**Project 3** – Spring REST, Spring Security, Spring LDAP, Apache POI, JQ Grid, ServiceNow Integration

**Configuration:**

1. For this project we will be using the same skeleton structure from Project 2. Use the Project 2 structure posted in the Course Content area on Blackboard called “**Project 2 - House Disburse Examples**”.
2. Add the following lines to the dependency section of your **pom.xml**:

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>5.1.4.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-ldap</artifactId>

<version>5.1.4.RELEASE</version>

</dependency>

<dependency>

<groupId>org.thymeleaf.extras</groupId>

<artifactId>thymeleaf-extras-springsecurity5</artifactId>

<version>3.0.4.RELEASE</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-webflux</artifactId>

</dependency>

<dependency>

<groupId>org.projectreactor</groupId>

<artifactId>reactor-spring</artifactId>

<version>1.0.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>4.1.2</version>

</dependency>

<dependency>

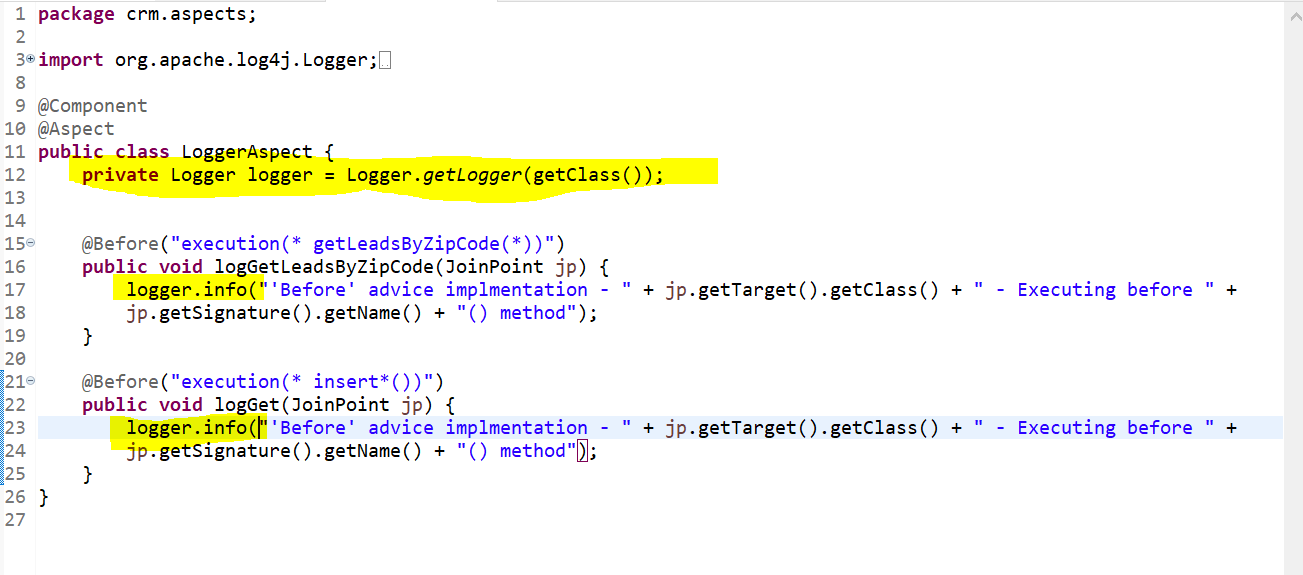
<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.1.2</version>

</dependency>

1. When asked to use log4j, please place the included file “**log4j.properties**” file in your **src/main/resources** folder. To use the logger, you may use **logger.info** as shown below, and initialize it like in line 12 below.



1. **Code Challenge**
2. Create a basic ThymeLeaf named “**signup.html**”. Be sure to use the basic ThymeLeaf structure that includes the normal tag library and security namespaces. (**Hint:** See the first lines of code in the properties of the html tag)
3. Put some basic text on this page in a <div> tag that says “Sign up here!”
4. **Code Challenge**

Create a basic ThymeLeaf page with the same namespace requirements as the previous.

1. Call the file of the page “**logoutRedirection.html**”.
2. Put some basic text on this page in a <div> tag that says “You have logged out!”
3. This is just a shell of a page. We will be re-directing to it later.
4. **Code Challenge**

Secure the Website using Spring Security.

