Report

Introduction

In this document I will be discussing the application created with relevance towards the design patterns used in the application as well as the techniques for the application in which it should follow i.e design patterns and style of coding to ensure that the application is meeting the enterprise level of code to ensure that the application fits the brief but also contains the correct elements to make it a fully fleshed out application.

Design Patterns

MVC

One of the design patterns that has been used in the application is the MVC pattern, this is essentially a way of splitting up the application in to manageable sections, so that each section provides a certain service so MVC stands for model view controller. The model will essentially maintain the structure and the data potentially needed in the application, so in my application the model Film will provided the information for the films with the relevant information like getters and setters for each variable method, so title will have a deceleration as well as getters and setters for the application this also applies to another class like FilmDAO which will model the requests that i need preforming like listing all the films will always do the same thing in each project i need, so the model will have the main data making it module and therefore reusable like in the other projects developed for the assignment (Wattis, 2017).

The view section is the main point in which the user will interact with the application, so web pages like the JSP or html files will be the view elements of the application setting the structure for the data from the model and using the controller to show the logic. An example in application developed index.html will show the buttons for the ajax request for text, XML and JSON as well as showing the results from that request on the screen or the GUI

implementation of the CRUD actions the table and SQL formatting is due to the view handling the request for my application and displaying to the user (Wattis, 2017).

Then finally the controller this is an element that will deal with the main business logic of the application. This will essentially bring together the elements from the model and then pass this over to the view but the controller and all the elements are very literal, as they do as they say an example could be that you want to list all the films, so you perform the request in the URL if the controller was not implemented that request would lead to nothing but due to it acting like the middleman and dealing with the request for the model get films method. You then attribute this to a certain action from the view which is a URL request and then pass that information back to the user with the information they have requested (Wattis, 2017).

When discussing why this design pattern is a good idea in application it is constructed in a way in which the components make for a more improve software development experience from both the user side and the developers side, as from the users perspective this will speed up the process of getting the information as having this all in one file would increase loading times and lead to less users (Wattis, 2017).

From the perspective of the developer this is a massive pattern to use as this ensures that when working in a team you have a readable and more streamline application so that any developer regardless if they have seen the pattern perform or not they can gauge what is happening in the application (Wattis, 2017).

Another point is modular design this may seem like a novelty in software as you can copy and paste elements which is true, but this leads to more errors in the code due to naming conflicts missing imports or incorrect implementation but with the ability to have each aspect in its own class for that purpose it means less writing of code, as you have done the majority of the implementation already. This is very applicable to my situation as areas such as the search element uses from the list all films method implementing the same format output and using the same method in the model made for a simpler experience when implementing (Wattis, 2017).

Finally on why use this in my application is that it follows a tried and test techniques that is not just used in java but in other software development techniques, so having this used ensured that it is a reliable pattern and effective due to its longevity in web based applications but also documentation for this pattern is used vastly in tutorials and documentation for the application that implementing. This was rather simple as others are using the same pattern in which can be better understood when trying to implement a library into my application (Wattis, 2017).

Connection Pooling

One of the design patterns used within the application is the connection pooling design pattern, this is a way of handling connection between a user and a database this type of pattern will essentially deal with caching the connection, as when dealing with the connection from a database it will be a very resource intensive action due to the constant opening and closing for simple request like updating or deleting, so having this implemented is a very useful tool to have implemented as this means the application will no longer take resource which could be used for other applications (A Simple Guide to Connection Pooling in Java | Baeldung, 2019).

The way that I have implement the connection pooling is by defining a separate file called datasource then calling the datasource config property to a variable, in which i can then call certain properties that are required like the driver and the user information to actually connect to the database but having other configuration settings is vital, for improving certain database features like setting timeouts and pool size. The best way to compare connection pooling is that in Progressive web apps you cache the resource that you will be using, so this will ensure that this will store the connection of usage although it may not be for offline use it speeds the process up. As this information can be accessed from the cache rather than keep requesting the data then i have called the config data into a new datasource variable with a method that will return the connection then once this has been used it will called in the try and catch statement for each request so that it will preform the action in that method to the servlet (A Simple Guide to Connection Pooling in Java | Baeldung, 2019).

The reason that i have implemented this in the application is that due to this being a web application the resources and servlets need to send the data in as little time as possible with low over head. This will ensure that this will happen as many user will find that if a connection is slow or preforming odd then having. This will ensure that it can be dealt with in a manner that is better for the user as they can access the data quicker but from a developers point of view it will give the data a more streamlined look and use techniques that are being used in industry making it a perfect choice especially since it separates the connection from the rest of the FilmDAO ensuring a connection is optimized and reduces load times for build times (A Simple Guide to Connection Pooling in Java | Baeldung, 2019).

Software engineering techniques

Agile

This is a software development techniques used mainly in teams but applies to single users as well, like in this situation but it essentially splits the application it to requirements for the application. If you are baking a cake it will have a recipe list this is essentially it. This technique ensures elements are done in the correct order like baking you cant start at step one the go straight to the last step as elements are required to get the action working. The reason it is used in teams is that when giving developers tasks to perform they can tackle a section each to then reduce the load on others but to speed the process up this will usual be shown on a board, which shows what needs to be done if it has been complete or requires further action this is a process i have used using to-do list to preform these actions (Denning, 2016).

The way that i have preform the agile method is by using a to-do list with all the needed element for the application to be complete, so this includes required resources as well as coding implementation not yet finished. This method help organize and structure the application process as this means that i can focus on element for the application that need completing so it can get other elements working to complete the application (Denning, 2016).

Scrum

In terms of the scrum techniques this is a very useful technique, as this will tackle the problem in a group fashion and by having meetings with the scrum master you will give informative and collection of tasks that have been preformed. The main issue that scrum presents in my situation is that it is directly aimed at a large group of developers who are preforming sprints which are short couple of week burst to complete a software feature for the application so this includes add and fixing the implementation for the application (What is Scrum?, 2020).

The main reason this is something not used in my application is the timescale would be far too long to get this complete as getting a feature complete with a long time frame is not practical especially, when as a lone developer that needs to juggle. Many different tasks all at once that limiting it to burst to complete actions would result in a application and methodology that doesn't suit the project i'm developing (What is Scrum?, 2020).

Style of code

Camel casing

When looking at camel casing within the application i have used this extensively in application areas, so this includes the areas like variable names so within all the servlets i have a FilmDAO object called filmDAO which then calls upon methods that use the same convention so update film method is called updateFilm which future sticks to the naming convention and structure within the application (Yeri, 2019).

The reason for the use of camel casing is that when creating an application it is best to follows certain convention for naming and structure of methods and variables. This is something i have done but where possible so if i have a variable that is a one word name then it will not have camel casing, as there is nothing to camel case the variable name but those with more than one word need this to differentiate whether or not a name is that is says it is. As two words could have an ending character and start character the same letter and if they are lower cased together then reading this name can be confusing to the developer hence why in my case this was used within the application but it has not be used in classes as they require capitalization for the beginning character but also on another note they still stick to a naming convention that makes sense while sticking to camel casing structure (Yeri, 2019).

Indenting

When looking at the indenting convention used in certain software techniques it is clear to see that this adds to the applications readability and general understanding of methods within the application. The usage in my application is throughout and used when necessary so within the class it will have an indentation like in getallfilms method in the FIlm DAO then it will contain at the top variable definitions then a try and catch statement which will indent the code within it (Pal, 2020).

The reason for the use of indenting is that readability in the code is essential for create and understanding the application as many lines of code amount of code used would be hard to make sense as text can be lost. As developers will not see certain areas so have areas separated is very useful techniques that can be used within the

application or even software development, as a whole but this doesn't mean you should over use this technique as having to many indents where not needed will make the whole point of readability and understanding null and void as you cannot properly make out areas in the code which thankfully this is something i have not managed to do (Pal, 2020).

Naming Conventions

When looking at the software engineering techniques used in the application all the variables methods and classes stick to a naming convention that is rather simple to grasp, so in the application the the class FilmDAO is telling the users that this contains the Film database access object and the Film class contains Film getters and setters. As they do a job different to one another they have names to match this and the same is applicable to methods so the getAllFilms method will get all the films from the database and the delete films will delete the films (Malik, 2019).

The reason for using a naming convention is that when looking at the application at quick glace the development should not be hindered due to the fact i cannot remember the name for the method or variable. This will speed up the process of development but also when working in a team of developers you want it, so that when they look at the code it makes sense which each section is doing so that there is not any confusion when implementing other methods for the application that use these pre-made functions (Malik, 2019).

Readablity

When talking about the readability of the application there are many aspects that prove this to be a fact that it is very readable., so as method above using the naming conventions indenting and camel casing but these aspect make it read well but there are other aspects such. As the the flow of the application i.e why have something like the open connection method for a database at the bottom on the class as everything proceeding that will use this method, so logically it makes sense to have it flow within a way that makes sense when reading it this also applies to other elements like global variables in the class define this at the top as the content below will be using it. I believe i have done this so in the application i have made sure elements are logical in there ordering so the getnext film method which is used in all the methods for the crud and search operations is near the top and the methods using it will be below it.

Another readability element used is formatting in a sense where if i have a statement what takes multiple values i have made sure to but them on a newline, so that it is readable from a glance an example is in my get text JavaScript file i have made sure that the application when it is running the ternary operators in on one line for that element with the next element being on the next line. As this mean when looking at the code it is not on one long line with similar content that can be confused with other elements that are so similar to one another so in this aspect i have made sure that this is applied to the application.

The last readability point is using industry standard naming and techniques, so what is meant by this is that stick to a format that everyone else would be used to if another developer needed to look at my application. For example in my for loops i have made sure to use the i variable for the iterator value, as this is standard practice within the development field but like in the JavaScript file get text which required two for loops i have made sure to use the next letter to ensure that some consistency is maintained in the application.

Maintainability

When discussing the maintainability for the application it is clear that most element can be easily maintained, as the application is a very modular setup and as a result of this it means that it can be maintained in a way that if you don't need a certain aspect you are able to remove it making it a nice implementation. This applies to the methods as they preform the role like creating a record, so you don't have to worry about editing vasts amount of code to get the aspect you want it is clearly shown what it is and as well due to this separation of methods this means that if you are trying to adapt the code for a different use case like a blog instead of a film database. This means that instead of struggling to adapt it you can simple change the names to the desired application, so instead of film and filmdao it could be blog and blogdao making it a very nice system to implement in the application, as this can be reused in different aspects this applies to adding more code as well making it a versatile project that if you wish to scale up or back it will work seamlessly.

Another point on maintainability is due to the heavy usage of API's the application will always contain the information you needed without worrying about updates to certain libraries. As they have be imported from that specific version so that since this is all stable and reliable libraries this mean the application will always work but if something needs changing like a newer java SDK it should all be backwards compatible, so issue such as this will not be an issue. This also applies to the self written code that is used in conjunction with certain element as this uses a simple

infrastructure so the filmtolist class will uses a simple implementation to convert the data to a list that is in XML format, so this means if you are to use this with other aspect it will work seamlessly with each aspect due to the simple implementation using different API's.

The last point is the deployment and host of the application due to the usage of cloud solutions such as app engine and cloud SQL this means that the application will be maintainable from the a location that is always accessible. This means that maintaining aspects such as error logs or statistical information on the application in a way that maintains that the application will stay online and the service that will always ensure the application is dealt with in a manner that is safe and supporting for my service without having to setup servers and databases for hosting and maintaining.

Options

Using a cloud database

In this application i have used the cloud database known as cloud SQL, this is a google cloud service and this is used to store SQL data on the internet this type of connection enables the database to be accessed from anywhere using this service with a simple interface and tools to get important information. The main disadvantages that cloud SQL presents is the needed pricing plan to get the database operational, this means you need to input card information which other service do not have to like IBM cloud DB2 and unlike DB2 you are unable to manage the data on the internet requiring a connection using a GUI interface to get database data that shows a lack of data manipulation and more hoops to get working where as DB2 from IBM included so much on a free tier like finding and monitoring the SQL (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020) (IBM Cloud, 2020).

Where cloud SQL shines when using the credits included you have full access to the database tooling such as the integration with the cloud platform from google, as well have support into many different software stacks with vasts amount of documentation that fit the needs of my application, where as services like DB2 limit certain functionality like connections to the database that result in it being unusable for my application and other do not offer better functionality with its documentation making for the development a much simpler experience (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020) (IBM Cloud, 2020).

why

The main reason for using cloud SQL is the simple integration with app engine that makes using the database simple and made the learning curve for getting up and running a simple experience that i was able to get working in a very quick manner but areas could be better like the free tier support and the ability to do more on the web interface that other providers have even in the free version (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020).

Using a cloud platform

In my application i have used a cloud platform for hosting the application this is done using the google app engine on a free tier. Which for a free service is a very useful way of getting up and running on the cloud and web hosting from a well known company they do many other services like virtual machines data analytics and like the service about cloud SQL hosting for databases. This service was relativity easy to use especially from a deployment aspect all i needed to do was install the plugin for the IDE and from the there configure my app engine account to point to the required location and every time the application needed deploying i press run and it will upload and deploy it and show information in the run-time console something that app engine provided, that was a very useful additions was the console debugging and error section in the interface. It would show where the errors were as well as console information i would have had locally this lead to faster development time and solving as it points out the error in the java code that would not appear when running locally mainly database connectivity issue but it still showed this information (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020) (IBM Cloud, 2020).

This is a platform i would consider as i have tried to use many other alternatives for the this application such as IBM cloud foundry but the setup was not as quick and easy compared to Google's implementation. This also applies to places like Heroku that use a git to upload but lack intuitive documentation for this application plus when trying to integrate a google SQL database it makes sense to stick with the same platform provider for compatibility like in the SQL department is not an issue (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020) (IBM Cloud, 2020).

Although it was not always plain sailing as certain areas such as java version required an older version which is something i did not for see happening due to always wanting the better and newer software this is similar to pricing. Although the free tier is very nice other providers like IBM cloud and AWS provide a much more feature rich tier with

greater functionality that if i did not use the free credits provided elements like the database would not have worked (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020) (IBM Cloud, 2020).

why

Why use cloud platform like app engine well this is rather simple the application i wanted to created need a strong infrastructure that uses a simple interface and a nice starting point for a developer trying aspects out to get things working and app engine has accomplished this, massively. With simple deployment and easy integration with the database platform although it may not offer better free options like IBM or AWS does but it takes the stress out of self hosting a service to be a network admin as well as a developer so the main reason is to make development easier which is the main goal as then i can worry about the application rather than server issues (Google Cloud Computing, Hosting Services & APIs | Google Cloud, 2020) (IBM Cloud, 2020).

Commenting

In this application i have used commenting extensively in the application this will show areas and give a brief description on what the code is doing, so this could be if it returning a value like in many of the methods it will show what is actual doing giving the other developers and myself a better understanding on what the application is doing, so the elements i have implemented this is throughout both in a self written and generated sense.. The self written is generally me explaining the action using the comment tag and writing a message this could be the action but it could also be for debugging purpose so like in the JavaScript file i have ensured that all the console logs are commented with the debug text to show that this piece of code is not essential for the application but for the developer (Sourour, 2017).

When on the topic of generated this refers to JavaDoc and code generation, so in many sections of the application i.e the soap project it is generated so classes and the web service to get the application working. The main commenting in the application is commented based as this is something provided by the generation tool as some user will want to know what the application methods are doing, so in soap for each method there is an explanation on what it is that the application is doing but there in terms of generation from the java docs it will layout a commenting system in place with the developer filling in the options for the application, so that you have documentation read when you are developing the file which is something that is very useful when implementing into an application like this as this takes a job out off what the code should be doing in a java registered format that all developers will be able to understand (Sourour, 2017).

API's and Self written

When looking at my application you can see that i have used many different API's for the development but also i have developed elements in my application that would only fit my certain needs so this section will aim to show this in my application.

API

In the application the API's used are extensive enough to justify a section on the use in the web application it may seem that the application would have been better implemented if the application used self written code, but when looking at the application. I have only used them to solve a problem that requires speed and reliability and what is meant by this is that API's have been developed and used in other application with several developers contributing to the API's used, so the reliability will be taken care of as they will find ways to streamline the performance by seconds, whereas if used self written code in areas then it would result in an application that slower and possible incorrect implementation that the API is providing. On the topic of speed this will speed up the process as not worrying about if code will work is essential when some else has created the same tool as documentation will show how to use the library required and if trying to implement this into the application. My self it will slow down the development that is needed in other areas to perform the main action rather than struggling to create something that is already is use a good example in my application is GSON library i could create this library but if i want the application to work correctly without debugging an element which is simple to implement in that application and reduces the work load on my developmental duties (Using a JavaScript library versus building the functionality yourself, 2018).

In terms of where i believe generation or API usage is necessary is through repetitive takes so IDE's implement an API for the getters and setters with a press of a button like in SOAP, which uses an old technology that requires more time than is necessary with other technologies ie HTTP or REST or like in the ajax request using the API jQuery. Although this could be created using vanilla JavaScript but why try and reinvent the wheel if it has been working fine for many developers with jQuery being multi purpose like many API's are implementing ajax request with a front end framework

to show the data rather than trying to use two separate API's that may not be compatible. As the benefits of having a industry standard and simple coding library will ensure that one it will work everywhere as browser compatibility and tested functions will work out of the box with extensive documentation with this being used on the front end to render the information. In most aspects front end GUI's are necessary as the deal with the vanilla aspects of JavaScript which can be solved with a simpler implementation but in terms of the servlet aspect and jsp i think using standard java and libraries is vital for the application to work on the web as many different self written can effect performance as a slow application will result in less users (Using a JavaScript library versus building the functionality yourself, 2018).

Self Written

The self written aspect of the application is something that i debated in many aspects as having methods from other API's would mean it would work, as intended but having self written aspects for the application meant that i could then personalize the application to fit the needs for the action i would like to preform. Although i have used other API's to implement my method it has been written to fit a certain need for the application, so in terms of justifying the usage of self written is that when looking to implement CRUD and search into my application, i would need to use SQL classes due to the fact this is a library used by several users ad developing something like this my self would be a pointless exercise, as the tool is implemented and would take longer than the brief to implement something like this. So using it with CRUD to then create my own methods for the application is something that was vital as it needs to fit my needs. An example is the search query very few API's would have the method to search then use this within the servlet or the get all films uses result set and prepared statements with an while loop o get the result from my specific database which is in an arraylist i could have used a different application to get the crud implemented like Netbeans uses a generator to get all the elements working but this would require certain elements that would have interfered with other elements like the connection pooling so self written is best for aspect that require them to work within a certain situation (Using a JavaScript library versus building the functionality yourself, 2018).

Which access for a web service

In this section I will recommend which service is better for a certain user and use case this may seem like a simple element to diagnose but many different options could occur that may make more than one suitable for a given task.

REST

When you are dealing with anything web based it is a good idea to use are REST API especially if you are looking into this as being a back end service rather than anything else as for the client you adapt it to fit the REST API rather than changing the back end to get it working since it uses endpoints to send the data using get put post and delete to call upon them this means when calling the crud request it will line up for this type of service but the since this is aimed toward the web it will not suit certain areas like internal application or native application as you only really get useful information when dealing with it when collecting it from the REST API (Using a JavaScript library versus building the functionality yourself, 2018).

A REST API is modular and quite simple to implement this is a big improvement over soap which its is spiritual successor has many element that is a much better implementation like using JSON to store the information thus showing how it is a web paged service as JSON is a JavaScript storage method but soap uses XML which slightly more universal but lacks a better storage implementation as XML should not be for storage it should be to configure or point to certain elements in a more controller format but JSON allows for simpler information without fields with more complex element like embedded JSON and array objects leading to a better and more featured rich data type to interactive with (Using a JavaScript library versus building the functionality yourself, 2018).

SOAP

SOAP is a very old technology it is considered a legacy implementation as this will preform web service request but since soap is a more universal service this mean it will sent it over different protocols like HTTP and SMTP which makes a it a jack off all trades this type of data over HTTP is using XML and it is structured in a way like a letter it uses an envelop XML element to encapsulate the whole data then it but the best use case may be in asynchronous tasks as the lightweight nature means that for tasks like in finance and data collection sources it will work better as you don't have to worry about this type of transaction due to the inclusion of the other protocols it can preform more action in a single instance rather than calling different services to preform this action (Using a JavaScript library versus building the functionality yourself, 2018).

