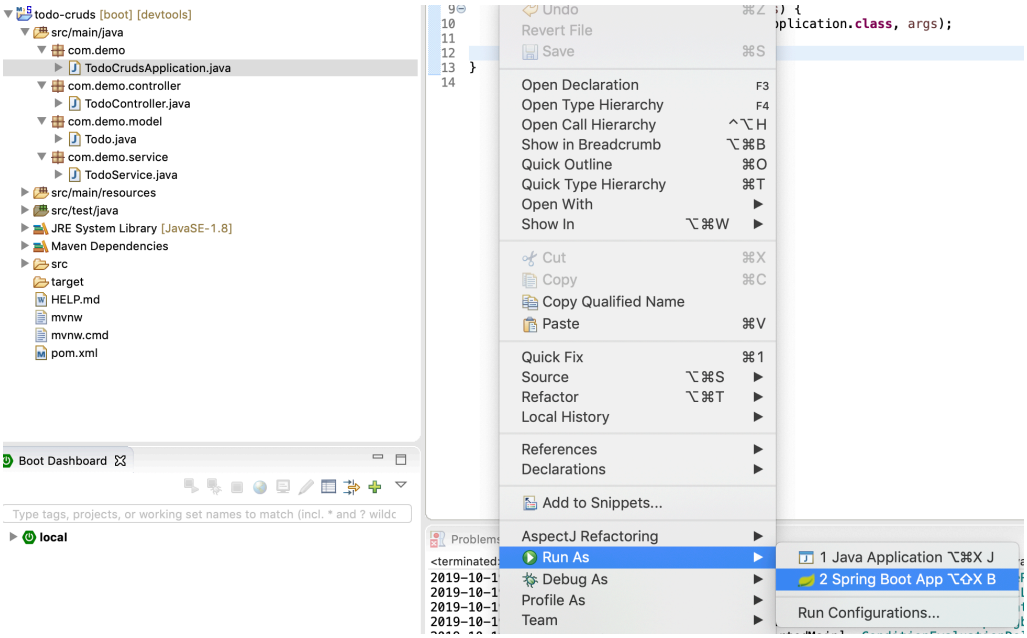
16.2 **TodoController.java**

Add an API resource method to find a Todo based on an ID

```
@GetMapping(path = "/users/{name}/todos/{id}")
public Todo retrieveTodo(@PathVariable String name, @PathVariable int id) {

	return todoService.retrieveTodo(id);

}
```