1. Demonstrate in code an external LDAP method using the forumsys and the “in-memory” Spring method. **Hint:** See the demo project. For the forumsys method make sure to look at the **application.properties** file in your **src/main/resources** folder!
2. Permit all unauthenticated users to go to “/login” and “/signup”. Keep your code to default to the “in-memory” method by marking ldap enabled false in your **application.properties** file.
3. Specify a logout redirection to page.
4. **Code Challenge**
5. On the “**disburse.html**” page make a static link with the text “Logout” (preferably in the upper right) with an href of “/logout”.
6. Trigger the page and verify when clicking this link, your app redirects to the logout redirection page.
7. **Code Challenge**

Throw an **AccessDeniedException** if the “**USER**” role attempts to access any method that has a name that begins with “getHighestAmount”. You must use Spring AOP.

1. **Code Challenge**

Log all users’ names with log4j who access methods annotated with **@PostMapping**, **@GetMapping**, and **@RequestMapping** annotations. You must use Spring AOP.

1. **Code Challenge**

Log with log4j any User’s name and return values from any method which executes that begins with “getHighestAmount”. The return values from these methods return List<HouseDisburseDetail> types. Only log the “office” fields in each HouseDisburseDetail element in the ArrayList. You must use Spring AOP.

1. **Code Challenge**

Create a web page called “**createIncident.html**”. Use some basic text in a <h1> tag that says “Report Incident”. Underneath this create a form that includes a textarea and a button that submits an action to a URI “/submitIncident”.

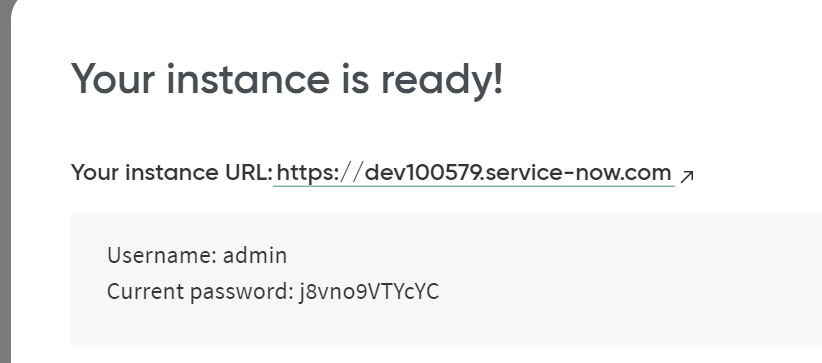
1. **Code Challenge**
2. On the Controller side consume the contents of the textarea from the previous code challenge into a String.
3. Check and submit the form from the previous code challenge (on the **createIncident.html** page) to see if the previous code challenge is working. I want you to at least INCLUDE YOUR NAME when typing the details in the textarea.
4. **Challenge**
5. Log into ServiceNow by going to the following link: (*This link will expire in about 30 days, but you always have the ability to create your own free developer instances on ServiceNow. Sign up here:* [*https://developer.servicenow.com/dev.do*](https://developer.servicenow.com/dev.do) *)*

<https://dev100579.service-now.com/>

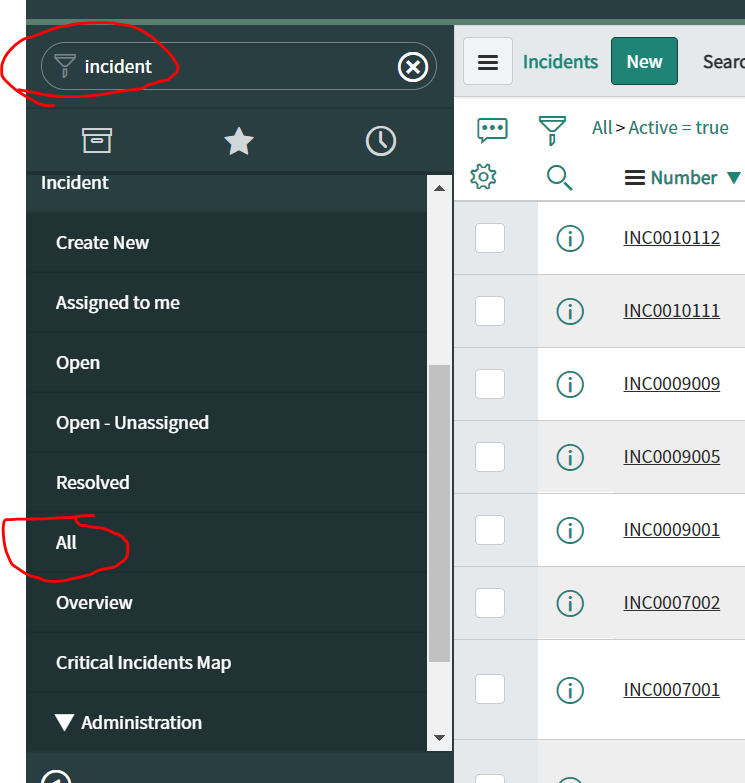
1. Log in with the following credentials:

**u**: admin

**pw**: j8vno9VTYcYC



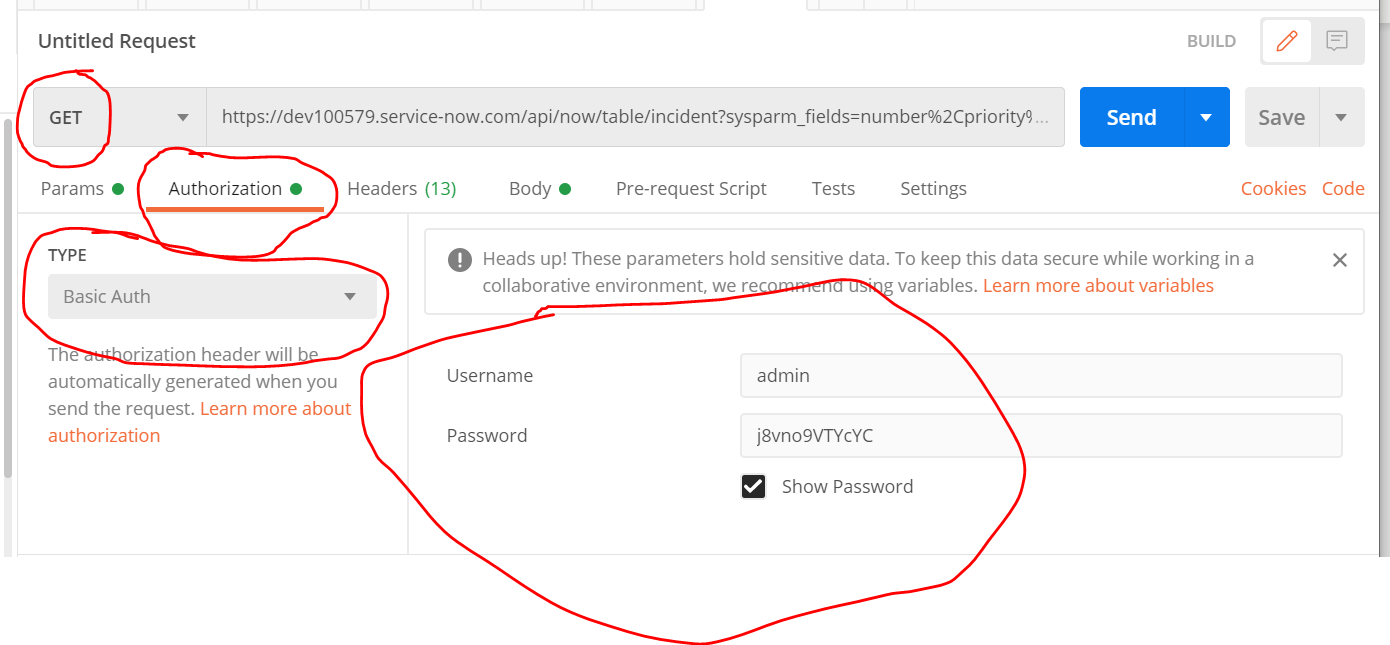
1. In the top left search box type the word “Incident”. Then scroll down in the left pane scrollbar and click “All”.