In terms of why soap may not be a useful implementation for the web application is that i do not require this type of request type as this data doesn't need to be collected from many sources into one area or asynchronously and this the default data type is JSON the use of HTTP and rest is more much useful implementation as i don't have to convert a XML to JSON which is a harder implementation that the other way around to accomplish the same task and JSON also is a much lighter data type meaning on HTTP and rest which are already light will not have much impact on the user (Using a JavaScript library versus building the functionality yourself, 2018).

HTTP

This is the method i have properly spent the most time on in the application as this is the main way in which the application has been created the way that this type of technology has been developed using this protocols but the main point if HTTP is to transmit data over the internet using HTTP or HTTPS a more secure version but both do the same action this was vital in my application for the servlets as this meant that when i had deployed it it can be accessed from anywhere on the internet which the others would not as they would use this protocol to get to the required destination (Using a JavaScript library versus building the functionality yourself, 2018).

In terms of compared to the other technologies like soap and rest this is a wrapper for both as they need this service to send and revive data so in actuality this service in my option is best suited for any online interaction where as the rest is web focused and soap is more internal focused they need this to make any of there respective functions work so in that aspect it is useful in so many ways as you can use this protocol to get any data from the respective location but this doesn't mean it is best solution so the standard HTTP is unsecure and lacks newer features that HTTPS which is something i have not used in the application meaning it will result in certain use cases like soap which could be used for banking information or rest for application end points will not encrypt this data (Using a JavaScript library versus building the functionality yourself, 2018).

Coding Proof

In the section I'm explaining what the files are doing in the projects for SOAP, REST and HTTP to give a better understanding to users.

Model

Film

```
package model;
import com.fasterxml.jackson.annotation.JsonInclude;
import com.google.gson.annotations.SerializedName;
import javax.xml.bind.annotation.*;

@JsonInclude(JsonInclude.Include.NON_EMPTY)
public class Film {
   public Film(int id, String title, int year, String director, String stars,
      String review) {

    super();
     this.id = id;
     this.title = title;
    this.year = year;
    this.director = director;
    this.stars = stars;
    this.review = review;
  }

  public Film(int id) {
  super();
  this.id = id;
}

  @com.google.gson.annotations.SerializedName("id")
  int id;
  @com.google.gson.annotations.SerializedName("title")
   String title;
  @com.google.gson.annotations.SerializedName("year")
   int year;
  @com.google.gson.annotations.SerializedName("director")
   String director;
  @com.google.gson.annotations.SerializedName("stars")
   String stars;
  @com.google.gson.annotations.SerializedName("review")
```

```
    String review;

  public Film(String title) {
    super();
    this.title=title;
  }

  public int getId() {
  return id;
}
public void setId(int id) {
  this.id = id;
}
public String getTitle() {
  return title;
}
public void setTitle(String title) {
  this.title = title;
}
public int getYear() {
  return year;
}
public void setYear(int year) {
  this.year = year;
}
public String getDirector() {
  return director;
}
public void setDirector(String director) {
  this.director = director;
}
public String getStars() {
  return stars;
}
public void setStars(String stars) {
  this.stars = stars;
}
public String getReview() {
  return review;
}
public void setReview(String review) {
  this.review = review;
}
@Override
public String toString() {
  return "Film [id=" + id + ", title=" + title + ", year=" + year
      + ", director=" + director + ", stars=" + stars + ", review="
      + review + "]";
}
}
```

This is the Film class that is the application it is using many different tools and techniques within this class this will establish the constructors and the getters and setters for the application.

At the top of the application it is defining the imports and libraries that are being used in the application as well as the package of model the reason for this is that i am using the MVC design pattern. This will essentially construct the application in a manner in which the model will provide a blueprint with how the data should form then the imports comprise of JSON include class which will essentially set the application up so that when calling a JSON object from this class it will give the correct names and data for that corresponding element next is the GSON import this will ensure that each element is correctly called upon when serializing the names for the methods and variables.

In the main body of the application you can see the main film class with a annotation for JSON include which essentially means it will create an object for the JSON if its not empty which ensure some validation for the application and its data within the class is a constructor for the film with the parameters in an integer of id and year and strings for title director stars and review within this constructor method it is calling the super class then defining the variables with the this to get the current variables this constructor will be used for the majority of the application with majority of the methods within the filmdao calling upon this then below is another constructor with id being used the reason for this is that in the update method in film dao it will search based on this data with another constructor using title to preform the same request for the search method each constructor are useful as they ensure that unnecessary data is not called upon when using the methods in the film dao so this ensures that the application is performing how it should.

A section is initializing the properties method above but each section has an annotation calling the serialization name from GSON with the corresponding name i.e the initializing for id will have a serialization name of id this is used in JSON format request and will set the actual names for the JSON output file as the data would show without the correct fields so this ensures that this happens with the application for the JSON formatting.

At the bottom of the call is the necessary getters and setters for the application which will set that data to the correct field or get that data when calling upon the SQL which is all shown in a tostring element that returns a string with the variable inside to show the current value when called upon.

In terms of design patterns you can clearly see that it is packaged as model and the contents are showing that is to be used as template for the controllers in the servlet and this is applicable to certain coding techniques as elements such as naming conventions follow throughout with the getters and setters showing the appropriate names as well as variable names with indentation and necessary API's used when action required thus showing a API's in use working with self written definitions for the application.

FilmDAO

```
package model;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import model.*;
import javax.naming.NamingException;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;




import java.util.ArrayList;
import java.util.List;


public class FilmDAO implements FilmAPI {



  Film oneFilm = null;
  boolean delete;
  boolean insert;
  boolean update;
  Connection connection;
  Statement statement;
  String user = "cbp25056";
  String password = "313+64bnkwc5lx90";
  // Note none default port used, 6306 not 3306
  String url = "jdbc:db2://dashdb-txn-sbox-yp-lon02-01.services.eu-gb.bluemix.net:50000/BLUDB" ;
  String title = null;




  private void openConnection () {
      // loading jdbc driver for mysql
      try {
        Class.forName("com.ibm.db2.jcc.DB2Driver");
      } catch (Exception e) {
        System.out.println(e);
      }

      // connecting to database
      try {
        // connection string for demos database, username demos, password demos
        Connection connection = DriverManager.getConnection(url,user,password);
         Statement statement = connection.createStatement();
        System.out.println(connection);

      } catch (SQLException se) {
        System.out.println(se);
      }
    }
    private void closeConnection () {
      try {

        connection.close();

      } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
```

```
  }
}
@Override
public Film getNextFilm (ResultSet rs){
  Film thisFilm = null;
  try {
    thisFilm = new Film(
        rs.getInt("id"),
        rs.getString("title"),
        rs.getInt("year"),
        rs.getString("director"),
        rs.getString("stars"),
        rs.getString("review"));
  } catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
  return thisFilm;
}

@Override
public ArrayList<Film> getAllFilms () {

  ArrayList<Film> allFilms = new ArrayList<Film>();
  String selectSQL = "select * from films.films";
  // Create select statement and execute it
  try(Connection conn = DataSource.getConnection();
    PreparedStatement pst = conn.prepareStatement(selectSQL);
    ResultSet rs1 = pst.executeQuery();) {

    // Retrieve the results
    while (rs1.next()) {
      oneFilm = getNextFilm(rs1);
      allFilms.add(oneFilm);


    }

  } catch (SQLException se) {
    System.out.println(se);
  }



  return allFilms;
}
@Override
public Film getFilmByID ( int id){

  oneFilm = null;
  String selectSQL = "select * from films.films where id=" + id;
  // Create select statement and execute it
  try (Connection conn = DataSource.getConnection();
     PreparedStatement pst = conn.prepareStatement(selectSQL);
     ResultSet rs1 = pst.executeQuery();){

    // Retrieve the results
    while (rs1.next()) {
      oneFilm = getNextFilm(rs1);
    }



  } catch (SQLException se) {
    System.out.println(se);
  }

  return oneFilm;
}
@Override
public boolean deleteFilm (Film film) throws SQLException {
  String sql = "DELETE FROM films.films where id = ?";
  try (Connection conn = DataSource.getConnection();
     PreparedStatement pst = conn.prepareStatement(sql);) {

    pst.setInt(1, film.getId());

    delete = pst.executeUpdate() > 0;


  } catch (SQLException se){
    System.out.println(se);
  }
  return delete;
}
@Override
public boolean insertFilm (Film film) throws SQLException {
```

```
    String sql = "INSERT INTO films.films (id, title, year, director, stars,review) VALUES (?, ?, ?, ?, ?, ?)";
    try (Connection conn = DataSource.getConnection();
       PreparedStatement pst = conn.prepareStatement(sql);) {
      pst.setInt(1, film.getId());
      pst.setString(2, film.getTitle());
      pst.setInt(3, film.getYear());
      pst.setString(4, film.getDirector());
      pst.setString(5, film.getStars());
      pst.setString(6, film.getReview());

      insert = pst.executeUpdate() > 0;
    } catch (SQLException se){
      System.out.println(se);
    }
    return insert;
  }
  @Override
  public boolean updateFilm (Film film) throws SQLException {
    String sql = "UPDATE films.films SET  title = ?, year = ?, director = ?, stars = ?, review = ? ";
    sql += " WHERE id = ?";

    try (Connection conn = DataSource.getConnection();
       PreparedStatement pst = conn.prepareStatement(sql);) {
      pst.setString(1, film.getTitle());
      pst.setInt(2, film.getYear());
      pst.setString(3, film.getDirector());
      pst.setString(4, film.getStars());
      pst.setString(5, film.getReview());
      pst.setInt(6, film.getId());

      update = pst.executeUpdate() > 0;
    } catch (SQLException se){
      System.out.println(se);
    }
    return update;
  }
  @Override
  public ArrayList<Film> searchFilms(String title){
    openConnection();
    ArrayList<Film> al = null;
    oneFilm = null;

      try {
        al= new ArrayList<>();
        String SQLsearch = "select * from films.films";
        if(title !=null && !title.equals("")){
           SQLsearch = "select * from films.films where title like '%" + title + "%'";
        }
        ResultSet rs1 = statement.executeQuery(SQLsearch);
        // Retrieve the results
        int i = 0;
        while (rs1.next()) {

          oneFilm = getNextFilm(rs1);
          al.add(oneFilm);



          System.out.println("al :: " + al+ i);
          System.out.println(title);

        }
        statement.close();
        closeConnection();
      } catch (SQLException se) {
        System.out.println(se);
      }
    return al;
    }

}
```

In this section it contains the main elements for the application to work it has elements such as the crud and search so at the top of the class is the imports required for the dao this includes SQL all the element from the SQL library as well as the array list within the class i have initialized certain information like declaring the crud methods and connection information within the top two methods it contains information for connecting and disconnecting the connection due to the inclusion of the connection pool this is no longer needed but is left in the class encase this needed to be reused again containing how getting the driver and getting the URL connection for the SQL with the close being called in the close connection and the necessary catch statements for each section.

Next is the getnextfilm method this is using the film model with resultset rs being used as a parameter for getting the data from the SQL database inside this it is defining thisfilm as null to ensure nothing is stored in this variable when

called upon and it is calling the try and catch statement in which thisfilm is creating a new film object that is within this is using rs get on each film aspect required to obtain the data from that specific column and the catch is getting the errors for this method but outside the try and catch but still in the method it is returning thisfilm so that this data is available for the other methods to use .

The next method is the get all film that issuing an array list to store this information within this method it contains an arraylist called allfilms what is creating new arraylist then below is a SQL statement stored in a string that is calling from films.films which is the database and the table then the try statement is using resource that is the definition for the connection pool so the connection variable con is calling datasource.getconnection which is opening the connection and thus a pool for that database then calling the prepared statement with the SQL statement as a parameter and finally calling resultset as rs1 with prepared statement to execute the query method the reason it is in this format and not in the try block as this will open and close the connection based on each method thus creating a pool that will close when it has complete all the actions in the try so next it uses a while loop the takes rs1.next what loops through until it has completed the query within this it calls the onefilm variable defined at the top of the class and it equals the method getnextfilm with rs1 as a parameter this will mean the query is passed to this request next it is added to the arraylist variable allfilms it catches any exceptions in the catch statement then returns the allfilms arraylist as this is then passed to other servlets to use this action.

The next method is the getfilmbyid this will call the same information that getallfilms does with the only exceptions being it takes the id as a parameter in the method and the SQL statement is to find the id in the database rather than everything with the while loop only have the result set variable called instead of the adding it to an arraylist as this will be passed to other methods like update and delete as they require the id with it returning onefilm to pass to the servlets.

The next method is the delete method this is called as Boolean as this means that when calling this statement it will call it as true or false thus reducing the work load on the database and application the SQL is the delete request for a SQL database with the id being passed the try is the same as the other methods within the try it is calling pst variable that is setting the film id by including the index and the film.getid method from the model and below this a delete available defined globally is equal to the execute query that is greater than 0 as this means it is true and will delete the id from the database when calling this method with it catching the exceptions and returning the delete value for the servlets to use for the controls.

The insert method is also a Boolean and uses the same try statement resource with the SQL for inserting into the database but it takes the ? in the parameter as this is requesting the information from the result set with the parameter index set to insert into the correct field in the SQL string so in the try it uses this within a set method for that data type with the next parameter being get the film model attributes for the corresponding element in the SQL statement so index of 1 and film.get id is setting the id for the database and the same as the delete it checks if the query is greater than 0 to see if it can execute the command with a catch to catch the errors and with it returning the insert for the users to access the data from servlets the update method is largely the same as the create except the SQL string includes a where id to update in that particular id elements.

The final method is the search method this will use the title element from the database to get so it defines a SQL statement outside the method along with an arraylist then the try is the same regarding the resource but the al variables is called to create a new arraylist then an if statement that checks if the title variable is not null or not equal to a blank string if this is true it will run the SQL statement where it finds title based on the like method with % symbol used to get it from anywhere before a string start or the end so if it contains this text it will be returned which for a database and the application is a very useful function as this means i can search for the request that could be anywhere in the title string so for example typing in Bronx would return a Bronx tale and rumble in the Bronx as text is begining and ending before it finds the text then the while loop will return the same information that the get allfilms returns but limited due to the SQL query with a system out with the contents for the request data for debugging purposes.

DataSource

```
package model;

import java.sql.*;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import com.google.cloud.sql.mysql.SocketFactory;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import java.sql.Connection;
import java.sql.SQLException;

public class DataSource {

    private static HikariConfig config = new HikariConfig();
    private static HikariDataSource dataSource;


    static {
        config.setJdbcUrl("jdbc:mysql://"+ "films");
        config.setUsername("root"); // e.g. "root", "postgres"
        config.setPassword("MowICs9nJdwL4HDC"); // e.g. "my-password"
        config.setDriverClassName("com.mysql.jdbc.Driver");
        config.addDataSourceProperty("socketFactory", "com.google.cloud.sql.mysql.SocketFactory");
        config.addDataSourceProperty("cloudSqlInstance", "film-web-application:europe-west1:ryan-love");
        config.addDataSourceProperty("useSSL", "false");

        config.addDataSourceProperty("cachePrepStmts", "true");
            config.addDataSourceProperty("prepStmtCacheSize", "250");
            config.addDataSourceProperty("prepStmtCacheSqlLimit", "2048");

            dataSource = new HikariDataSource(config);

    }

    public static Connection getConnection() throws SQLException {
        System.out.println("about to return connection "+ dataSource.isRunning());
        System.out.println("Thread "+ dataSource.getMaximumPoolSize());
        return dataSource.getConnection();
    }




    private DataSource(){}


}
```

In this section it is showing how a database connection pool is created and will work on the cloud SQL platform on google cloud this type of connection is needed for online hosting as this will reduce the load on the server and since app engine the host platform use limited resources i needed to reduce the load by opening the connections once and reusing the connection until it has finished almost like caching.

At the top I'm importing the required elements needed for the connection pool using hikariCP to control most of the aspect to do with the connection pool then in the main body of the class I'm creating a variable based on the hikari config which created a new config instance for the connection pool then making a initializing variable for the datasource which is where the data is collecting its information from and using that as the source for the pool.

Then in a static block I'm defining all the configuration options for the database connection with the config variable called for each method defining the information for the connect like you would for a standard connection so the JDBC URL which takes films as the last parameter as this is the database name and the database type which is MySQL same with user and password then a driver for the MySQL which is in my library then the next three properties for the config are cloud SQL specific so the socket factory will deal with the URL socket for the database connection then the cloud instance is the IPv6 address with a domain attached to it that is an instance name from my cloud SQL account specifying what database it is and which application which is for my app engine then it uses the set SSL to false otherwise this would need a SSL certificate on the server connection to work which is something I'm not worried about due to the security and encryption that app engine and cloud SQL already have in place and the last three settings are essentially for caching the SQL queries when called upon then in the last line for the config I'm setting the datasource to a new hikari data source with a parameter of config which is from the other methods above.

In the bottom method this is defining the connection for the database connection pool which is using the SQL connection class as the type and within the method it is logging the content for if the connection is running and the connection pool size for debugging in the console on app engine then the actual connection it will return datasoruce variable with the SQL getconntection method so that when called upon it essentially acts as a wrapper for SQL connection with caching this will speed up the application but also simplifies the connection to the database in the dao class.

FilmToList

```
package model;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import java.util.List;

@XmlRootElement(namespace = "com.ryanlove")
@XmlAccessorType(XmlAccessType.FIELD)
public class FilmtoList {
    @XmlElement(name = "film")
    private List<Film> films;

    public void setFilms(List<Film> films){
        this.films = films;
    }
    public List<Film> getFilms(){
        return films;
    }
    public FilmtoList(){
        super();
    }
}
```

In this class i have implement a class which will create a list for the film model which will be used in the XML formatting for the application as this includes the necessary information that when creating the XML object will see all the objects and implement this in the formatting for this request.

At the top the package is a model as this still is setting up a model to be used in the application as this is a blueprint for which the view and the controller will be following with the inclusion of some imported elements as the API's to be used in the application as the function they provide is to essentially convert this model and turn it into an XML compatible file hence why it is calling from the XML library then the list library as this will create a simple list as this is less complex than an arraylist which is being used in the film dao methods for the data.

Then in the main body is the annotations for XML root element which sets the main element which the rest of the XML data from the database will go in with the name being called as com.ryan as this makes sense to name the data after me as i am the one who has created the function for it then there is the accessory type which will essentially tell the XML what type of data it will be and in the class another XML annotation called element will define what each element name will be for that object which in my case is film due to all the data being film data this is showing why an API is essentially for the application as these type of functions deliver a seamless integration that as a developer could not be delivered upon if this was created by myself but like this function self written can use these API's and self code to create a method that uses both of these technologies.

This is the main function of the class it will define the variable for the films using the list data type with films used to define what to expect for this data type with it then defining getters and setters forthis method as this will enable the data for the XML request to append to this format when called upon using the correct naming conventions and camel case styling for the application then below is the constructor for this class calling upon the super method to retrieve information from its parent as mentioned above this self written part of the class will preform an action that will act like an API implementation and thus will ensure that when preforming the actions required but since this is only for my use case other API's will not implement these methods so creating them will ensure that all elements are considered for the application development.