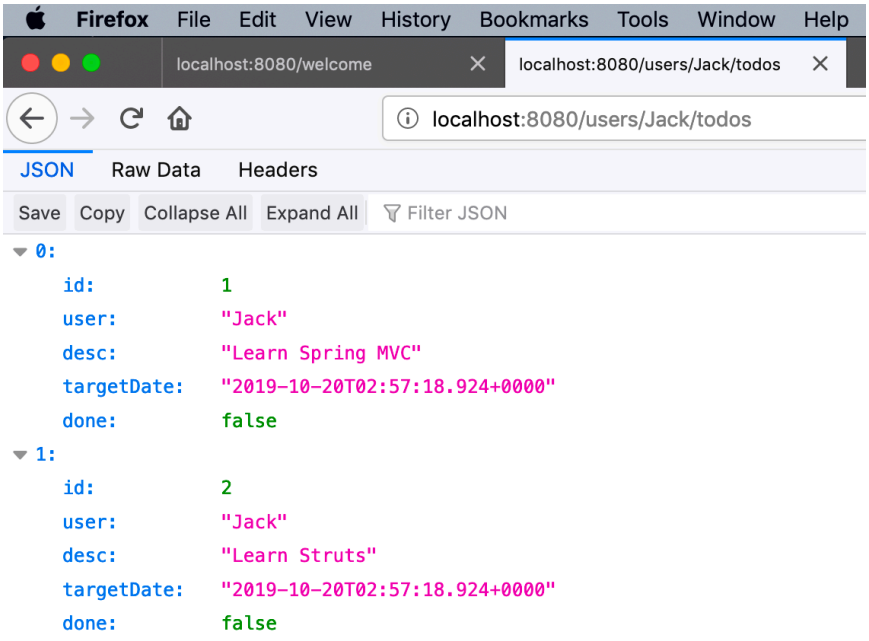
17. RUN the App to test these APIs

17.1

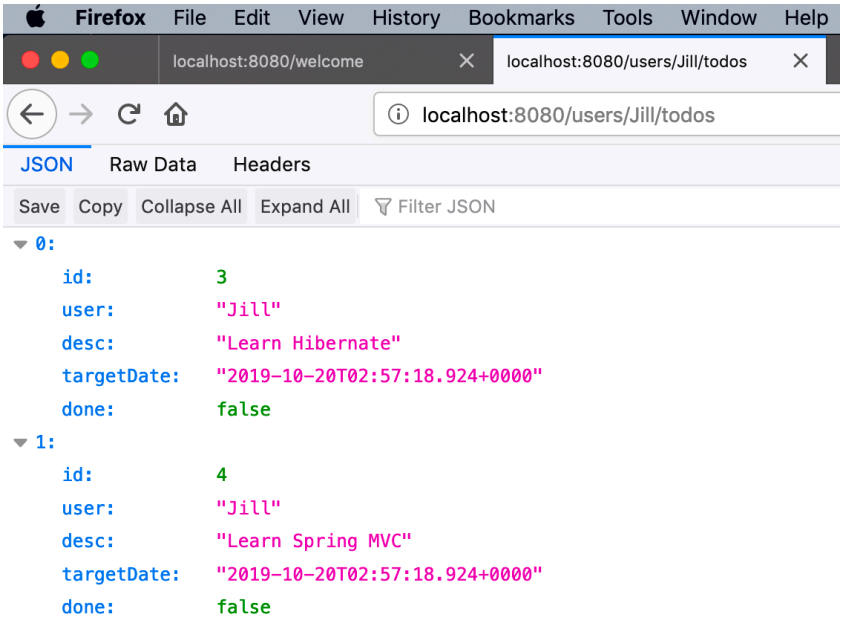


18. Observe the output for the list of todos for each user

18.1 Visit: ***http://localhost:8080/users/Jack/todos***

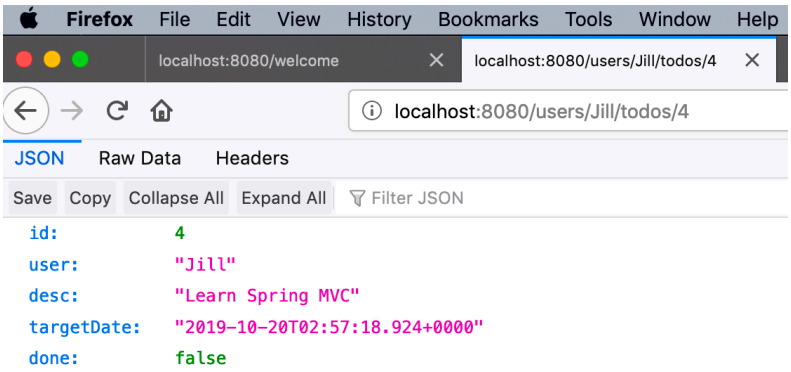


18.2 Visit: ***http://localhost:8080/users/Jill/todos***



19. Find a specific todo task of a user

19.1 Visit: ***http://localhost:8080/users/Jill/todos/4***



20. Similarly, add all the other operations in TodoService.java for respective API resource method

TodoService.java

```
public Todo addTodo(String name, String desc, Date targetDate, boolean isDone) {
	Todo todo = new Todo(++todoCount, name, desc, targetDate, isDone);
	todos.add(todo);
	return todo;
}

public Todo retrieveTodo(int id) {
	for (Todo todo : todos) {
		if (todo.getId() == id)
			return todo;
	}
	return null;
}

public Todo update(Todo todo) {
	Todo deletedTodo = deleteById(todo.getId());
	if (deletedTodo == null) {
		throw new RuntimeException("Todo not found");
	}
	todos.add(todo);
	return todo;
}

public Todo deleteById(int id) {
	Todo todo = retrieveTodo(id);

	if (todo == null)
		throw new RuntimeException("Todo not found");

	if (todos.remove(todo)) {
		return todo;
	}

	throw new RuntimeException("Delete Unsuccessful");
}
```