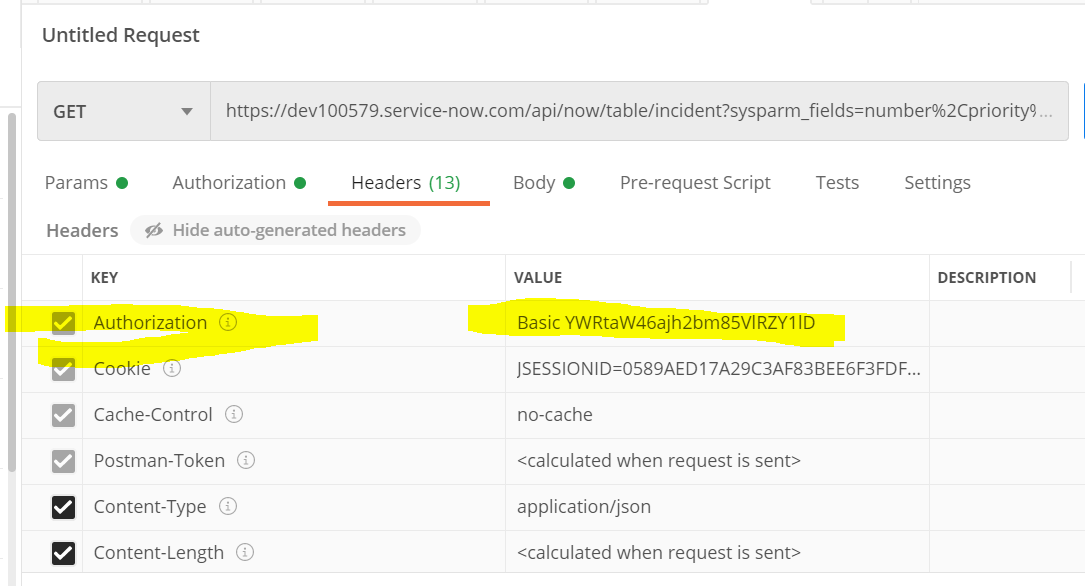
1. Notice the incidents that appear in the right pane. Click a few to see what it looks like.
2. Open Postman. Form a GET request. Use the following URL and the following settings (see screenshot) to make a REST request:

<https://dev100579.service-now.com/api/now/table/incident?sysparm_fields=number%2Cpriority%2Cshort_description%2Ccategory&sysparm_limit=10>

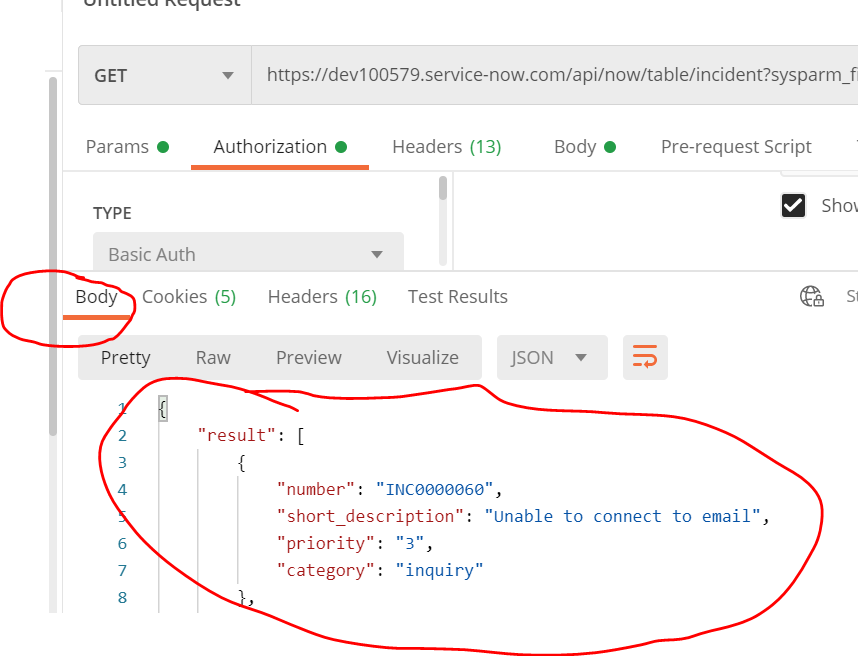
1. Select the “Authorization” tab. Change the type to “Basic Auth”. Plug in the username and password. These settings should look like the following:



1. Click on the **Headers** tab and take note of the Authorization header. Notice the “Value” column, and the string after the word Basic is your “username:password” encoded in Base64.