Controller

createFilms

```
package controller;


import model.Film;
import model.FilmDAO;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
```

```
import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class createFilms extends HttpServlet {
        private static final long serialVersionUID = 1L;
        private FilmDAO filmDAO;

        public void init(){
            filmDAO = new FilmDAO();
        }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String action = request.getServletPath();
        try {
            switch(action) {
                default:createFilms(request, response);
            }
        } catch (SQLException e) {
            throw new ServletException(e);
        }

    }

    private void createFilms(HttpServletRequest request, HttpServletResponse response)
            throws SQLException, IOException {
            int id = Integer.parseInt(request.getParameter("id"));
        String title = request.getParameter("title");
        int year = Integer.parseInt(request.getParameter("year"));
        String director = request.getParameter("director");
        String stars = request.getParameter("stars");
        String review = request.getParameter("review");


        Film newFilm = new Film(id,title, year, director,stars,review);
        filmDAO.insertFilm(newFilm);
        response.sendRedirect("GUI");
    }

    }
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting

the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt library with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion in the main body of the application the file is to get the fields are in this section so this will create for example a int of id and year will parse thestring that it has passed to the user and then when it has parsed it will get the parameter from the request parameter with the argument as id or year for the integer values but the other values like title will do the same request as the id but without the parse integer method as these values are strings although this may seem simple but formatting this on the method will ensure that the validation properties for the application are being maintained then a Film data type creates a variable called newFilm with the new Film object with the parameters being the constructors from the film model then with the filmDAO object defined above add a method called insertFilm from the FilmDAO with the parameter newFilm this will essentially call that method from that class with the data collected from this film model constructor that is collected from the get parameter and finally respond method is called with the sendRedirect with the argument of GUI being passed as this will send the user to a certain URL from the application which is just the list of the films.

deleteFilms

```
package controller;

import javax.naming.NamingException;
import javax.servlet.http.HttpServlet;
import model.Film;
import model.FilmDAO;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class deleteFilms extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private FilmDAO filmDAO;

    public void init(){
        filmDAO = new FilmDAO();
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String action = request.getServletPath();
        try {
            switch(action) {
                default:deleteFilm(request, response);
            }
        } catch (SQLException e) {
            throw new ServletException(e);
        }

    }
    private void deleteFilm(HttpServletRequest request, HttpServletResponse response) throws SQLException, IOException {

        int id = Integer.parseInt(request.getParameter("id"));
        Film removeFilm = new Film(id);
        filmDAO.deleteFilm(removeFilm);
        response.sendRedirect("GUI");

    }


}
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt libary with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion then in the main body of the application is a variable of integer being defined thee request parameter from the URL which is id in this instance then it is parsed to an integer as the id data type is an integer and the request is a number value that is collected as a string hence why this type of convention is needed then the Film class is called with a variable called removeFilm which is equal to a new Film of id being created this is defined in the model then calling the filmdao variable above with the deletefilm method called with removefilm being called in the parameters with the response after this action is to redirect to the GUI to essentially refresh the page to see that the request has been performed.

editForm

```
package controller;

import model.Film;
import model.FilmDAO;

import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;

public class editForm extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private FilmDAO filmDAO;

    public void init() {
        filmDAO = new FilmDAO();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String action = request.getServletPath();
    try {
        switch (action) {
            default:
                editForm(request, response);
        }
    } catch (SQLException e) {
        throw new ServletException(e);
    }
}

    private void editForm (HttpServletRequest request, HttpServletResponse response)
        throws SQLException, ServletException, IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        Film editFilm = filmDAO.getFilmByID(id);
        RequestDispatcher dispatcher = request.getRequestDispatcher("FilmForm.jsp");
        request.setAttribute("film", editFilm);
        dispatcher.forward(request, response);

    }
}
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt library with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion then in the main body of this method the id is collecting the URL request id and storing it in the int as a parsed integer as the URL is a string value then calling the film class with the name edit film and calling the filmdao get by id as this will show the page in which the id that can be edited then calling the dispatcher to essentially show the page i want which is the filmform jsp then after this has been set to the dispatcher variable it will then set up the attributes as film and the edit film variable as this means that from the filmform page film can then be called as a variable in the jsp that is passed from the editform variable.

formFilm

```
package controller;

import model.FilmDAO;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class formFilm extends HttpServlet{
    private static final long serialVersionUID = 1L;
    private FilmDAO filmDAO;

    public void init(){
        filmDAO = new FilmDAO();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String action = request.getServletPath();
        try {
            switch (action) {

                default:
                    form(request, response);
            }
        } catch (IOException e) {
            throw new ServletException(e);
        }

    }


        private void form (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            RequestDispatcher dispatcher = request.getRequestDispatcher("FilmForm.jsp");
            dispatcher.forward(request, response);
        }



}
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that

may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt libary with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion then in the main body it will simple return the filmform jsp by calling the dispatcher library in a dispatcher named variable with it then forward to that URL request leading to it showing this page.

GuiList

```
package controller;

import model.Film;
import model.FilmDAO;

import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

public class GuiList extends HttpServlet {
        private static final long serialVersionUID = 1L;
        private FilmDAO filmDAO;

        public void init(){


            filmDAO = new FilmDAO();
        }



        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            doGet(request, response);
        }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {


            String action = request.getServletPath();
            try {
                switch(action) {
                    default:listFilm(request, response);
                }
            } catch (SQLException e) {
                throw new ServletException(e);
            }

        }




        private void listFilm(HttpServletRequest request, HttpServletResponse response)
                throws SQLException, IOException, ServletException {
            List<Film> allFilms = filmDAO.getAllFilms();
            request.setAttribute("allFilms", allFilms);
            RequestDispatcher dispatcher = request.getRequestDispatcher("FilmList.jsp");
            dispatcher.forward(request, response);
        }
    }
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt library with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion then in the main body i am calling the the getall films from the filmdao and storing it in a list type that is taking film as a parameter and is called allfilms this will ensure that when called upon it is called as a list rather than advanced type of arraylist as the XML format cannot handle this type of datatype due to its multi dimensional structure then it is setting a request attributes as allfilm using the allfilm variable as this will pass this data from the model to the view then requesting the page in a dispatcher variable that is showing the filmlist jsp that is processing all this information and dispatcher is forwarded to the URL request.

listFilms

```
package controller;


import model.Film;
import model.FilmDAO;
import com.google.gson.*;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.annotation.WebServlet;



public class listFilms extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private FilmDAO filmDAO;
```

```
    public void init(){
        filmDAO = new FilmDAO();
    }



    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {


        String action = request.getServletPath();
        try {
            switch(action) {
                default:listFilm(request, response);
            }
        } catch (SQLException e) {
            throw new ServletException(e);
        }

    }




    private void listFilm(HttpServletRequest request, HttpServletResponse response)
                throws SQLException, IOException, ServletException {
            List<Film> allFilms = filmDAO.getAllFilms();
            request.setAttribute("allFilms", allFilms);
        String format = request.getParameter("format");
        String out = null;

        if(format == null || "json".equals(format)){
            response.setContentType("application/json");
            out = "filmJSON.jsp";
        }
        if ("text".equals(format)){
            out = "filmText.jsp";
        }
        if("xml".equals(format)){
            response.setContentType("text/xml");
            out = "filmXML.jsp";

        }
        RequestDispatcher dispatcher = request.getRequestDispatcher(out);
        dispatcher.include(request, response);
    }

}
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that

may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt library with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion then in the main body of the similar to the GUI list the beginning uses the same variables ant methods to get all the films and stored it in a list then pass the allfilm variable to the view with allfilms being called as the variable in the view but something that is different is the use of the format request which takes a parameter as format which essentially means that in the URL if you are calling this it should do something then an out string variable that is essentially declaring an empty variable using null then there are three if statements which will return the request format for the data so the first one is JSON and since this is the default request it will take null as this return this automatically when called upon with the or operator called to make the text of JSON when request is equal to the format in the URL with out being set to the film for that data type so with JSON it shows the JSON page and text if statement will do the same but for the text request then once the data type requested is shown the dispatcher variable will take out as a parameter and include this in the URL when requested.

searchFilm

```
package controller;
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import model.Film;
import model.FilmDAO;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.bind.JAXBException;

public class searchFilm extends HttpServlet {
        private static final long serialVersionUID = 1L;
        private FilmDAO filmDAO;

        public void init() {
            filmDAO = new FilmDAO();
        }


        protected void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            doGet(request, response);
        }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            String action = request.getServletPath();
            try {
                switch (action) {

                    default:
                        searchFilm(request, response);
                }
            } catch (IOException | JAXBException e) {
                throw new ServletException(e);
            }

        }
```

```
        private void searchFilm(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException, JAXBException {
            String title = request.getParameter("title");
            String format = request.getParameter("format");
            ArrayList<Film> search = filmDAO.searchFilms(title);
            request.setAttribute("search", search);
            String out = null;

            if(format == null || "json".equals(format)){
                response.setContentType("application/json");
                out = "filmJSON.jsp";
            }
            if ("text".equals(format)){
                out = "filmText.jsp";
            }
            if("xml".equals(format)){
                response.setContentType("text/xml");
                out = "filmXML.jsp";

            }
            RequestDispatcher dispatcher = request.getRequestDispatcher(out);
            dispatcher.include(request, response);
        }

    }
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt library with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion then with then method it contains the variable of title and format which stores the requested URL parameter when called upon then the search variables calling the arraylist data type with film being passed through to essentially show how everything should be stored which is storing the filmdao object that is getting the searchfilm request with a parameter of title as this will get the variable that the user is requesting from the URL and like the listfilms servlet it is using the same format if statement function to set the response content type and it will perform the same out method to show this data as the listfilms method preforms.

updateFilms

```
package controller;


import model.Film;
import model.FilmDAO;

import java.io.IOException;
import java.sql.SQLException;
import java.util.List;

import javax.naming.NamingException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class updateFilms extends HttpServlet {

    private static final long serialVersionUID = 1L;
    private FilmDAO filmDAO;

    public void init(){
        filmDAO = new FilmDAO();
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String action = request.getServletPath();
        try {
            switch(action) {
                default:updateFilm(request, response);
            }
        } catch (SQLException e) {
            throw new ServletException(e);
        }

    }


    private void updateFilm(HttpServletRequest request, HttpServletResponse response)
            throws SQLException, IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        String title = request.getParameter("title");
        int year = Integer.parseInt(request.getParameter("year"));
        String director = request.getParameter("director");
        String stars = request.getParameter("stars");
        String review = request.getParameter("review");

        Film editfilm = new Film(id,title, year, director,stars,review);
        filmDAO.updateFilm(editfilm);
        response.sendRedirect("GUI");
    }
}
```

At top of this class it is calling the package as controller the reason for this is that this will handle the request for the model and then pass them to the view hence the name controller as this is dealing with this design pattern next it is importing from the Film and FilmDAO from the model package as this will require this data for processing to actual handle the requests then the imports for the exceptions are required to show a log and catch any error within the application then the rest of the imports are servlet related and will handle elements like the HTTP request and response ensuring that this information can be used on the web and that all the functions are sending there information to the correct resources.

Next is the class its self which is extending the httpservlet as this is taking some of the functions from this class and using it within the servlet method this is required otherwise how else would the application get the data for the httpservlet without calling upon from that class next is a variable which is defining a serialization id this is used to ensure that when desensitization of this object errors like invalid class exception are not showing as they require this id and the FilmDAO data type is creating a variable called filmdao to be used in this servlet execution.

In terms of the standard methods for the servelt there is an int method that will create a new object for the filmdao to be called into otherwise it would be empty and return no value this is also showing that using the self written aspect

of the application is making my methods perform actions that enable faster development rather than implementing this in each servlet i can do this by calling the object of filmdao then there is a post method that is taking the parameters of request and response from the httpservelt library and it throwing the exceptions for the methods that may throw an exception if the application is showing errors within this is is calling the DoGet method with request and response which may seem odd in terms of implementation but this will essentially do the same action as do get meaning it will work the same way for both methods leading to better performance as this means only one method needs to be created with the other one calling upon it with the DoGet method calling the same parameters that DoPost is calling upon then within this method a string called action is calling the request parameter that is getting the servlet path for the application so it will get the correct servlet path then a try and catch statement which in the try is running a switch statement with action as the parameter and the default action is the servelt method declared below this method with the parameters of request and response to ensure this data can be used in that method and the catch method is throwing a servlet exception with the SQL exception parameter from the catch as a parameter.

The bottom method this is a create method the will take the parameters of request and response from the httpservelt library with exceptions of SQL and IO thrown to ensure that the application is collecting errors ensuring that the application is performing to a level that when debugging errors will appear but also helps the user as they get the data they need due to this inclusion in the main body of the method it is performing the same action as the createfilms method by getting all the information and storing it in variables then storing them in the new film constructor which is saved to editfilm variable and then the element that is different it that calling the updatefilm method from the filmdao method and then passing editfilm variable to update these elements and redirect to the GUI URL.

View

ajax.js

```
$(document).ready(function () {
    $("#allFilms").click(function () {
        $.get("Film?format=xml",function (data,status,request) {
            console.log(getData(request))

        },"xml")
    })
})

function getData(xml) {
    var i;
    var xmlFormat = xml.responseXML
    var tbl ="<table><tr><th>ID</th><th>Title</th><th>Year</th><th>Director</th><th>Stars</th><th>Reviews</th></tr>"
    var e = xmlFormat.getElementsByTagName("film")
    for (i=0; i < e.length; i++){
        tbl += "<tr><td>"+e[i].getElementsByTagName("id")[0].childNodes[0].nodeValue+"</td>"+
            "<td>"+e[i].getElementsByTagName("title")[0].childNodes[0].nodeValue+"</td>"+
            "<td>"+e[i].getElementsByTagName("year")[0].childNodes[0].nodeValue+"</td>"+
            "<td>"+e[i].getElementsByTagName("director")[0].childNodes[0].nodeValue+"</td>"+
            "<td>"+e[i].getElementsByTagName("stars")[0].childNodes[0].nodeValue+"</td>"+
            "<td>"+e[i].getElementsByTagName("review")[0].childNodes[0].nodeValue+"</td>"+"</tr></table>"
    }
document.getElementById("xml").innerHTML = tbl

}
```

In this JavaScript file it will call upon the XML data by using an ajax request to get the URL data to then be displayed on the web page this is being done by jQuery and this will be using ajax to call upon the data and formatting it with the getdata function that parses and and gives the data some structure so at the top of the script it is calling the jQuery document .ready function that will when the document has loaded and is able it will run this function which takes a callback function that is getting the allfilms id assigned on the html page for the button and when it is clicked it will run another callback function that is calling the ajax request called get which will take the URL that needs requesting so in this instance it will be the XML data then it will take a function as parameter with its parameters being data status and request and in this function it is just logging the getdata function below with request data being called upon and the get film function last parameter is the XML as this is the request type this would not be possible if i did not use the jQuery library to perform these actions as this simplified the process rather than learning the JavaScript default ajax request that is mainly for older browsers but with jQuery being used it ensured that browser compatibility is dealt with without having to worry about this type of issue this could have been accomplished by my

self but this would result in longer development times that are not necessary as a library accomplished this in a simple manner.

In the get data function it is passing a parameter of XML as this will allow the XML data to be obtained and dealt with in this function i define a variable of i as this will be used in a for loop then i define a XML format variable that takes the parameter XML value with a response XML which will ensure that XML data can be dealt with and formatted in an appropriate manner then another variable called tbl which defines the table in which the data will be stored in using html to define this information and below this a variable called e is taking the xmlformat variable and adding a method that will get the tag name from the XML request of film as this has been defined in the filmtolist model so that each film can be sorted rather than have a static value for each film as the structure of XML requires this type of data to store it.

n the for loop it is running a until e.length is less than i with it iterating until this has been met then in the variable tbl it is adding more data on to that variable by += symbol which speeds up the process and results in cleaner code with it including table row with td elements containing the data which is all the same information just with tag name replaced to match the request field to match the table layout ie id will get only the id but it uses the e variable with i in the array parameters as this will dynamically get the data until it has all this information on the tag name element it will use an array index of 0 to get the XML data index then a childnode with an index of 0 as this will the same data but turning it into a node object and finally getting the node value will essentially return the data from the XML that is required and the last element that is out of the for loop is the document get element by id with XML being passed as this is a div defined on the html page where the data will be rendered with the inner html method applied as this will take the tbl variable information like the html element in the text and display it in this format that equals tbl to sent this data to the html page.

FilmForm.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Ryan Love
  Date: 16/12/2019
  Time: 19:42
  To change this template use File | Settings | File Templates.
--%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>FilmDB</title>
</head>
<body>
<center>
    <h2>
        <a href="${pageContext.request.contextPath}/create">Add New Film Entry</a>
           
        <a href="${pageContext.request.contextPath}/Film">List All Film Entiies</a>

    </h2>
</center>
<div align="center">
    <c:if test="${film != null}">
    <form action="update" method="post">
        </c:if>
        <c:if test="${film == null}">
        <form action="create" method="post">
            </c:if>
            <table border="1" cellpadding="5">
                <caption>
                    <h2>
                        <c:if test="${film != null}">
                            Edit Film
                        </c:if>
                        <c:if test="${film == null}">
                            Add New Film
                        </c:if>
                    </h2>
                </caption>
                <c:if test="${film != null}">
                    <input type="hidden" name="id" value="<c:out value='${film.id}' />" />
```

```
                </c:if>
                <tr>
                    <th>ID: </th>
                    <td>
                        <input type="text" name="id" value="<c:out value='${film.id}' />"/>
                    </td>
                </tr>

                <tr>
                    <th>Title: </th>
                    <td>
                        <input type="text" name="title" value="<c:out value='${film.title}' />"/>
                    </td>
                </tr>
                <tr>
                    <th>Year: </th>
                    <td>
                        <input type="text" name="year" value="<c:out value='${film.year}' />"/>
                    </td>
                </tr>
                <tr>
                    <th>Director: </th>
                    <td>
                        <input type="text" name="director" value="<c:out value='${film.director}' />"/>
                    </td>
                </tr>
                <tr>
                    <th>Stars: </th>
                    <td>
                        <input type="text" name="stars" value="<c:out value='${film.stars}' />"/>
                    </td>
                </tr>
                <tr>
                    <th>Review: </th>
                    <td>
                        <input type="text" name="review" value="<c:out value='${film.review}' />"/>
                    </td>
                </tr>
                <tr>
                    <td colspan="2" align="center">
                        <input type="submit" value="Save" />
                    </td>
                </tr>
            </table>
        </form>
</div>
</body>
</html>
```

In this page it is a jsp page that is rendering it as a html page with the main elements being a table and a form element it which the user can submit the data for the film information but there are element that are different since this is a jsp file it can use java element in the page which it does it calls upon the jstl library from jsp and uses the prefix as c to define certain function to be preformed so element like c if will use an if statement and c out will output data the main advantage it that you can take data and use it in a stand html format making it dynamic rather than static in its implementation so the if statements will show that if an the film data passed from the servlet is null it will show the create form action if it is not null it will preform the update action for the corresponding submit type and with each for value it has a name that is acquired from the servlet to get this information but the value is set to the film and what ever attribute for that input so review will have film.review to either update or create this method this is useful implementing both the create and update form on the same page as this means that you don't have to load more pages that is required thus saving on loading times and user experience as the format f the html will be consistent throughout.

filmJSON.jsp

```
<%@ page import="model.Film" %>
<%@ page import="com.google.gson.Gson" %>
<%@ page import="java.util.List" %>
<%@ page import="com.google.gson.GsonBuilder" %>
<%@ page import="java.io.IOException" %>


<%
    List<Film> films = (List<Film>) request.getAttribute("allFilms");
    List<Film> search =(List<Film>) request.getAttribute("search");

    GsonBuilder gsonB = new GsonBuilder().setPrettyPrinting();
Gson gson = gsonB.create();
```

```
String json;
String jSearch;

json =gson.toJson(films);
jSearch = gson.toJson(search);



       out.println(jSearch);
        out.println(json);


%>
```

In this page it is converting the data from the MySQL database and converting it into JSON this is useful as this will ensure the data type can be set to the correct format this is all done in a jsp file so java code can run in this file when the java code in this page first the imports are selected like the mode for the data and GSON which will handle the conversion of java object to JSON so at the beginning I'm creating 2 variables from list called films which is then casting this to another list object with the get attribute of that request which is allfilms and search next I'm defining gsonb using the class as GsonBuilder which is creating a new GsonBuilder with set pretty print as a method to create a JSON format that is in a more readable way other than showing the json in a block of text then below that is the Gson class variable called gson that will use the create method on the gsonb variable to create the JSON then below is two variables for search and all films then they are taking the gson variable and using the method tojson with the either films or search to send the data from these variable to be correctly converted to JSON and finally it will print the results with out.println.