21. Add all the Resource API methods in the **RestController**

Here you go with all those API methods:

```
@DeleteMapping("/users/{name}/todos/{id}")
	public ResponseEntity<Void> deleteTodo(@PathVariable
String name, @PathVariable int id) {

		Todo todo = todoService.deleteById(id);

		System.out.println(todo);

		if (todo != null) {
			return ResponseEntity.noContent().build();
		}

		return ResponseEntity.notFound().build();
	}

	@PutMapping("/users/{name}/todos/{id}")
	public ResponseEntity<Todo> updateTodo(@PathVariable
String name, @PathVariable int id, @RequestBody Todo todo) {

		System.out.println(todo);

		todoService.update(todo);

		return new ResponseEntity<Todo>(todo, HttpStatus.OK);
	}

	@PostMapping("/users/{name}/todos")
	ResponseEntity<?> add(@PathVariable String name,
@RequestBody Todo todo) {
		Todo createdTodo = todoService.addTodo(name,
todo.getDesc(), todo.getTargetDate(), todo.isDone());
		if (createdTodo == null) {
			return ResponseEntity.noContent().build();
		}
		URI location =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}")
				.buildAndExpand(createdTodo.getId()).toUri(
);
		return ResponseEntity.created(location).build();
	}
```

22. **TodoCrudsApplication.java** –>
Right Click –> Run As –>
Spring Boot App

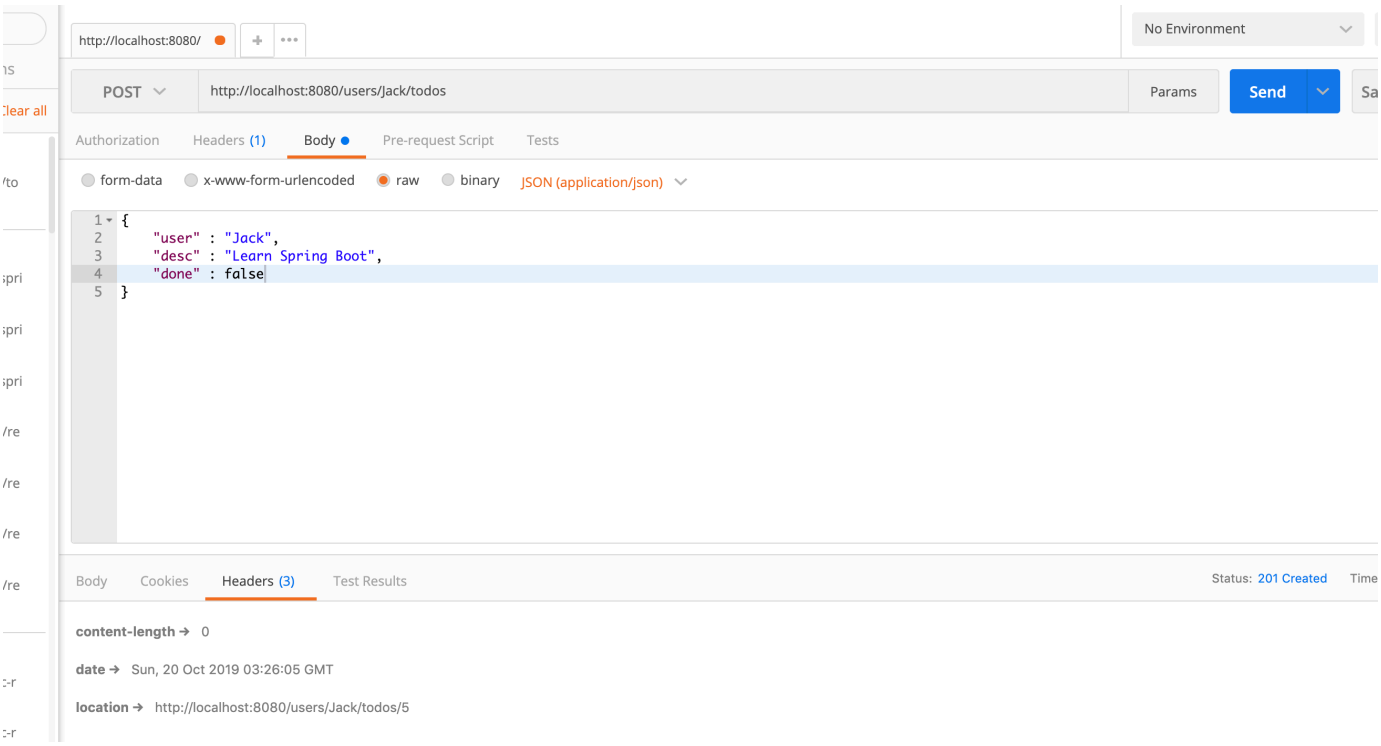
23. OUTPUT

23.1 Create a new Todo using **POSTMAN**

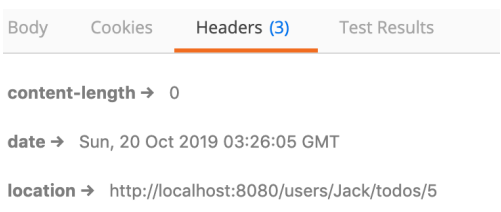
URL: ***http://localhost:8080/users/Jack/todos***
Data:

```
{
	"user" : "Jack",
	"desc" : "Learn Spring Boot",
	"done" : false
}
```

Method: POST



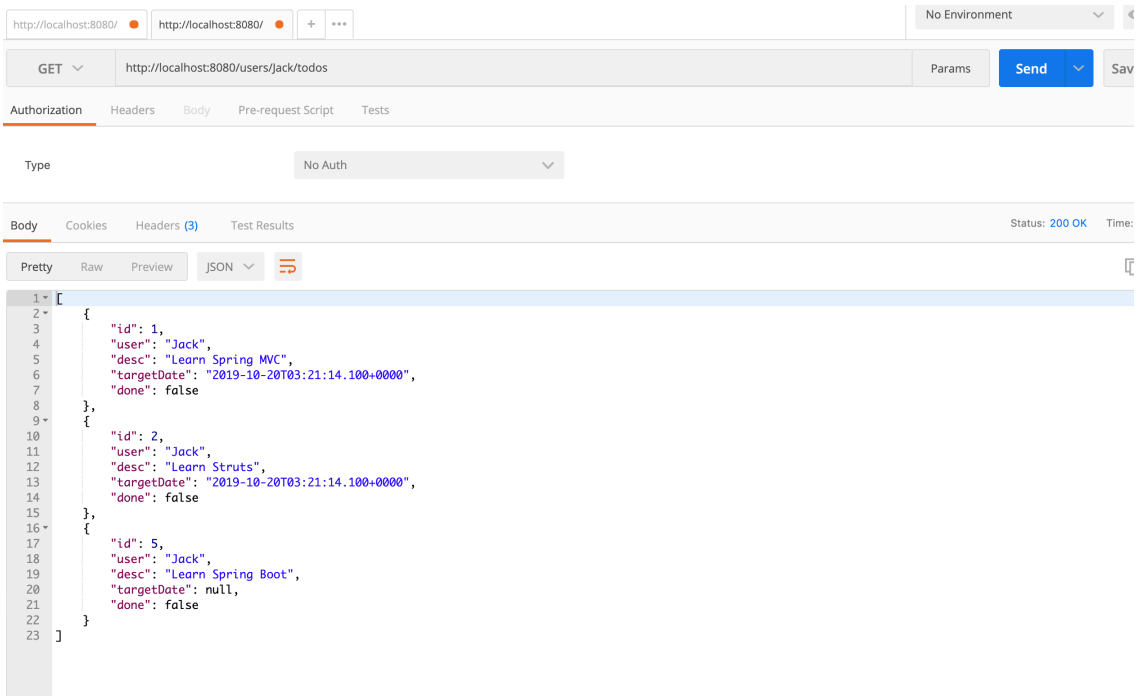
23.2 Click on **Send** button and observe the response in **Headers** Tab