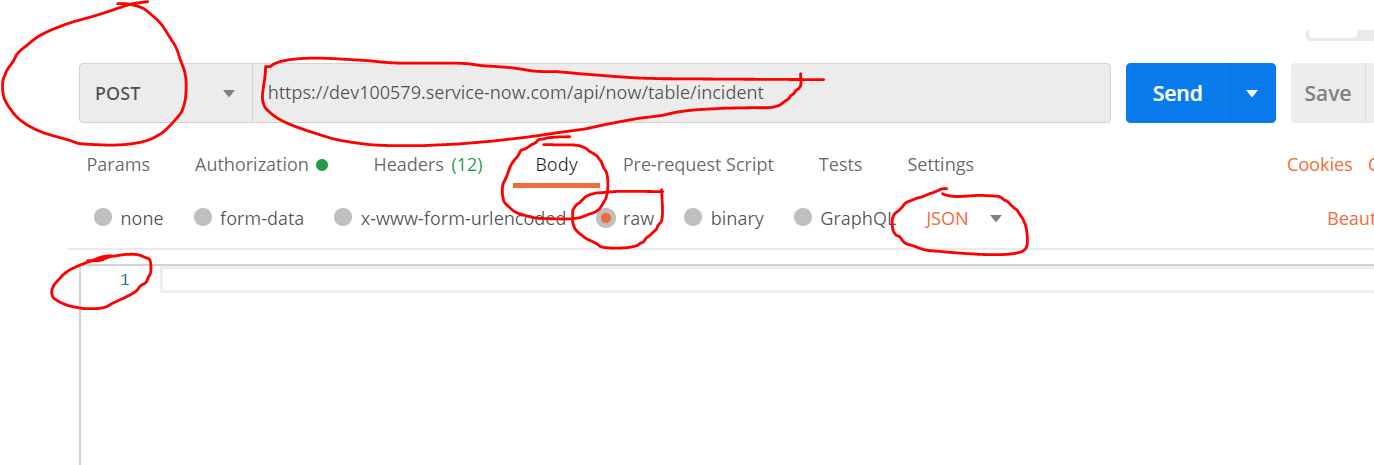
1. Click **Send**.
2. Pay close attention to the Response Body **STRUCTURE.** (After all -- this is why we used Postman to begin with! We need this information to consume the data!)



1. **Code Challenge**
2. Go back to Postman. This time form a **POST** request. Use the following URL:

<https://dev100579.service-now.com/api/now/table/incident>

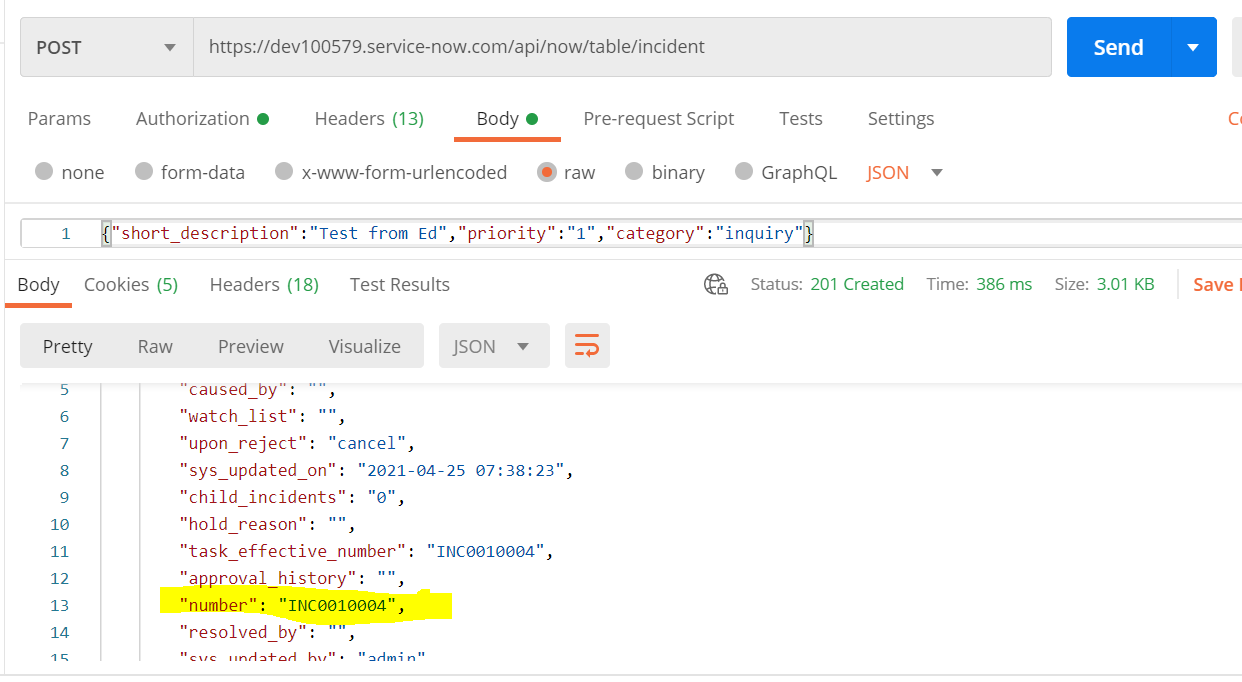
In the upper section click “**Body**” , “**raw**”, and change the dropdown on the far right to JSON. **Clear anything that might be in the Request Body window from a previous project**. Your screen should look like the following:



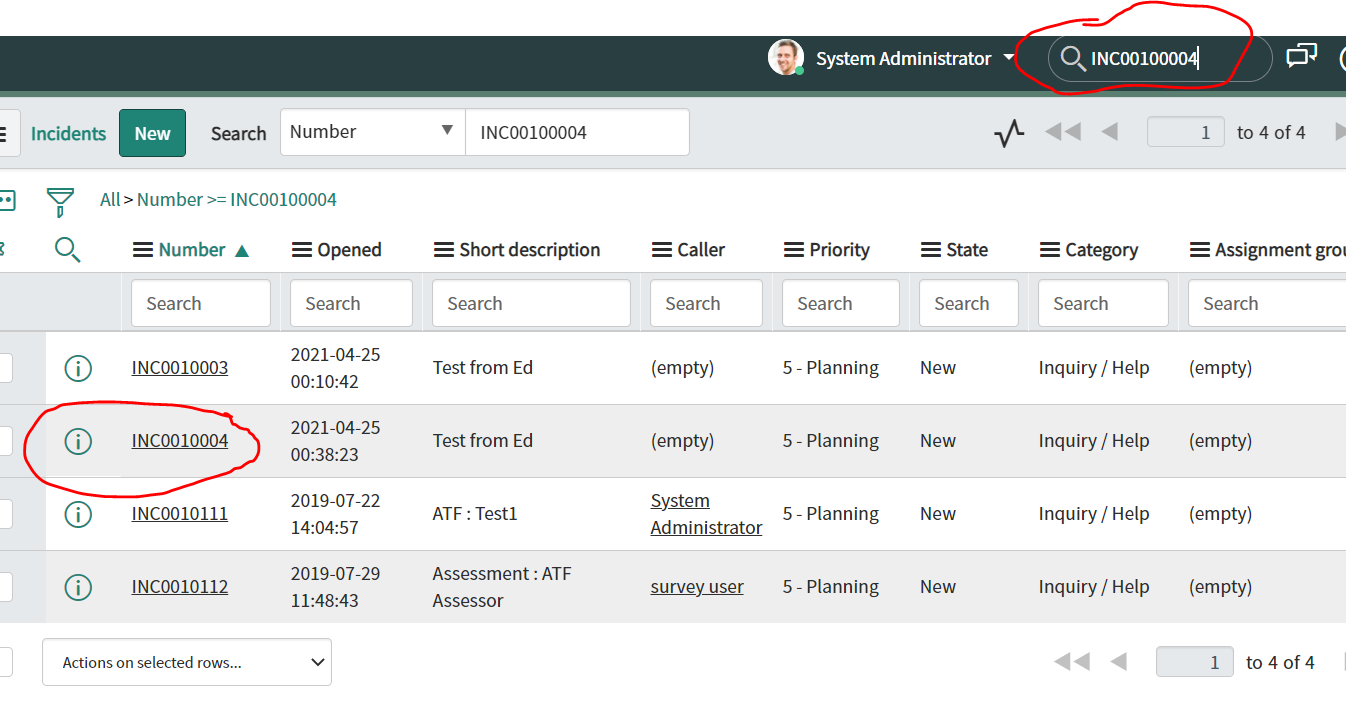
1. Paste the following request body. Change **“Test From Ed”** to “Test from..” your **full name**. This will create an Incident in ServiceNow. Part of your grade will to see if this was done.