FilmList.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Ryan Love
  Date: 16/12/2019
  Time: 16:35
  To change this template use File | Settings | File Templates.
--%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>FilmDB</title>
</head>
<body>
<center>
    <h1>Film</h1>
    <h2>
        <a href="${pageContext.request.contextPath}/show">Add New Film Entry</a>
           
        <a href="${pageContext.request.contextPath}/GUI">List All Film Entiies</a>

    </h2>
</center>
<div>
    <table border="1" cellpadding="10">
        <h2>List of All Films</h2>
        <tr>
            <th>ID</th>
            <th>Title</th>
            <th>Year</th>
            <th>Director</th>
            <th>Stars</th>
            <th>Review</th>
            <th>Operations</th>
        </tr>
        <c:forEach var="film" items="${allFilms}">
            <tr>
                <td><c:out value="${film.id}" /></td>
                <td><c:out value="${film.title}" /></td>
                <td><c:out value="${film.year}" /></td>
                <td><c:out value="${film.director}" /></td>
                <td><c:out value="${film.stars}" /></td>
                <td><c:out value="${film.review}" /></td>
                <td>
                    <a href="${pageContext.request.contextPath}/edit?id=<c:out value='${film.id}' />">Edit</a>
                        
```

```
                    <a href="${pageContext.request.contextPath}/delete?id=<c:out value='${film.id}' />">Delete</a>

                </td>
            </tr>
        </c:forEach>
    </table>
</div>
</body>
</html>
```

This page is similar to the for updating and creating films but the main element that is different is that it is using the for each method from the c prefix that will loop through each film value that is formatted in the table for html and for the a links to go to other locations it is using the servlet page context as the server deployment may use a different location each time so instead of hard coding it will edit the page based on the URL with both the edit and delete queries taking film id from c out to go to that location and page when requesting it with the for each in the c prefix looping over the films variable declare in the servlet.

filmText.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Ryan Love
  Date: 21/12/2019
  Time: 15:45
  To change this template use File | Settings | File Templates.
--%>
<%@ page import="model.Film" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.Arrays" %>
<%@ page import="model.FilmtoList" %>

<% List<Film> films = (List<Film>) request.getAttribute("allFilms");
    List<Film> search =(List<Film>) request.getAttribute("search");


    if(films != null) {
        for (int i = 0; i < films.size(); i++) {
           out.println(films.get(i));
        }
    } else {

    }
    if (search != null) {
        for (int i = 0; i < search.size(); i++) {
            out.println(search.get(i));
        }

    } else {

    }




%>
```

In this page it will return the data from the film database in a text format this is done in a jsp as this will allow for java code execution to be preformed which compared to other methods is very effective as this means you can get java libraries and java environment to complete actions so at the top it is declaring all the imports that are required like the list and array to give the data to a data structure and like the other request for the formats it is declaring the variables as a list that is equaling the attribute from its required area the element that is different is the if statements that will ensure that if the data is not null it will display the data with a for loop inside that iterates through films using the size method as this is list it uses a different action as length but preforms the same action which then prints the result based on the films.get method with i as a parameters as this is the dynamic value from the loop iterator with a blank else block as i want any null element to do nothing as this should not be displayed.

filmXML.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Ryan Love
  Date: 21/12/2019
```

```
  Time: 14:28
  To change this template use File | Settings | File Templates.
--%>
<%@page import="javax.xml.bind.*" %>
<%@page import="java.util.List" %>
<%@page import="model.FilmtoList" %>
<%@page import="model.Film" %>
<%@ page import="model.FilmDAO" %>
<%@ page trimDirectiveWhitespaces="true" %>
<%
    List<Film> film = (List<Film>) request.getAttribute("allFilms");
    List<Film> search = (List<Film>) request.getAttribute("search");
    FilmtoList ftl = new FilmtoList();
    ftl.setFilms(film);
    ftl.setFilms(search);


try{
    JAXBContext context = JAXBContext.newInstance(FilmtoList.class);
    Marshaller m = context.createMarshaller();
    m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);

        m.marshal(ftl, out);

} catch (JAXBException e){
 System.out.println(e);
}

%>
```

In this file it will essentially convert the data from the servlet into XML data so like the other converting files it calls for them to a list object but the element that is different is below it is calling the film to list class that is creating a new filmtolist object it then uses the setfilms on the ftl variable with the parameters for film and search being passed in separated objects then a try and catch block classes the JAXBcontext class which the creates a new instance for the filmto list class then it will use the marshal which will format the data which is then shown in the setproperty method that takes marshals formatted output and Boolean of true then in the m.machel statement it will take the parameters of ftl and out which is the saved variable and the print type with the final statement being a catch statement that prints the exception this is showing a self written class i had created to preform the XML data structure as this will set the XML based on this filmtolist class as their libraries did not perform this function i needed so i created a simple solution that fit my needs perfectly.

getJson.js

```
$(document).ready(function(){
    $("#getJSON").click(function(){
        $.getJSON(window.location.origin+window.location.pathname+"Film?format=json", function(data){
            console.log(data)
            for (var i = 0; i < data.length; i++){
                document.write("<br><br>array index: " + i);
                var obj = data[i];
                for (var k in obj){
                    var v = obj[k];
                    document.write("<br> - " + k + ": " + v);
                }
            }


            var d = data
                $("#json").html(`${d[i]}`)
                console.log(json)

        });

    });
});
```

In this section it is using the same function like the ready and the click but for the getjson button event but the area that is different is the getJson method as this will get the window location and path name and add this to the request URL for the JSON a callback function is request that runs a for loop of the data parameter from the callback function it will then write to the screen the index as well as declaring the data[i] as the obj then another for loop with k being defined as an index key in obj within this it uses a variable called v that is equal to the obj array with the index being k it is then writing to document with k being passed to get the key name and v being passed to get the actual data out of the loops I'm declaring data as d and writing this to the html in the JSON element id using es6 template literals

using and the index of i to show all the results from the this object this is useful for the usage of newer implementations as es6 template literals introduced passing data to the view without a technology like jsp to perform this actions which will also show es6 in the getText section and how it is so useful for getting the results by using these newer technologies.

getText.js

```
$(document).ready(function(){
    $("#getText").click(function(){
        $.get(window.location.origin+window.location.pathname+"Film?format=text", function(data){
            var d = new Array(data.split("Film"))
            for (let i = 0; i < d.length ; i++) {
                var arr = d[i];
                for (var j  = 0; j < arr.length; j++) {
                    arr[j].match(/id=[0-9][^,]*/g) ? $("#text").append($('<li>').html(`${arr[j].match(/id=[0-9][^,]*/g) }`)) :
                    arr[j].match(/title=[a-zA-Z0-9][^,]*/g) ? $("#text").append($('<li>').html(`${arr[j].match(/title=[a-zA-Z0-
                    arr[j].match(/year=[0-9][^,]*/g) ? $("#text").append($('<li>').html(`${arr[j].match(/year=[0-9][^,]*/g)}`))
                    arr[j].match(/director=[a-zA-Z0-9][^,]*/g) ? $("#text").append($('<li>').html(`${arr[j].match(/director=[a-
                    arr[j].match(/stars=[a-zA-Z0-9][^,]*/g) ? $("#text").append($('<li>').html(`${arr[j].match(/stars=[a-zA-Z0-
                    arr[j].match(/review=[a-zA-Z0-9][^\]/]*/g) ? $("#text").append($('<li>').html(`${arr[j].match(/review=[a-zA
                    console.log(arr[j])
                }
            }
        });
    });
});
```

In this section it similar to the other formats for the ajax as this is reading the document with a click event on the getText button for the specific id the area in which is different is the content to get the data from a string first m declaring a d variable that is creating a new array with the data parameter being passed like in the other formats for ajax that then is splitting up to every identify that starts with Film this intern will create two arrays next a for loop which is based on d.length to get the first array size which should be one but still basing the iterator on this as i will ensure consistency and dynamic data if more databases are added which is saved to arr variable in the other for loop inside it is using the arr.length for this to get the length of the contents inside this internal array then for each item in the film element it is using the j identifier as this is the other iterator name in the loop on arr with the .match used which then takes a regular expression formula which gets the requested name like id and then using regular expression to get the dynamic values within this reg expression i created a stopping point in the square brackets as a comma as stopping point as I'm getting the text format so there is no really formatting unless i create it in the array i then used a ternary operator which essentially simplifies the process of an if statement as this will match all that data collected using the match function and will preform a jQuery action in which it will append to the id of text in the html page a li tag with a .html added to pass the data using es6 template literals using the same format as the condition set in the match method to return only that film element then using the : symbol this correlates to a else statement if it doesn't match the format it will just console log the data it doesn't recognized with what identify is the issue using plain text each regular expression is filtered for that data type so id will return a number and text based ones will return text and numbers with only matching difference for each one being the review section as this needs to stop at the bracket as this text is shown in this array type so stopping at this point meant when going over the loop again it will show only the data and not extra characters from the sorted array this would not have been possible without es6 and new technologies the other solution that use es6 could be accomplished with other tools but in this instance i needed something that has not been created so i created the function my self and the result is clean text format without the need of any extra libraries that could slow down the program other than jQuery as the ajax request still needs to be preformed (Mardan, 2020).

index.html

```
<html>
<head>
    <title>Hello, I am a Java web app!</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="ajax.js"></script>
    <script src="getJson.js"></script>
    <script src="getText.js"></script>
</head>
<body>
<h1>Simple Java Web App Demo</h1>
<p>To invoke the java servlet click <a href="Servlet">here</a></p>
<button id="allFilms">Get XML</button>
<button id="getJSON">Get JSON</button>
```

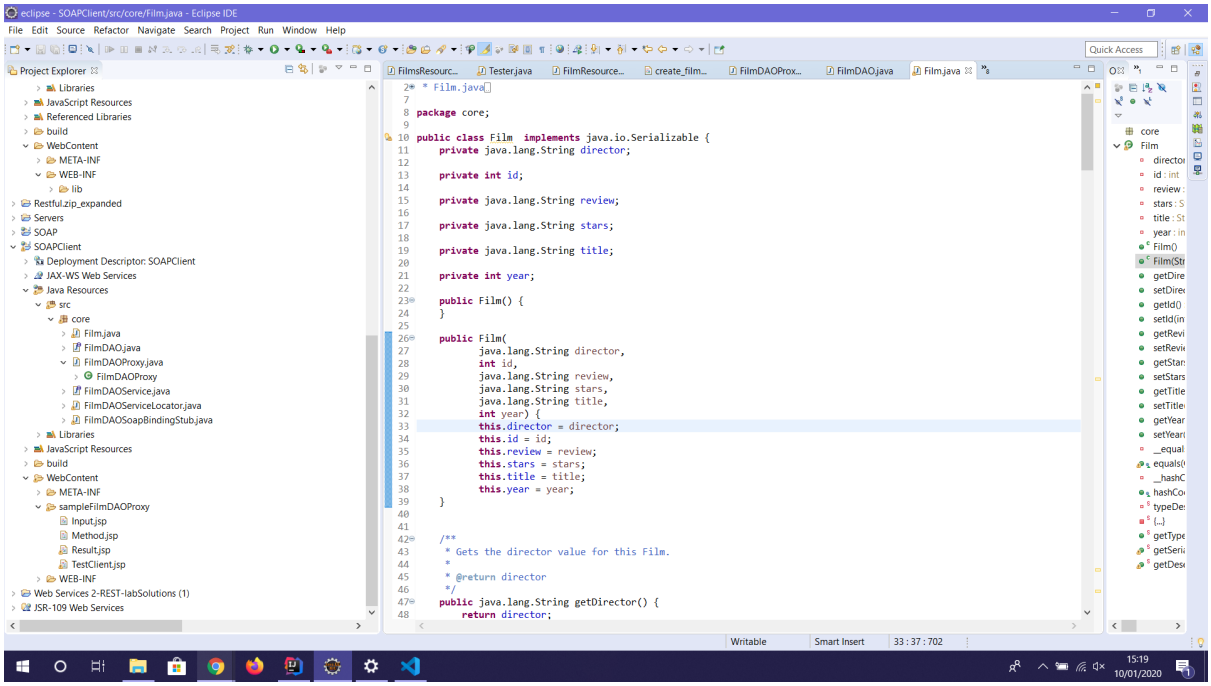
```
<button id="getText">Get Text</button>