23.3 List all the Todos for a user in **POSTMAN**

Method: GET

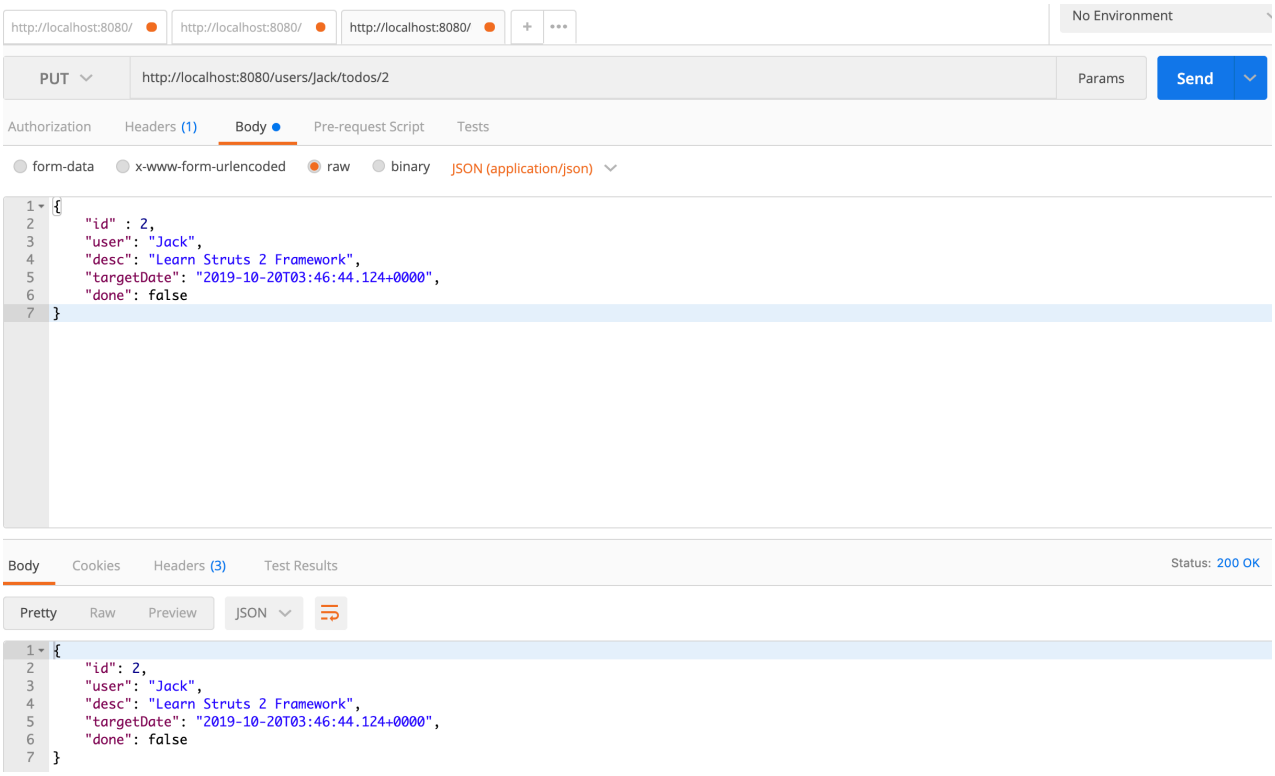
URL: ***http://localhost:8080/users/Jack/todos***



23.4 Edit a Todo task for a User

Method: PUT

URL: ***http://localhost:8080/users/Jack/todos/2***



23.5 Add a Todo Task with Date