{"short\_description":"Test from Ed","priority":"1","category":"inquiry"}

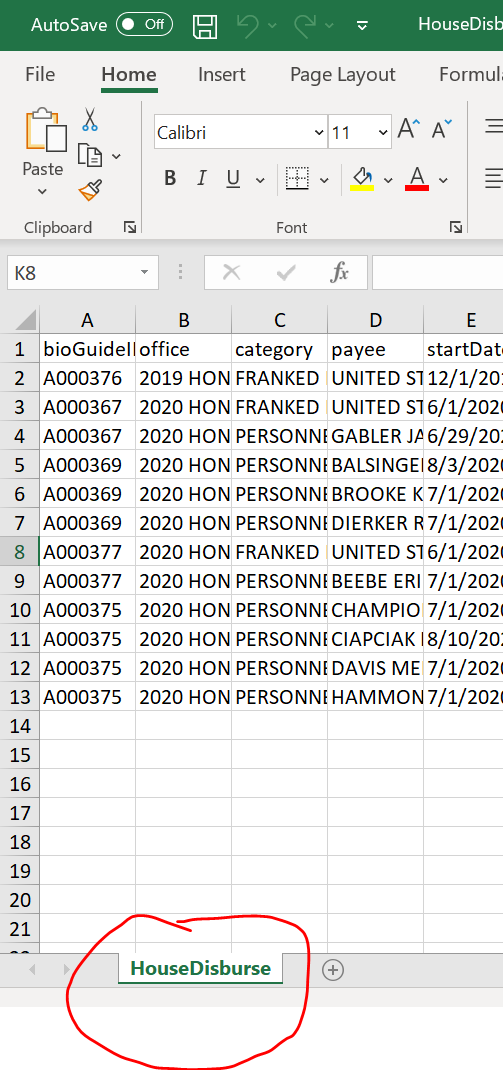
1. Click **Send**
2. Look at the lower Body window (This is the Response Body)
3. Look for the number that begins with INC. See below:



1. Return to ServiceNow. Type in (or copy/paste) this number in the field shown and press Enter. The top magnifying glass will take you directly to the record. Observe the values.



1. **Code Challenge**
2. When the user clicks the “Submit Incident” button on the **createIncident.html** page, create a Spring REST POST request using reactive webflux to post an Incident in ServiceNow.
3. Return the **Incident Number** from the response back to the web page (this is the **INC** number as shown in the previous Postman screenshot).
4. Display the result in a toast message.
5. **Code Challenge**
6. Add a button on the **createIncident.html** that says “**Export all Incidents to Excel**”.
7. **Code Challenge**
8. When a user clicks the “**Export all Incidents to Excel**” button - Use a GET request and return all Incidents in an .xlsx document.
9. Write a header line that uses the same column names as shown in the GET request response we did in the Postman example in the first Postman challenge.
10. Name the sheet “Incidents”. Hint see the pic below. The circled red is the name of the sheet.



1. Name the file “Incidents\_TODAYS\_DATE”. Replace **TODAYS\_DATE** to today’s date. So I want to see for example as a file name: **Incidents\_4-21-2021.xlsx**
2. **Code Challenge**
3. Create a web page called “**uploadToDB.html**”.
4. Provide a form with a submit button on the webpage to upload a file.
5. Add a method in the **HouseDisburseController** to consume the file.
6. Upload the included **HouseDisburse.xlsx** and insert the rows in the Excel sheet as records in the database.
7. **Code Challenge**

Transform the existing page at the **disburse.html** to display results in a JQGrid instead of a <table> tag.

1. **Code Challenge**

Adjust the JQ Grid to be able to edit a record inline on the grid and then by pressing “Enter”, the record will be saved into the database. When the save is executed, fire a toast message stating the record has been saved successfully. (**Hint**: This behavior is demonstrated exactly in the SpringMVCDemo project. See the “**onSelectRow**” js function and “**aftersavefunc**” function to properly set the parameters to do this.)