<div id="xml"></div>
<div id="json"></div>
<div id="text"></div>

</body>
</html>
```

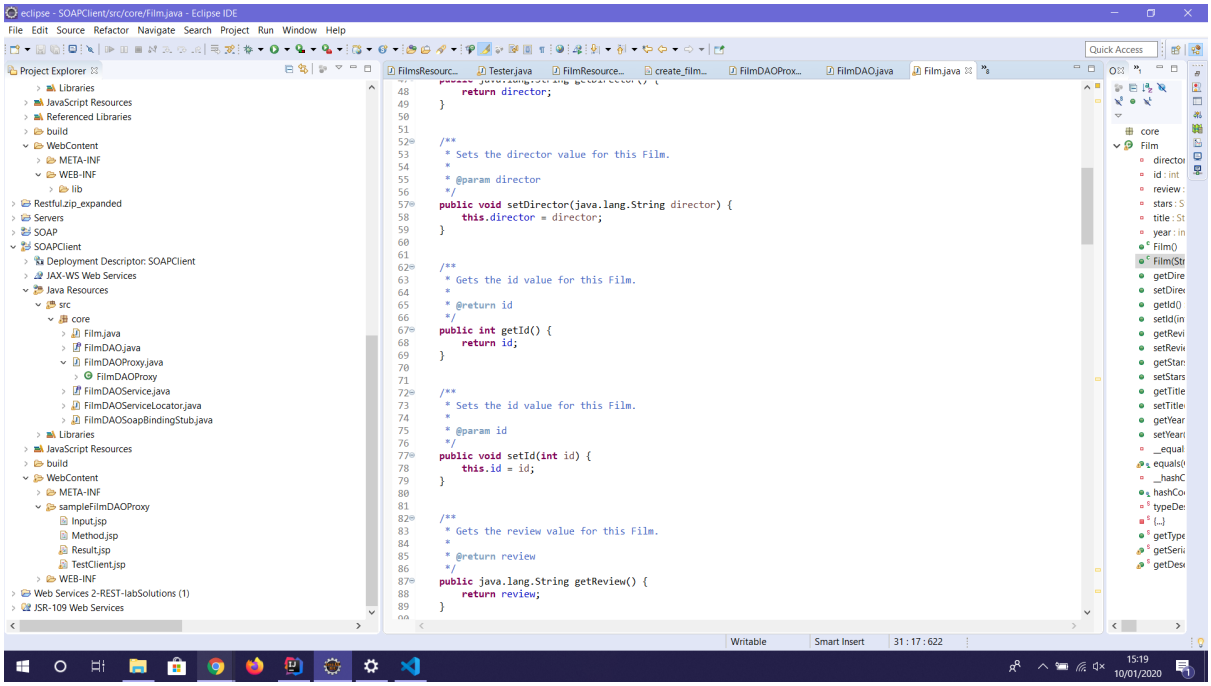
The index html page is the starting point for the application when the application it is structure like any normal html in the head tag it is declaring a title with 4 script tags one is importing the jQuery library that is used for the ajax requests then the others point to the area for each formatting option for the ajax then in the body there is a link for a servlet to test the functionality with a basic request then there are three buttons which will do the action for the ajax request and the divs below will get the data from each and show them in that area with this modular setup of the scrips it means that load times will be reduced as the JavaScript will not needed to be rendered if not called upon.

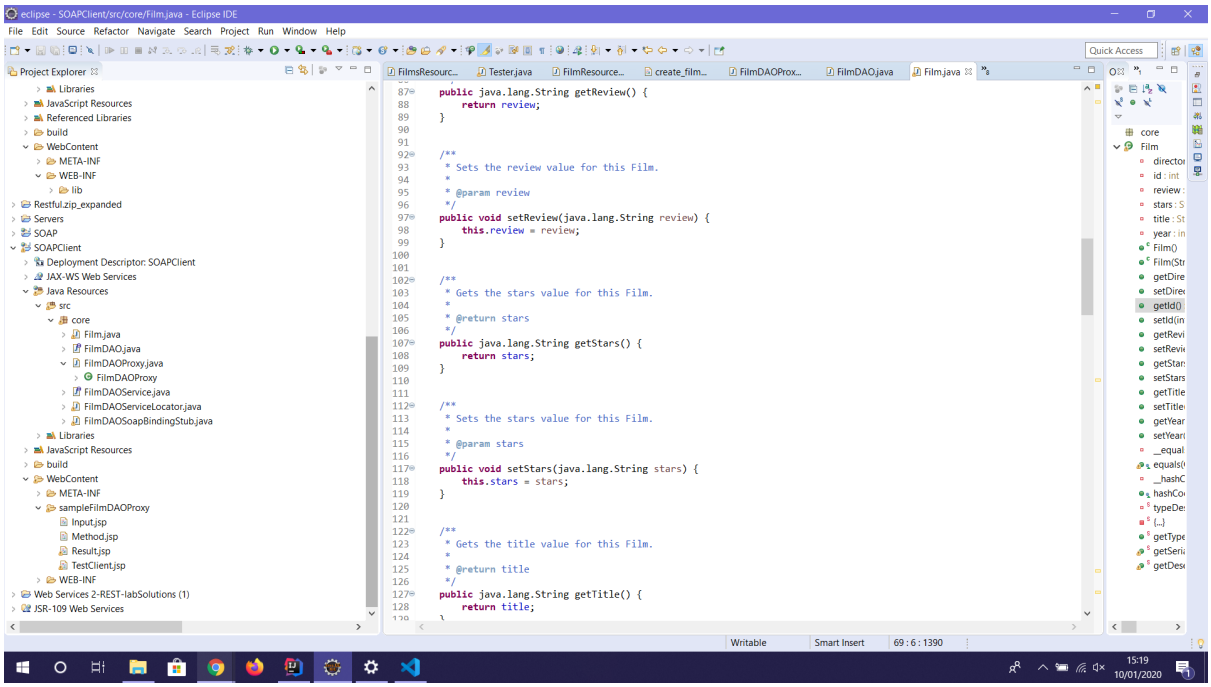
SOAP

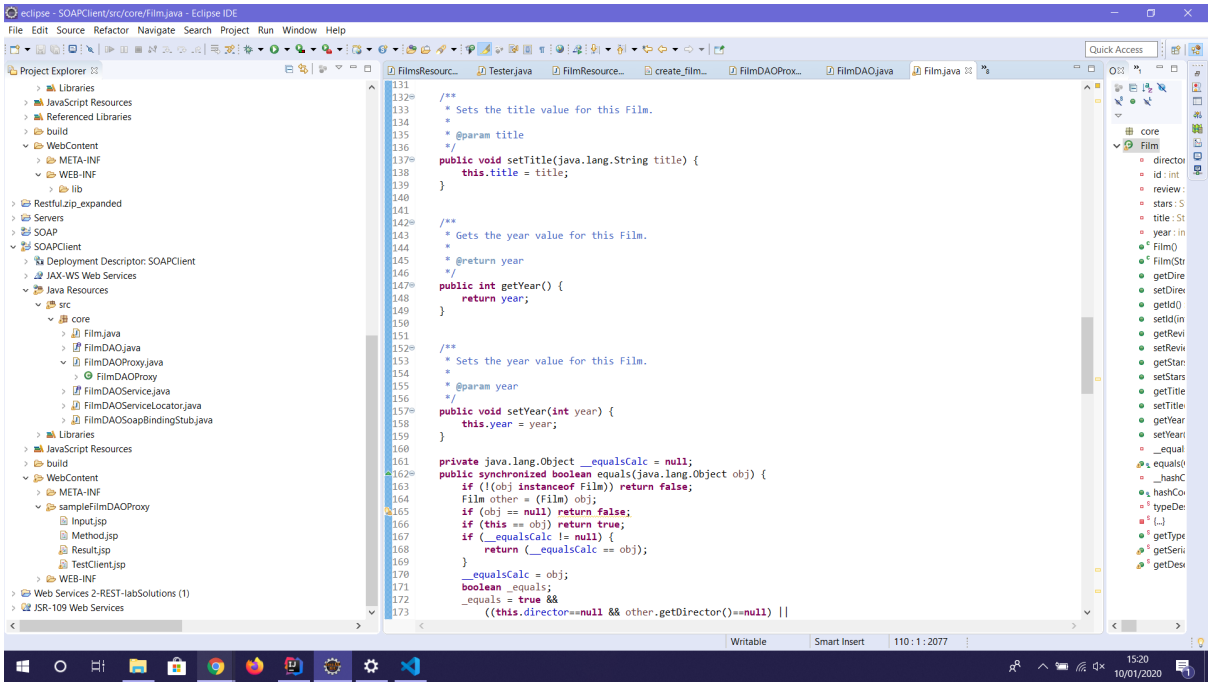
When talking about the soap project i produced this is something that with growing technologies can really show improvement with the type of elements that should be generated so in the soap project i created an new project in which i added my film dao class as well as the film model using the same functions as shown above in the HTTP section i then selected the filmdao class then selected the generated soap it then showed the pop up window in which it will create a soap web service from the code which didn't take very long to generate and then many different classes appeared which handle this type of request which will be shown below but this type of tool is very useful as the creating of a soap project with the little time i had as well as learning an entire outdated web service would have been a failed experiment so this ease the development process in this aspect ad this also created a client int which when run on a server will show all the endpoints on a web page and you are able to interact with it in a seamless manner thus making even more useful as this solves the back end and front end issue that could be potentially occurring all though this method mean that i will not understand the full implementation as the application uses generated code but since it will complete this action why not use a useful tool to aid in this process of development and since it is using the same methods from the HTTP dao everything explained in that section is applicable to the SOAP function.

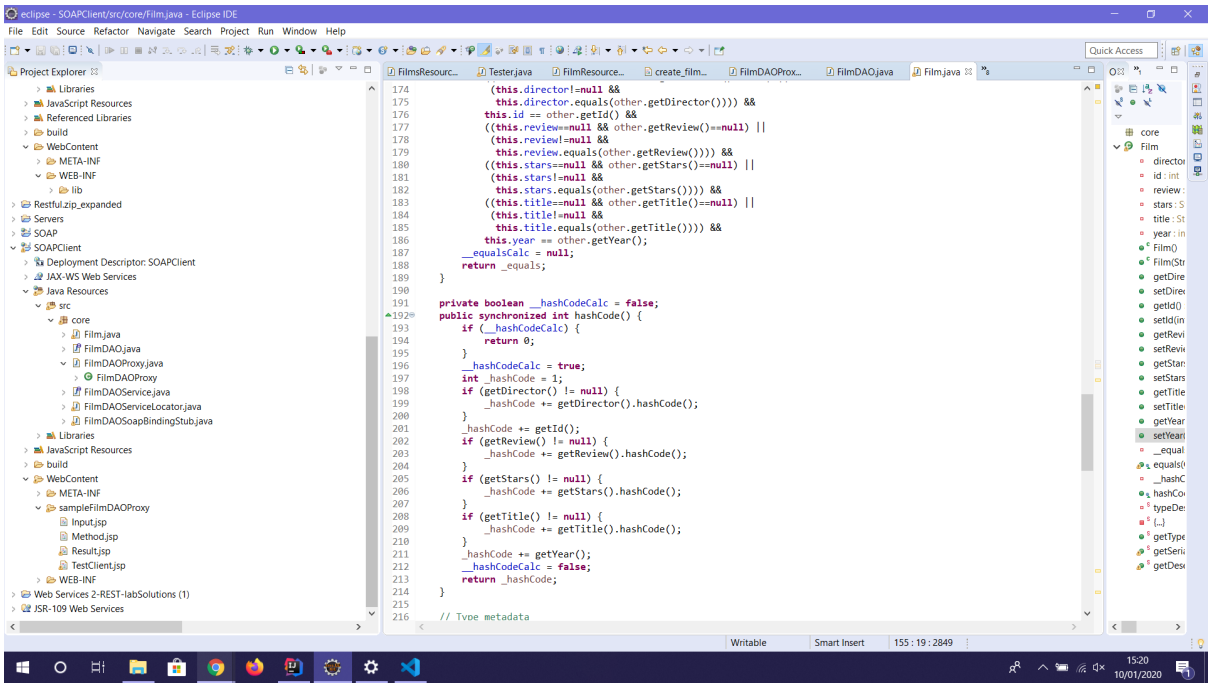
Film

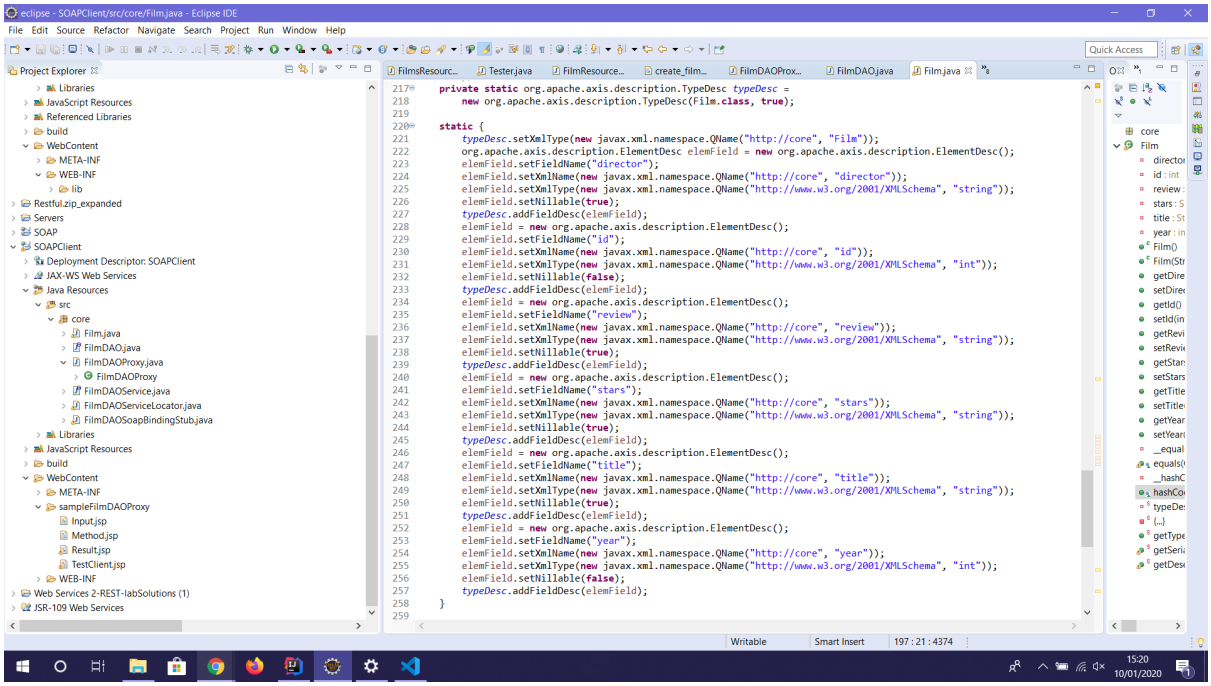


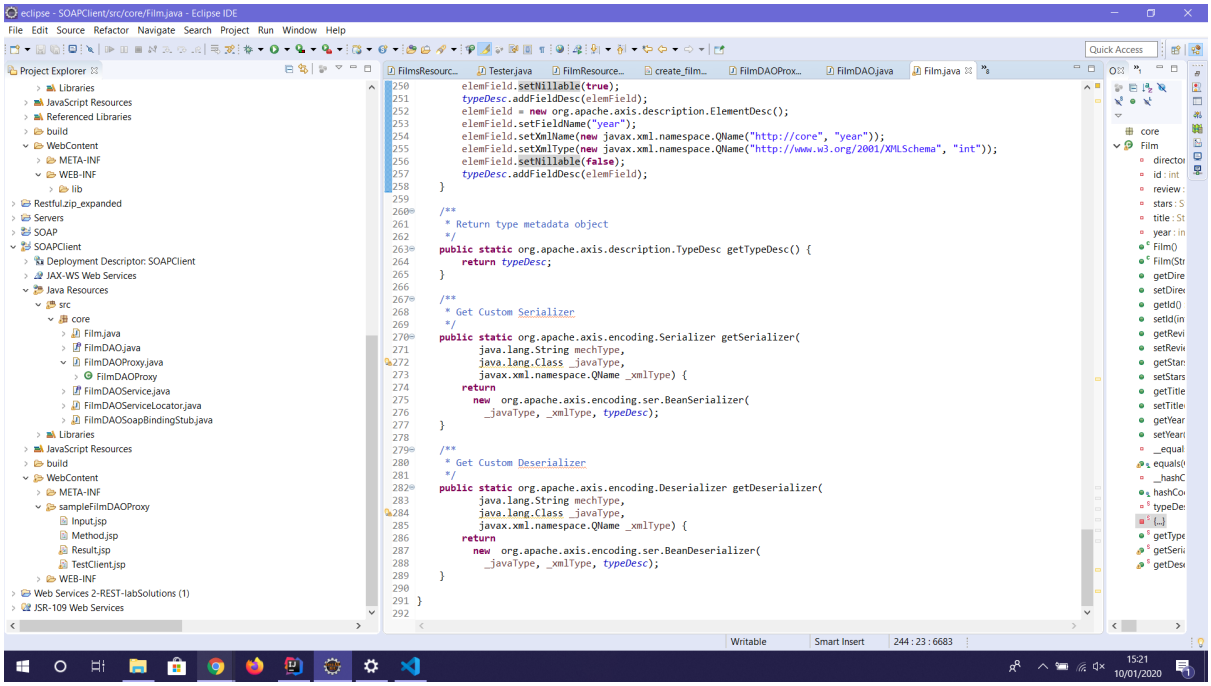

eclipse - SOAPClient/src/core/Film.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOProxy
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmsResourc...
Tester.java
FilmResource...
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
return director;
}
/**
* Sets the director value for this Film.
*
* @param director
*/
public void setDirector(java.lang.String director) {
this.director = director;
}
/**
* Gets the id value for this Film.
*
* @return id
*/
public int getId() {
return id;
}
/**
* Sets the id value for this Film.
*
* @param id
*/
public void setId(int id) {
this.id = id;
}
/**
* Gets the review value for this Film.
*
* @return review
*/
public java.lang.String getReview() {
return review;
}
Writable
Smart Insert
31 : 17 : 622
15:19
10/01/2020


eclipse - SOAPClient/src/core/Film.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
FilmsResourc...
Tester.java
FilmResource...
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
public java.lang.String getReview() {
return review;
}
/**
* Sets the review value for this Film.
*
* @param review
*/
public void setReview(java.lang.String review) {
this.review = review;
}
/**
* Gets the stars value for this Film.
*
* @return stars
*/
public java.lang.String getStars() {
return stars;
}
/**
* Sets the stars value for this Film.
*
* @param stars
*/
public void setStars(java.lang.String stars) {
this.stars = stars;
}
/**
* Gets the title value for this Film.
*
* @return title
*/
public java.lang.String getTitle() {
return title;
}
Writable
Smart Insert
69 : 6 : 1390
15:19
10/01/2020

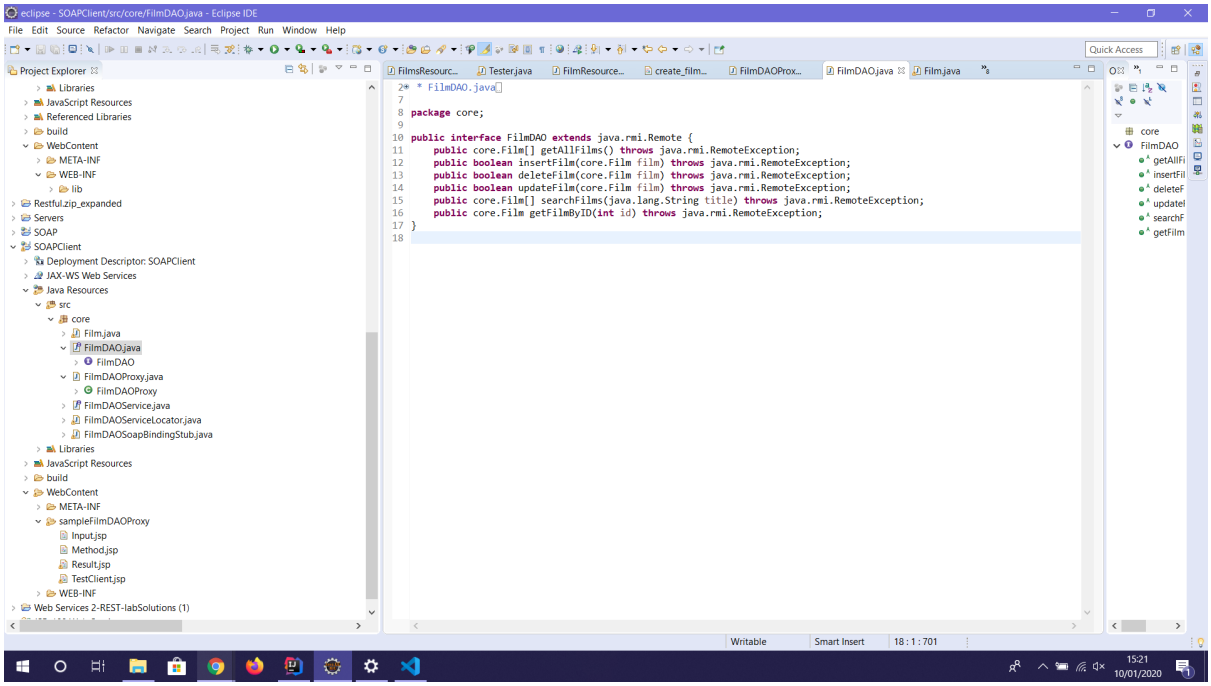

eclipse - SOAPClient/src/core/Film.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOProxy
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmsResourc...
Tester.java
FilmResource...
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
Quick Access
/**
* Sets the title value for this Film.
*
* @param title
*/
public void setTitle(java.lang.String title) {
this.title = title;
}
/**
* Gets the year value for this Film.
*
* @return year
*/
public int getYear() {
return year;
}
/**
* Sets the year value for this Film.
*
* @param year
*/
public void setYear(int year) {
this.year = year;
}
private java.lang.Object __equalsCalc = null;
public synchronized boolean equals(java.lang.Object obj) {
if (!(obj instanceof Film)) return false;
Film other = (Film) obj;
if (obj == null) return false;
if (this == obj) return true;
if (__equalsCalc != null) {
return (__equalsCalc == obj);
}
__equalsCalc = obj;
boolean _equals;
_equals = true &&
((this.director==null && other.getDirector()==null) ||
Writable
Smart Insert
110 : 1 : 2077
15:20
10/01/2020


eclipse - SOAPClient/src/core/Film.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
(this.director!=null &&
this.director.equals(other.getDirector()))) &&
this.id == other.getId() &&
((this.review==null && other.getReview()==null) ||
(this.review!=null &&
this.review.equals(other.getReview()))) &&
((this.stars==null && other.getStars()==null) ||
(this.stars!=null &&
this.stars.equals(other.getStars()))) &&
((this.title==null && other.getTitle()==null) ||
(this.title!=null &&
this.title.equals(other.getTitle()))) &&
this.year == other.getYear();
__equalsCalc = null;
return _equals;
}
private boolean __hashCodeCalc = false;
public synchronized int hashCode() {
if (__hashCodeCalc) {
return 0;
}
__hashCodeCalc = true;
int _hashCode = 1;
if (getDirector() != null) {
_hashCode += getDirector().hashCode();
}
_hashCode += getId();
if (getReview() != null) {
_hashCode += getReview().hashCode();
}
if (getStars() != null) {
_hashCode += getStars().hashCode();
}
if (getTitle() != null) {
_hashCode += getTitle().hashCode();
}
_hashCode += getYear();
__hashCodeCalc = false;
return _hashCode;
}
// Type metadata
Writable
Smart Insert
155 : 19 : 2849
15:20
10/01/2020

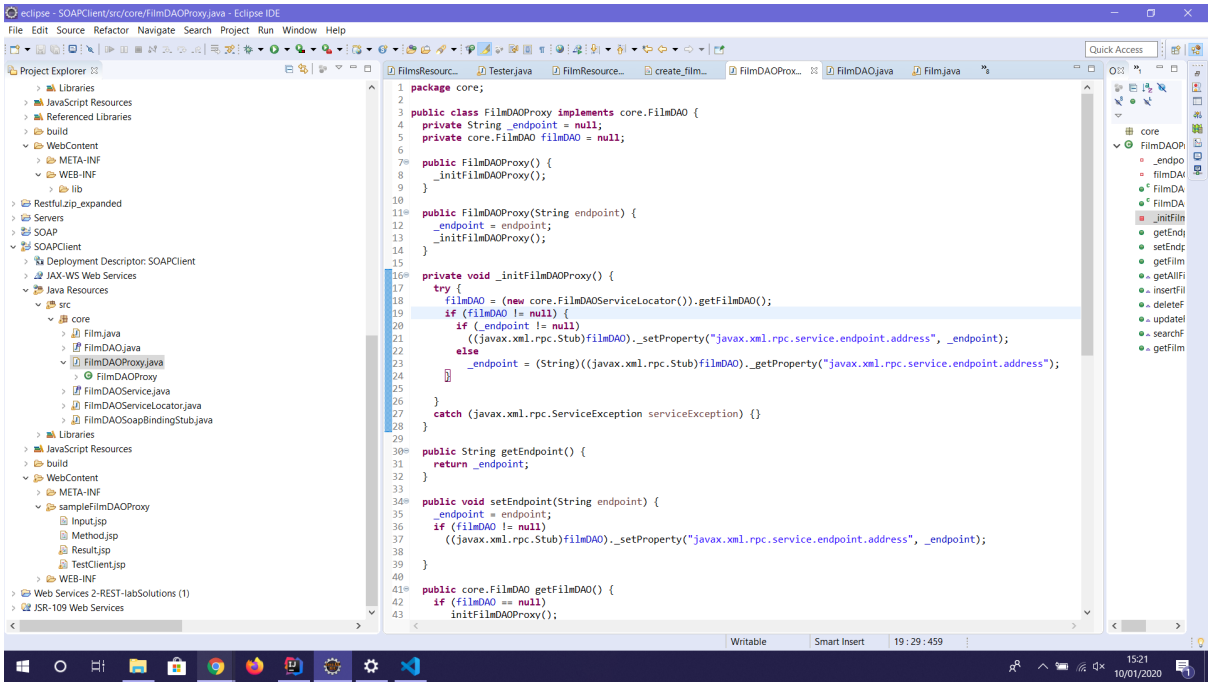

eclipse - SOAPClient/src/core/Film.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Film.java
Writable
Smart Insert
197 : 21 : 4374

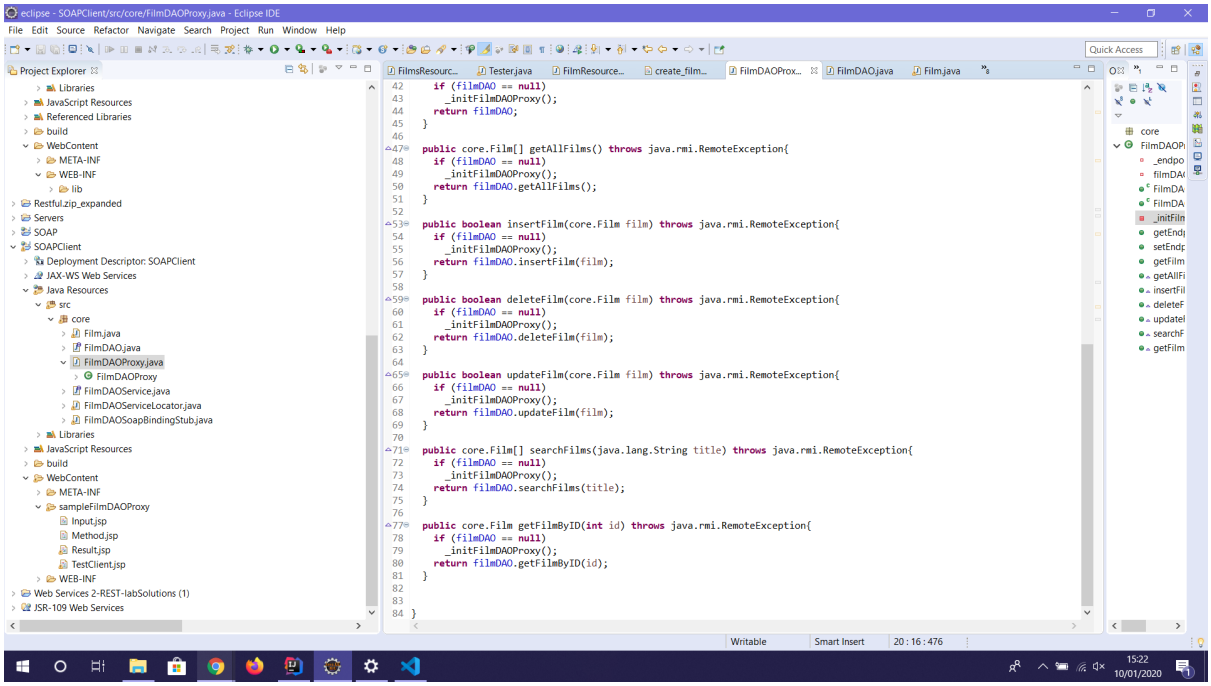

eclipse - SOAPClient/src/core/Film.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Film.java
Writable
Smart Insert
244 : 23 : 6683

FilmDAO

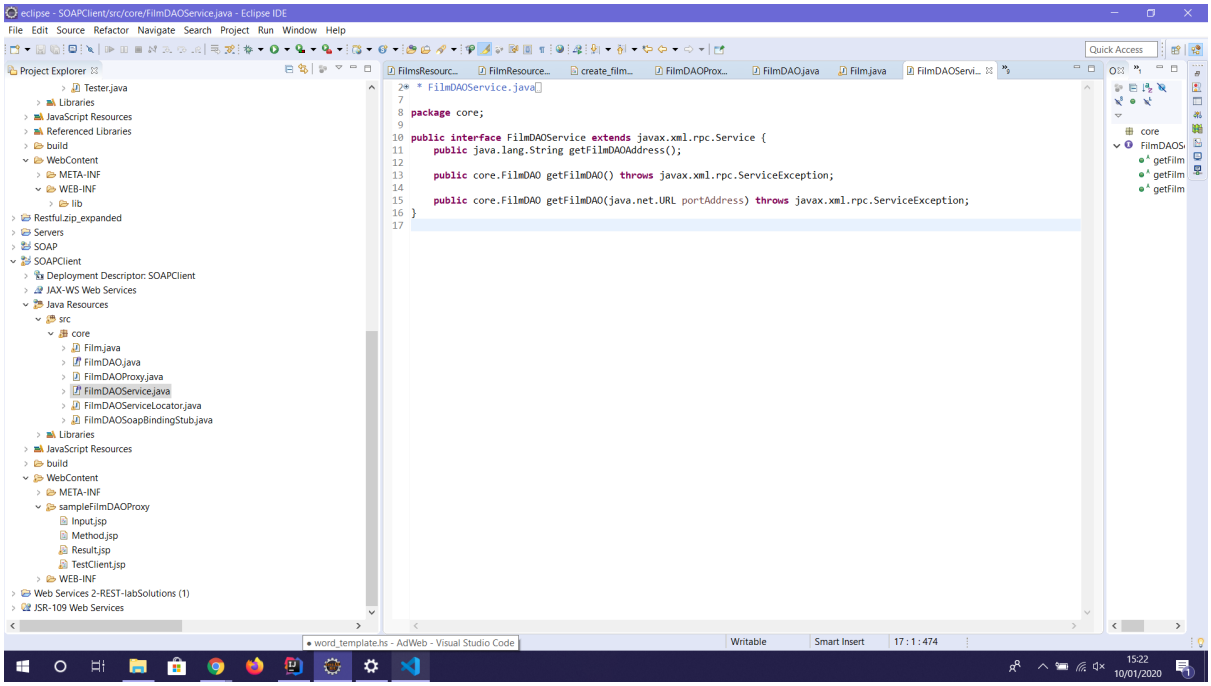


FilmDAOProxy

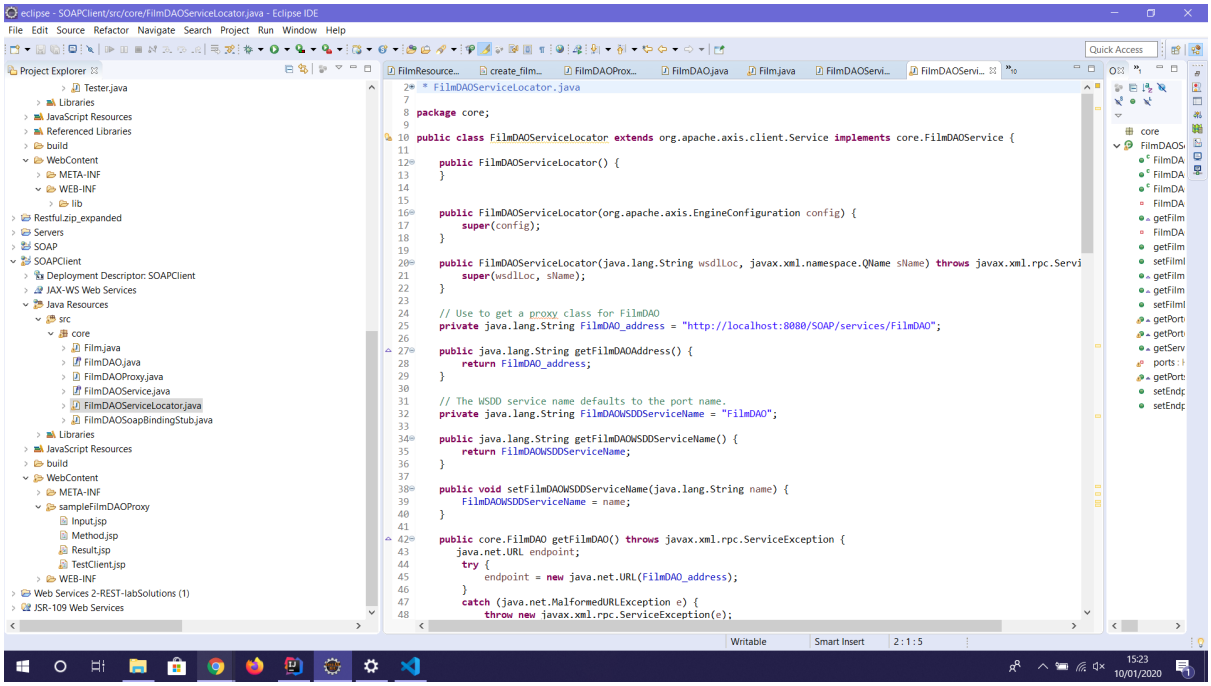


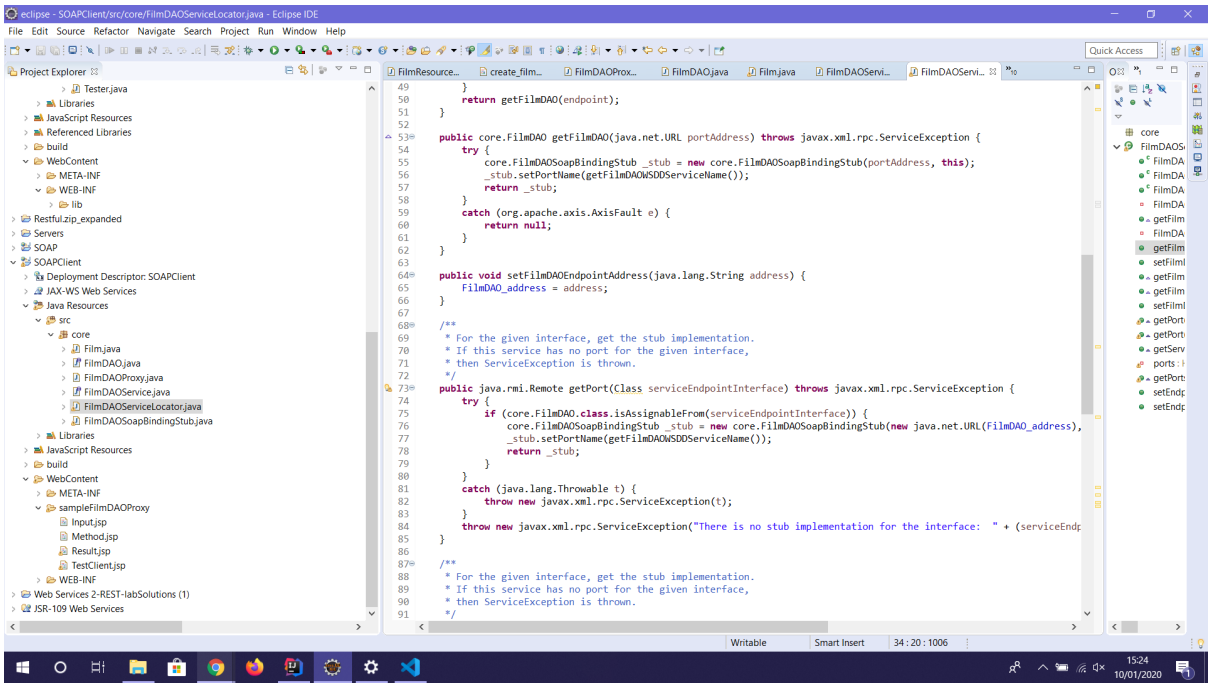


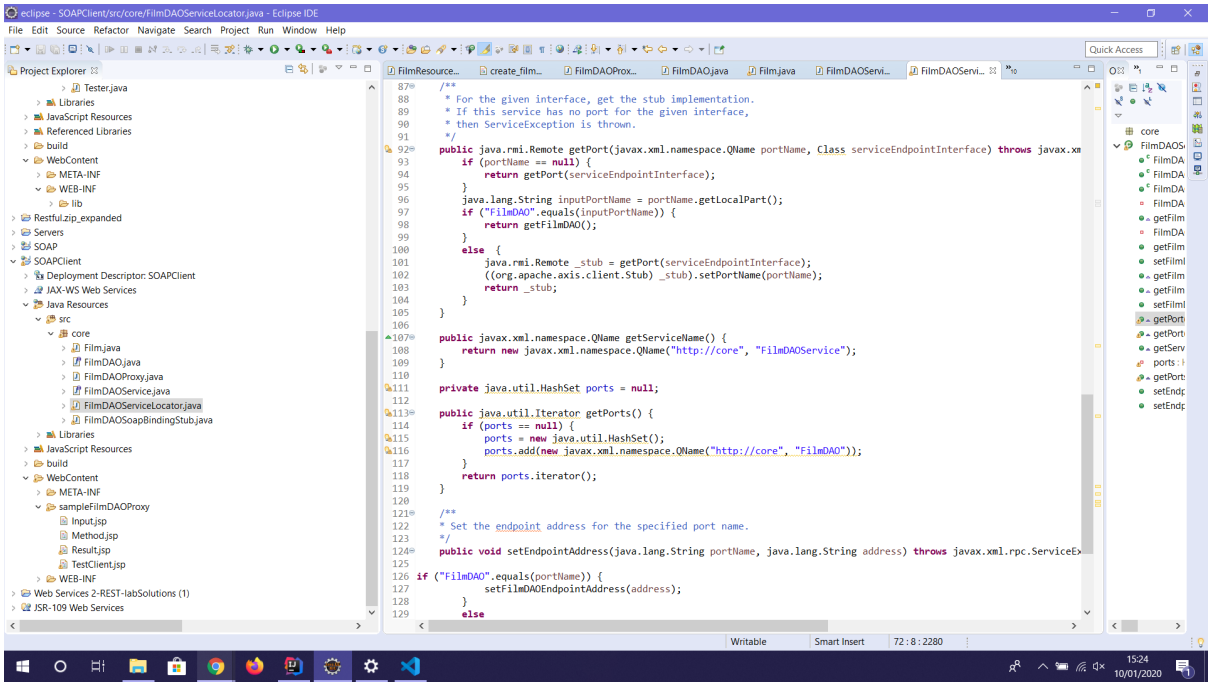
FilmDAOService

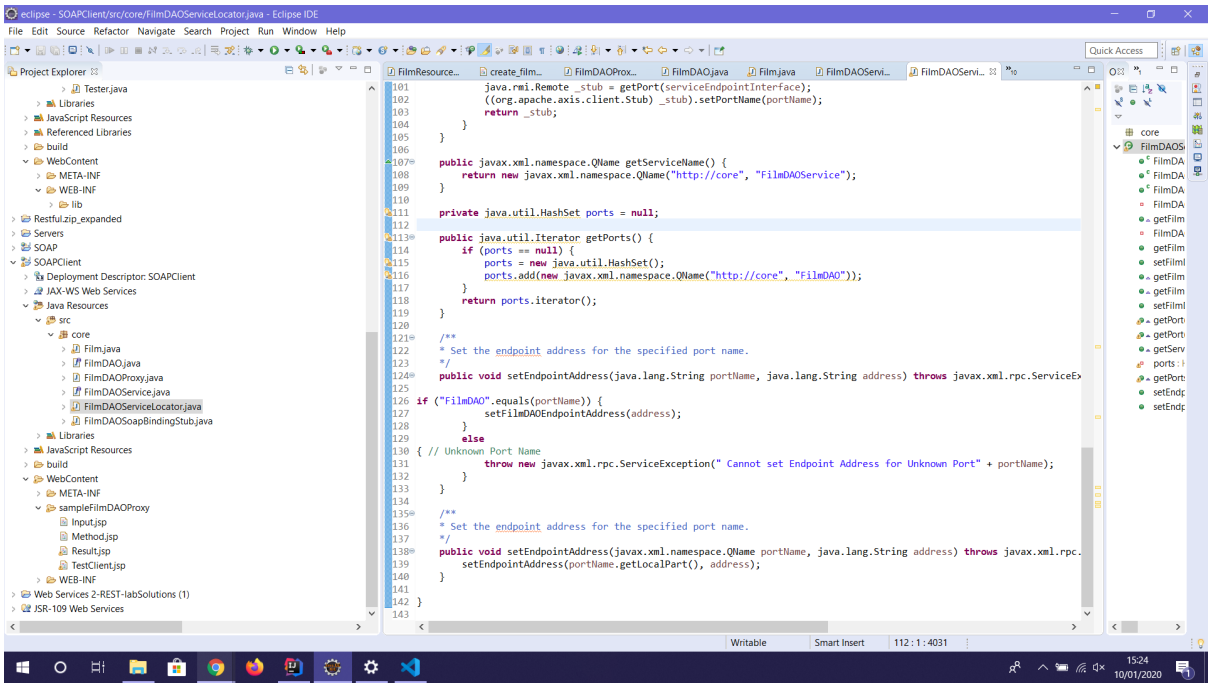


FilmDAOServiceLocator

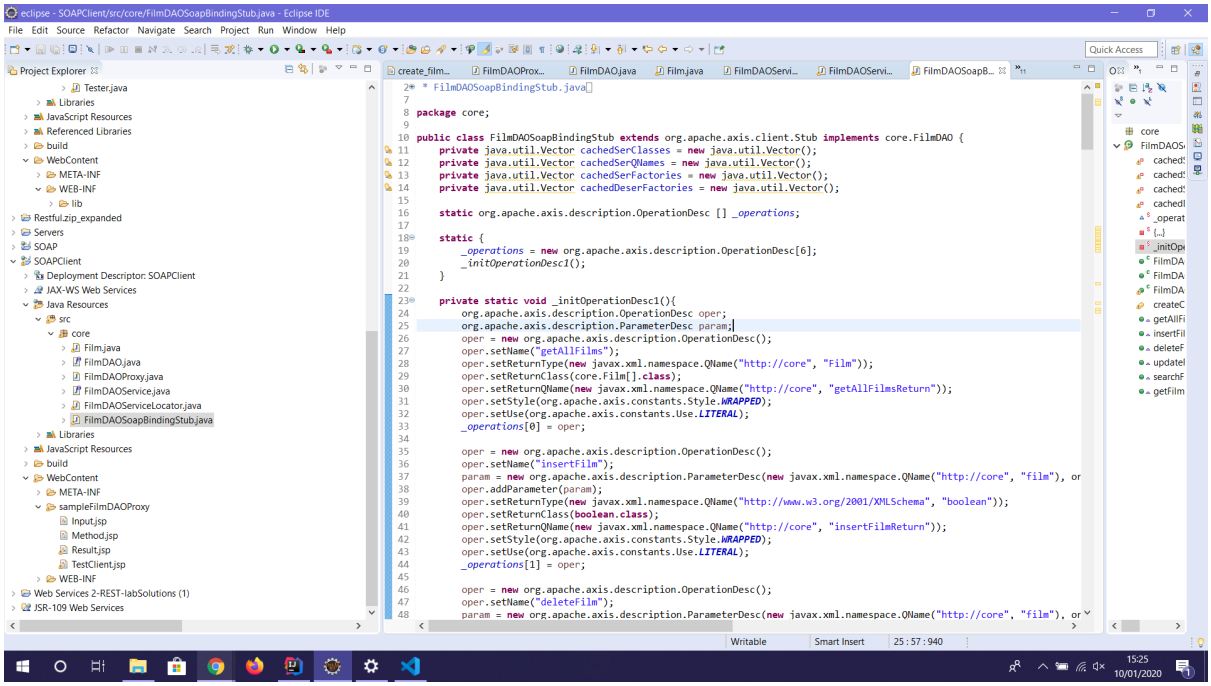

eclipse - SOAPClient/src/core/FilmDAOServiceLocator.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmResource...
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
FilmDAOServi...
FilmDAOServi...
Quick Access
* FilmDAOServiceLocator.java
package core;
public class FilmDAOServiceLocator extends org.apache.axis.client.Service implements core.FilmDAOService {
public FilmDAOServiceLocator() {
}
public FilmDAOServiceLocator(org.apache.axis.EngineConfiguration config) {
super(config);
}
public FilmDAOServiceLocator(java.lang.String wsdlLoc, javax.xml.namespace.QName sName) throws javax.xml.rpc.Servi
super(wsdlLoc, sName);
}
// Use to get a proxy class for FilmDAO
private java.lang.String FilmDAO_address = "http://localhost:8080/SOAP/services/FilmDAO";
public java.lang.String getFilmDAOAddress() {
return FilmDAO_address;
}
// The WSDD service name defaults to the port name.
private java.lang.String FilmDAOWSDDServiceName = "FilmDAO";
public java.lang.String getFilmDAOWSDDServiceName() {
return FilmDAOWSDDServiceName;
}
public void setFilmDAOWSDDServiceName(java.lang.String name) {
FilmDAOWSDDServiceName = name;
}
public core.FilmDAO getFilmDAO() throws javax.xml.rpc.ServiceException {
java.net.URL endpoint;
try {
endpoint = new java.net.URL(FilmDAO_address);
}
catch (java.net.MalformedURLException e) {
throw new javax.xml.rpc.ServiceException(e);
Writable
Smart Insert
2 : 1 : 5
15:23
10/01/2020

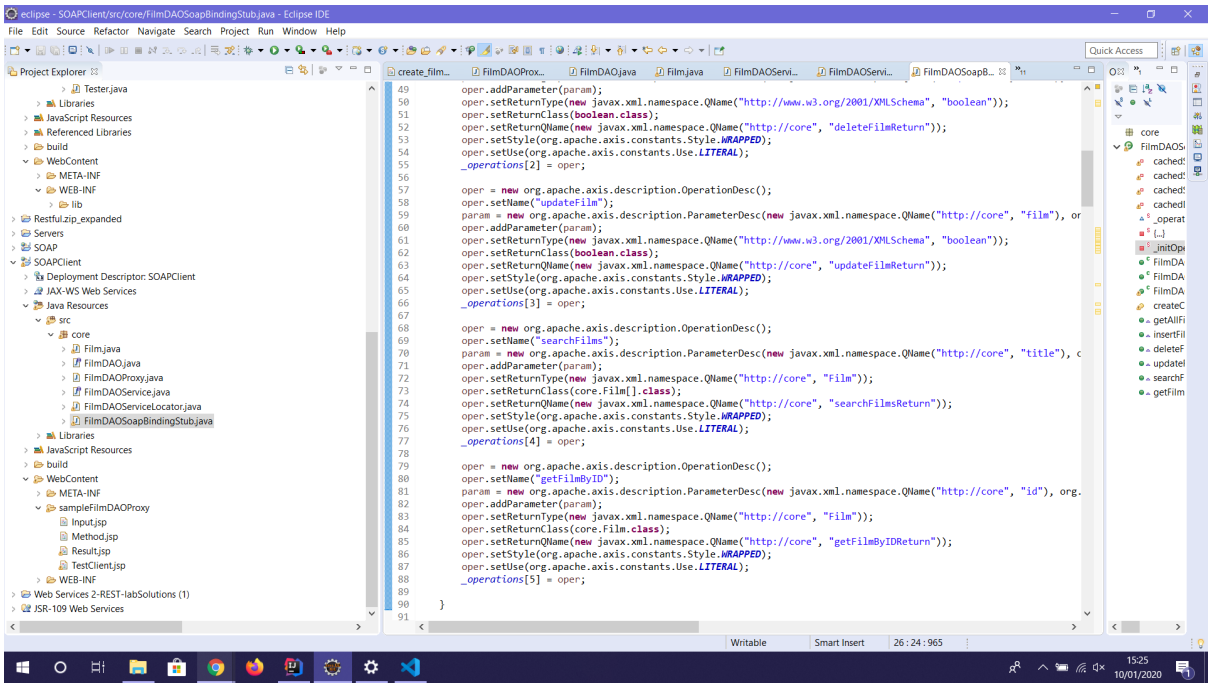

eclipse - SOAPClient/src/core/FilmDAOServiceLocator.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Quick Access
}
return getFilmDAO(endpoint);
}
public core.FilmDAO getFilmDAO(java.net.URL portAddress) throws javax.xml.rpc.ServiceException {
try {
core.FilmDAOSoapBindingStub _stub = new core.FilmDAOSoapBindingStub(portAddress, this);
_stub.setPortName(getFilmDAOWSDDServiceName());
return _stub;
}
catch (org.apache.axis.AxisFault e) {
return null;
}
}
public void setFilmDAOEndpointAddress(java.lang.String address) {
FilmDAO_address = address;
}
/**
* For the given interface, get the stub implementation.
* If this service has no port for the given interface,
* then ServiceException is thrown.
*/
public java.rmi.Remote getPort(Class serviceEndpointInterface) throws javax.xml.rpc.ServiceException {
try {
if (core.FilmDAO.class.isAssignableFrom(serviceEndpointInterface)) {
core.FilmDAOSoapBindingStub _stub = new core.FilmDAOSoapBindingStub(new java.net.URL(FilmDAO_address),
_stub.setPortName(getFilmDAOWSDDServiceName());
return _stub;
}
}
catch (java.lang.Throwable t) {
throw new javax.xml.rpc.ServiceException(t);
}
throw new javax.xml.rpc.ServiceException("There is no stub implementation for the interface: " + (serviceEndp
}
/**
* For the given interface, get the stub implementation.
* If this service has no port for the given interface,
* then ServiceException is thrown.
*/
Writable
Smart Insert
34 : 20 : 1006
15:24
10/01/2020

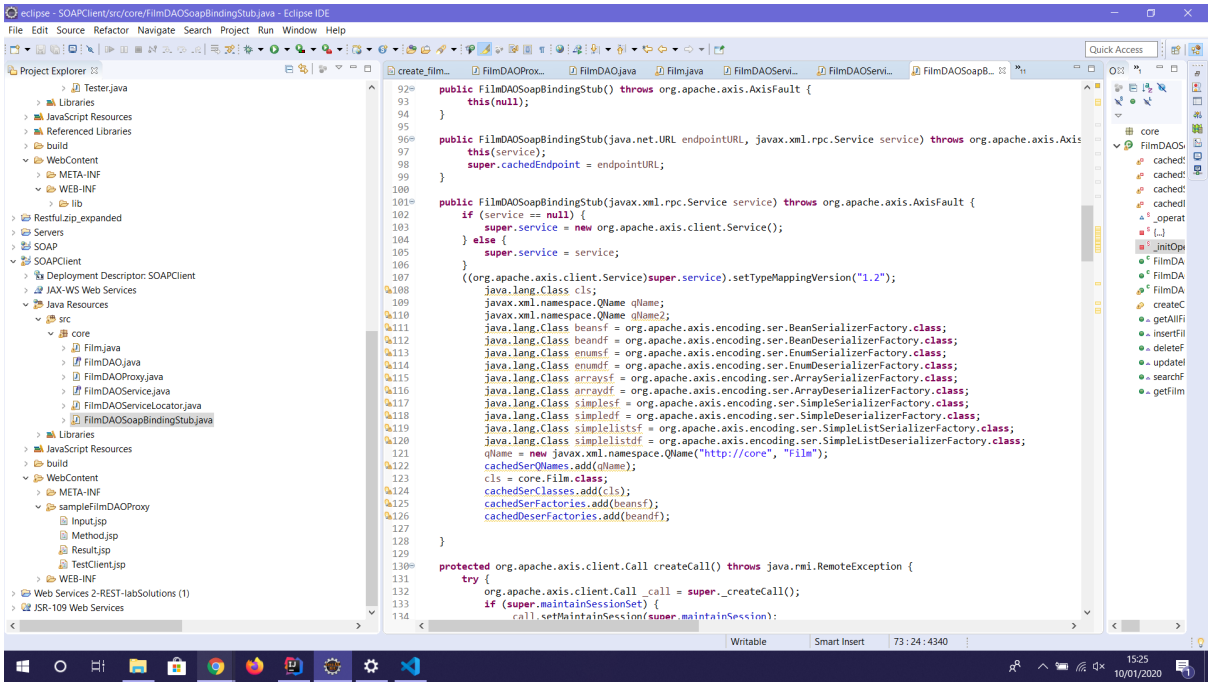


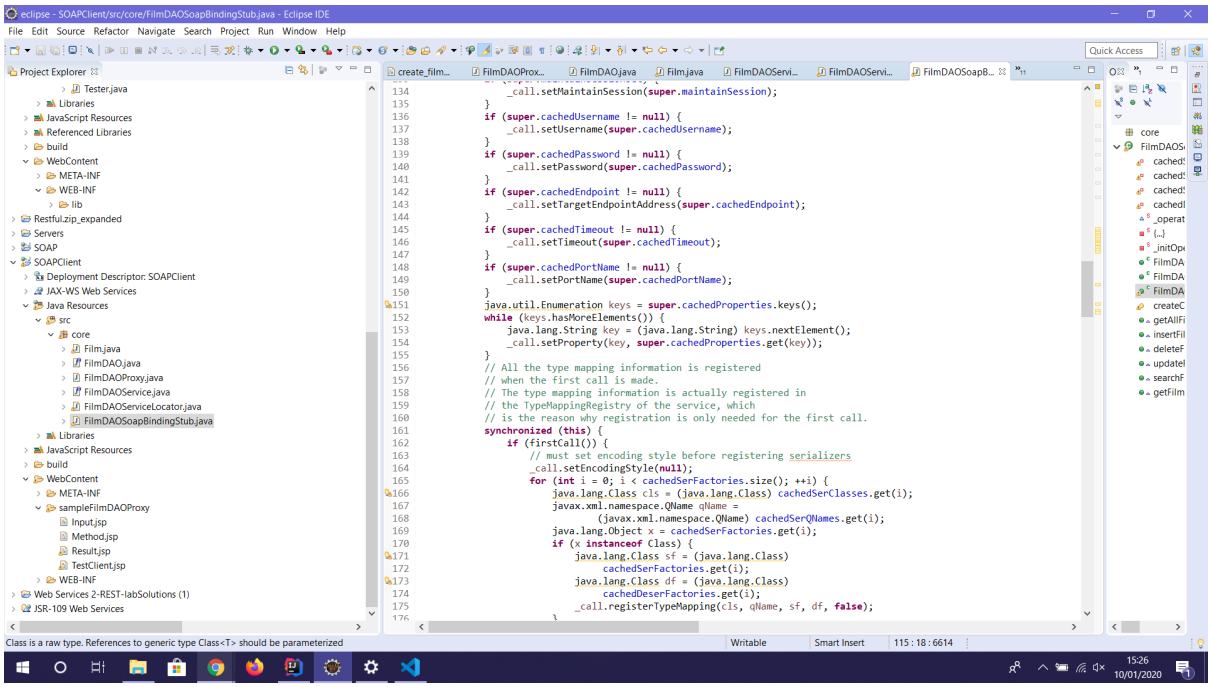


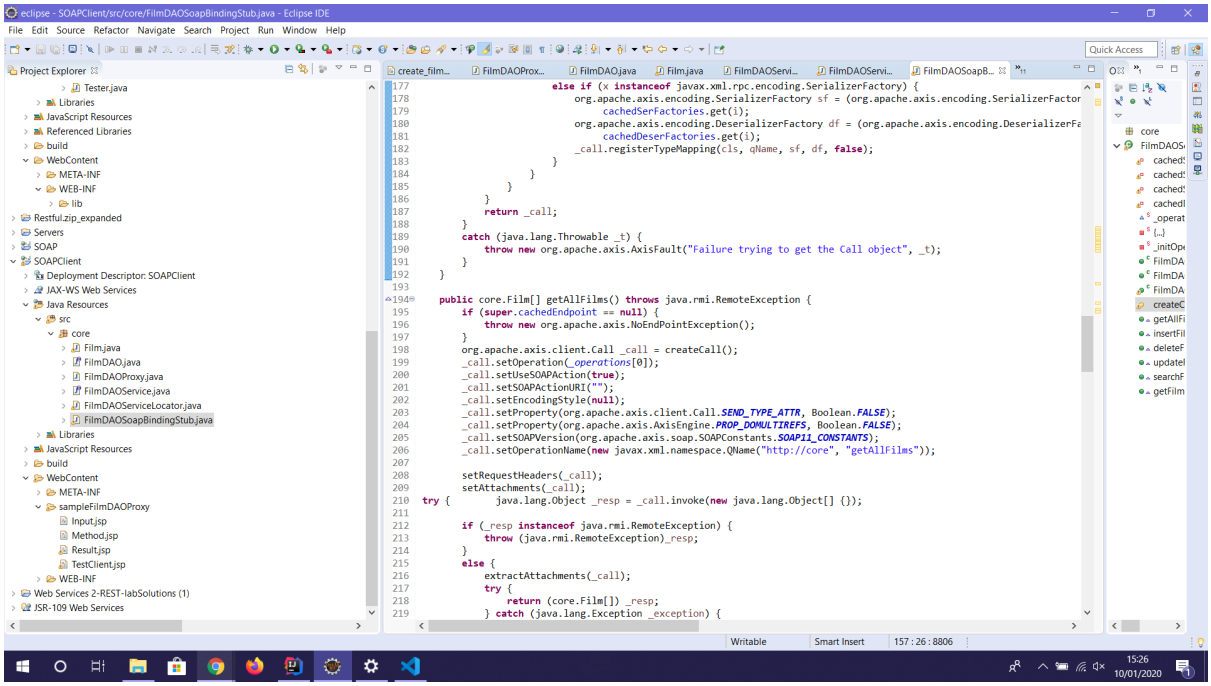
FilmDAOSoapBindingStub

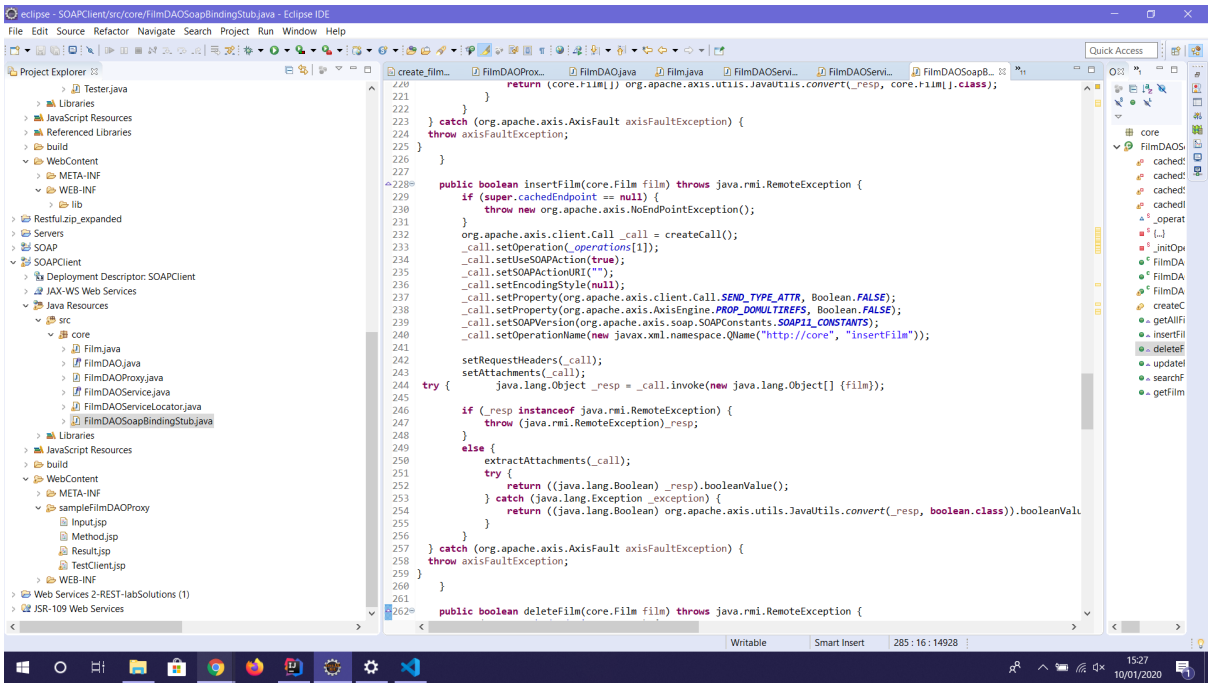

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
FilmDAOSoapBindingStub.java
* FilmDAOSoapBindingStub.java
package core;
public class FilmDAOSoapBindingStub extends org.apache.axis.client.Stub implements core.FilmDAO {
private java.util.Vector cachedSerClasses = new java.util.Vector();
private java.util.Vector cachedSerQNames = new java.util.Vector();
private java.util.Vector cachedSerFactories = new java.util.Vector();
private java.util.Vector cachedDeserFactories = new java.util.Vector();
static org.apache.axis.description.OperationDesc [] _operations;
static {
_operations = new org.apache.axis.description.OperationDesc[6];
_initOperationDesc1();
}
private static void _initOperationDesc1(){
org.apache.axis.description.OperationDesc oper;
org.apache.axis.description.ParameterDesc param;
oper = new org.apache.axis.description.OperationDesc();
oper.setName("getAllFilms");
oper.setReturnType(new javax.xml.namespace.QName("http://core", "Film"));
oper.setReturnClass(core.Film[].class);
oper.setReturnQName(new javax.xml.namespace.QName("http://core", "getAllFilmsReturn"));
oper.setStyle(org.apache.axis.constants.Style.WRAPPED);
oper.setUse(org.apache.axis.constants.Use.LITERAL);
_operations[0] = oper;
oper = new org.apache.axis.description.OperationDesc();
oper.setName("insertFilm");
param = new org.apache.axis.description.ParameterDesc(new javax.xml.namespace.QName("http://core", "film"), or
oper.addParameter(param);
oper.setReturnType(new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "boolean"));
oper.setReturnClass(boolean.class);
oper.setReturnQName(new javax.xml.namespace.QName("http://core", "insertFilmReturn"));
oper.setStyle(org.apache.axis.constants.Style.WRAPPED);
oper.setUse(org.apache.axis.constants.Use.LITERAL);
_operations[1] = oper;
oper = new org.apache.axis.description.OperationDesc();
oper.setName("deleteFilm");
param = new org.apache.axis.description.ParameterDesc(new javax.xml.namespace.QName("http://core", "film"), or
Writable
Smart Insert
25 : 57 : 940
15:25
10/01/2020

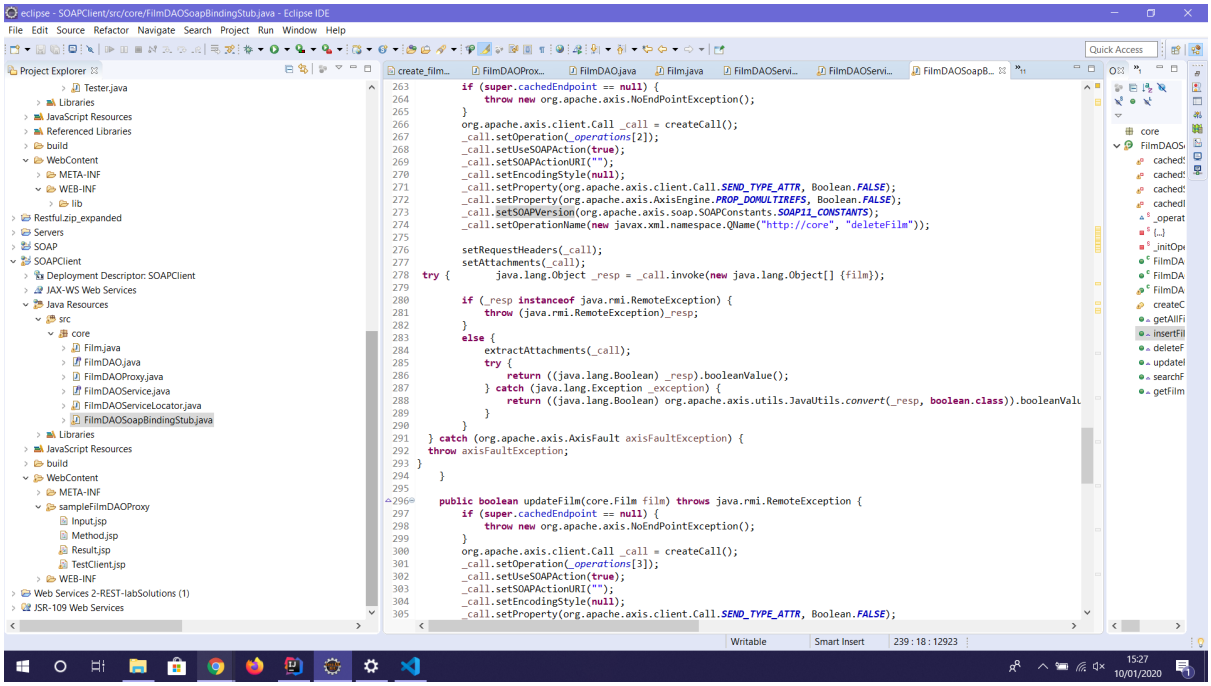

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
oper.addParameter(param);
oper.setReturnType(new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "boolean"));
oper.setReturnClass(boolean.class);
oper.setReturnQName(new javax.xml.namespace.QName("http://core", "deleteFilmReturn"));
oper.setStyle(org.apache.axis.constants.Style.WRAPPED);
oper.setUse(org.apache.axis.constants.Use.LITERAL);
_operations[2] = oper;
oper = new org.apache.axis.description.OperationDesc();
oper.setName("updateFilm");
param = new org.apache.axis.description.ParameterDesc(new javax.xml.namespace.QName("http://core", "film"), or
oper.addParameter(param);
oper.setReturnType(new javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "boolean"));
oper.setReturnClass(boolean.class);
oper.setReturnQName(new javax.xml.namespace.QName("http://core", "updateFilmReturn"));
oper.setStyle(org.apache.axis.constants.Style.WRAPPED);
oper.setUse(org.apache.axis.constants.Use.LITERAL);
_operations[3] = oper;
oper = new org.apache.axis.description.OperationDesc();
oper.setName("searchFilms");
param = new org.apache.axis.description.ParameterDesc(new javax.xml.namespace.QName("http://core", "title"), c
oper.addParameter(param);
oper.setReturnType(new javax.xml.namespace.QName("http://core", "Film"));
oper.setReturnClass(core.Film[].class);
oper.setReturnQName(new javax.xml.namespace.QName("http://core", "searchFilmsReturn"));
oper.setStyle(org.apache.axis.constants.Style.WRAPPED);
oper.setUse(org.apache.axis.constants.Use.LITERAL);
_operations[4] = oper;
oper = new org.apache.axis.description.OperationDesc();
oper.setName("getFilmByID");
param = new org.apache.axis.description.ParameterDesc(new javax.xml.namespace.QName("http://core", "id"), org.
oper.addParameter(param);
oper.setReturnType(new javax.xml.namespace.QName("http://core", "Film"));
oper.setReturnClass(core.Film.class);
oper.setReturnQName(new javax.xml.namespace.QName("http://core", "getFilmByIDReturn"));
oper.setStyle(org.apache.axis.constants.Style.WRAPPED);
oper.setUse(org.apache.axis.constants.Use.LITERAL);
_operations[5] = oper;
}
Writable
Smart Insert
26 : 24 : 965
15:25
10/01/2020

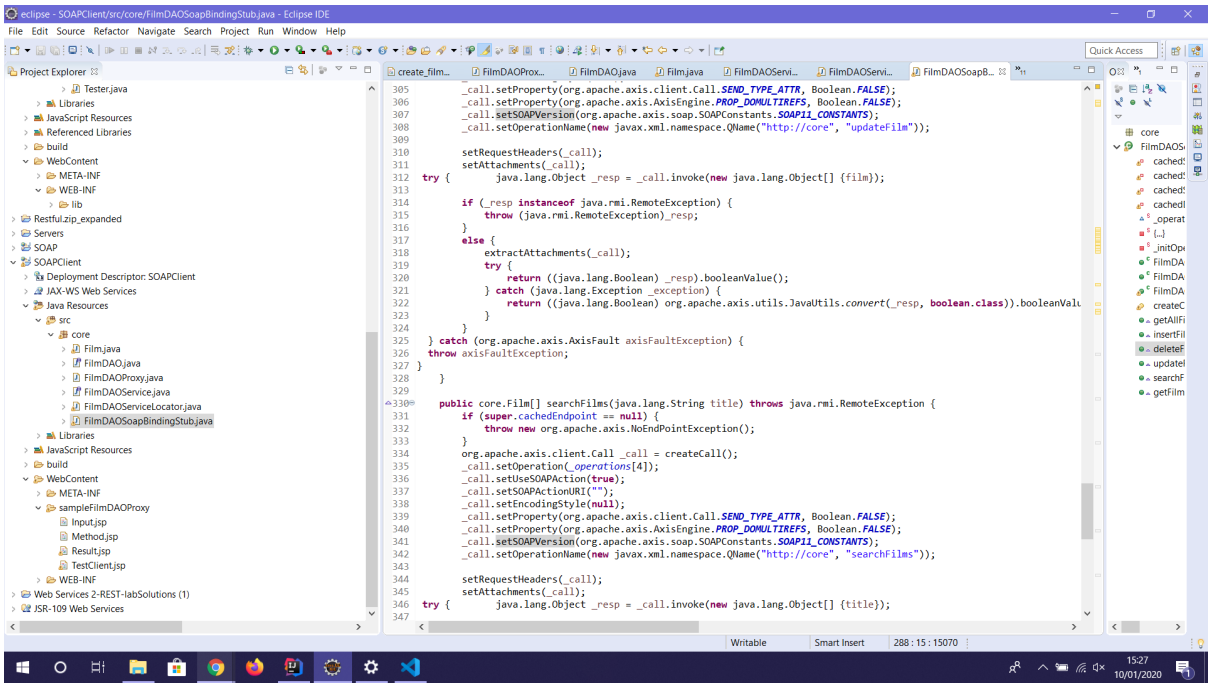

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
public FilmDAOSoapBindingStub() throws org.apache.axis.AxisFault {
this(null);
}
public FilmDAOSoapBindingStub(java.net.URL endpointURL, javax.xml.rpc.Service service) throws org.apache.axis.Axis
this(service);
super.cachedEndpoint = endpointURL;
}
public FilmDAOSoapBindingStub(javax.xml.rpc.Service service) throws org.apache.axis.AxisFault {
if (service == null) {
super.service = new org.apache.axis.client.Service();
} else {
super.service = service;
}
((org.apache.axis.client.Service)super.service).setTypeMappingVersion("1.2");
java.lang.Class cls;
javax.xml.namespace.QName qName;
javax.xml.namespace.QName qName2;
java.lang.Class beansf = org.apache.axis.encoding.ser.BeanSerializerFactory.class;
java.lang.Class beandf = org.apache.axis.encoding.ser.BeanDeserializerFactory.class;
java.lang.Class enumsf = org.apache.axis.encoding.ser.EnumSerializerFactory.class;
java.lang.Class enumdf = org.apache.axis.encoding.ser.EnumDeserializerFactory.class;
java.lang.Class arraysf = org.apache.axis.encoding.ser.ArraySerializerFactory.class;
java.lang.Class arraydf = org.apache.axis.encoding.ser.ArrayDeserializerFactory.class;
java.lang.Class simplesf = org.apache.axis.encoding.ser.SimpleSerializerFactory.class;
java.lang.Class simpledf = org.apache.axis.encoding.ser.SimpleDeserializerFactory.class;
java.lang.Class simplelistsf = org.apache.axis.encoding.ser.SimpleListSerializerFactory.class;
java.lang.Class simplelistdf = org.apache.axis.encoding.ser.SimpleListDeserializerFactory.class;
qName = new javax.xml.namespace.QName("http://core", "Film");
cachedSerQNames.add(qName);
cls = core.Film.class;
cachedSerClasses.add(cls);
cachedSerFactories.add(beansf);
cachedDeserFactories.add(beandf);
}
protected org.apache.axis.client.Call createCall() throws java.rmi.RemoteException {
try {
org.apache.axis.client.Call _call = super._createCall();
if (super.maintainSessionSet) {
Writable
Smart Insert
73 : 24 : 4340
15:25
10/01/2020

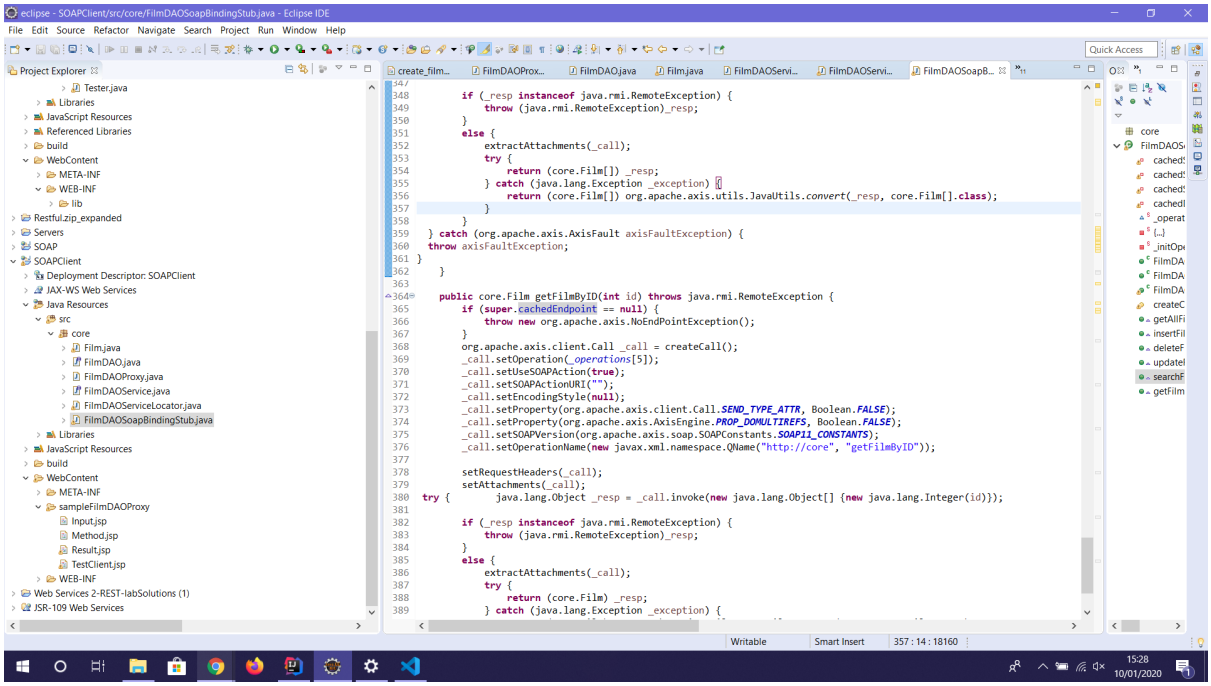

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
_call.setMaintainSession(super.maintainSession);
}
if (super.cachedUsername != null) {
_call.setUsername(super.cachedUsername);
}
if (super.cachedPassword != null) {
_call.setPassword(super.cachedPassword);
}
if (super.cachedEndpoint != null) {
_call.setTargetEndpointAddress(super.cachedEndpoint);
}
if (super.cachedTimeout != null) {
_call.setTimeout(super.cachedTimeout);
}
if (super.cachedPortName != null) {
_call.setPortName(super.cachedPortName);
}
java.util.Enumeration keys = super.cachedProperties.keys();
while (keys.hasMoreElements()) {
java.lang.String key = (java.lang.String) keys.nextElement();
_call.setProperty(key, super.cachedProperties.get(key));
}
// All the type mapping information is registered
// when the first call is made.
// The type mapping information is actually registered in
// the TypeMappingRegistry of the service, which
// is the reason why registration is only needed for the first call.
synchronized (this) {
if (firstCall()) {
// must set encoding style before registering serializers
_call.setEncodingStyle(null);
for (int i = 0; i < cachedSerFactories.size(); ++i) {
java.lang.Class cls = (java.lang.Class) cachedSerClasses.get(i);
javax.xml.namespace.QName qName =
(javax.xml.namespace.QName) cachedSerQNames.get(i);
java.lang.Object x = cachedSerFactories.get(i);
if (x instanceof Class) {
java.lang.Class sf = (java.lang.Class)
cachedSerFactories.get(i);
java.lang.Class df = (java.lang.Class)
cachedDeserFactories.get(i);
_call.registerTypeMapping(cls, qName, sf, df, false);
Class is a raw type. References to generic type Class<T> should be parameterized
Writable
Smart Insert
115 : 18 : 6614
15:26
10/01/2020

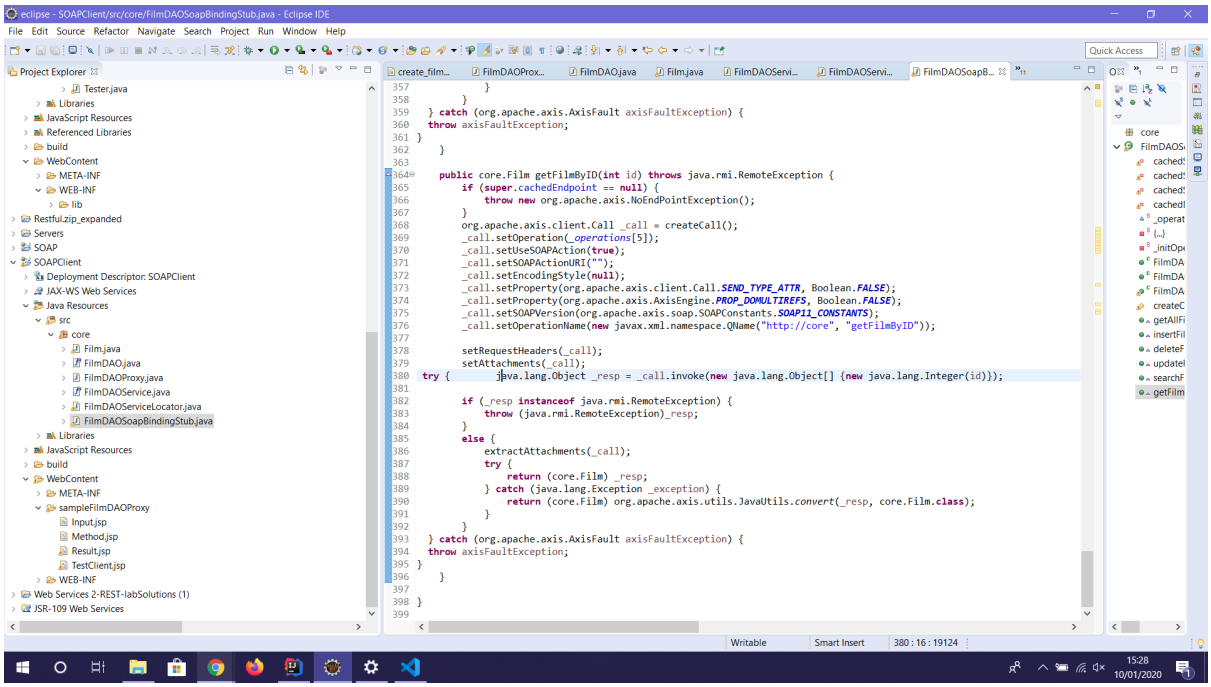

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
Quick Access
Writable
Smart Insert
157 : 26 : 8806
15:26
10/01/2020


eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Quick Access
Writable
Smart Insert
285 : 16 : 14928
15:27
10/01/2020

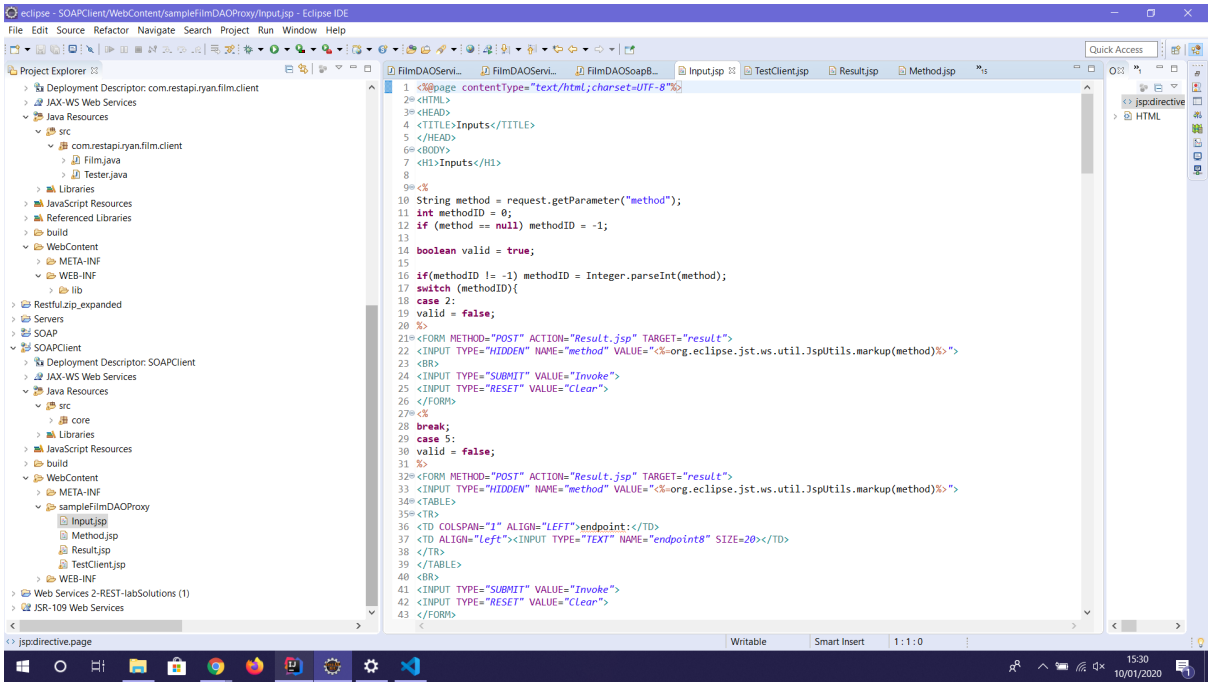

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Quick Access
263 if (super.cachedEndpoint == null) {
264 throw new org.apache.axis.NoEndPointException();
265 }
266 org.apache.axis.client.Call _call = createCall();
267 _call.setOperation(_operations[2]);
268 _call.setUseSOAPAction(true);
269 _call.setSOAPActionURI("");
270 _call.setEncodingStyle(null);
271 _call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
272 _call.setProperty(org.apache.axis.AxisEngine.PROP_DOMULTIREFS, Boolean.FALSE);
273 _call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
274 _call.setOperationName(new javax.xml.namespace.QName("http://core", "deleteFilm"));
275
276 setRequestHeaders(_call);
277 setAttachments(_call);
278 try { java.lang.Object _resp = _call.invoke(new java.lang.Object[] {film});
279
280 if (_resp instanceof java.rmi.RemoteException) {
281 throw (java.rmi.RemoteException)_resp;
282 }
283 else {
284 extractAttachments(_call);
285 try {
286 return ((java.lang.Boolean) _resp).booleanValue();
287 } catch (java.lang.Exception _exception) {
288 return ((java.lang.Boolean) org.apache.axis.utils.JavaUtils.convert(_resp, boolean.class)).booleanValu
289 }
290 }
291 } catch (org.apache.axis.AxisFault axisFaultException) {
292 throw axisFaultException;
293 }
294 }
295
296 public boolean updateFilm(core.Film film) throws java.rmi.RemoteException {
297 if (super.cachedEndpoint == null) {
298 throw new org.apache.axis.NoEndPointException();
299 }
300 org.apache.axis.client.Call _call = createCall();
301 _call.setOperation(_operations[3]);
302 _call.setUseSOAPAction(true);
303 _call.setSOAPActionURI("");
304 _call.setEncodingStyle(null);
305 _call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
Writable
Smart Insert
239 : 18 : 12923
15:27
10/01/2020


eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Quick Access
305 _call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
306 _call.setProperty(org.apache.axis.AxisEngine.PROP_DOMULTIREFS, Boolean.FALSE);
307 _call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
308 _call.setOperationName(new javax.xml.namespace.QName("http://core", "updateFilm"));
309
310 setRequestHeaders(_call);
311 setAttachments(_call);
312 try { java.lang.Object _resp = _call.invoke(new java.lang.Object[] {film});
313
314 if (_resp instanceof java.rmi.RemoteException) {
315 throw (java.rmi.RemoteException)_resp;
316 }
317 else {
318 extractAttachments(_call);
319 try {
320 return ((java.lang.Boolean) _resp).booleanValue();
321 } catch (java.lang.Exception _exception) {
322 return ((java.lang.Boolean) org.apache.axis.utils.JavaUtils.convert(_resp, boolean.class)).booleanValu
323 }
324 }
325 } catch (org.apache.axis.AxisFault axisFaultException) {
326 throw axisFaultException;
327 }
328 }
329
330 public core.Film[] searchFilms(java.lang.String title) throws java.rmi.RemoteException {
331 if (super.cachedEndpoint == null) {
332 throw new org.apache.axis.NoEndPointException();
333 }
334 org.apache.axis.client.Call _call = createCall();
335 _call.setOperation(_operations[4]);
336 _call.setUseSOAPAction(true);
337 _call.setSOAPActionURI("");
338 _call.setEncodingStyle(null);
339 _call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
340 _call.setProperty(org.apache.axis.AxisEngine.PROP_DOMULTIREFS, Boolean.FALSE);
341 _call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
342 _call.setOperationName(new javax.xml.namespace.QName("http://core", "searchFilms"));
343
344 setRequestHeaders(_call);
345 setAttachments(_call);
346 try { java.lang.Object _resp = _call.invoke(new java.lang.Object[] {title});
347
Writable
Smart Insert
288 : 15 : 15070
15:27
10/01/2020


eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Film.java
FilmDAO.java
FilmDAOProxy.java
FilmDAOService.java
FilmDAOServiceLocator.java
FilmDAOSoapBindingStub.java
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
Quick Access
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
FilmDAOServi...
FilmDAOSoapB...
if (_resp instanceof java.rmi.RemoteException) {
throw (java.rmi.RemoteException)_resp;
}
else {
extractAttachments(_call);
try {
return (core.Film[]) _resp;
} catch (java.lang.Exception _exception) {
return (core.Film[]) org.apache.axis.utils.JavaUtils.convert(_resp, core.Film[].class);
}
}
} catch (org.apache.axis.AxisFault axisFaultException) {
throw axisFaultException;
}
}
public core.Film getFilmByID(int id) throws java.rmi.RemoteException {
if (super.cachedEndpoint == null) {
throw new org.apache.axis.NoEndPointException();
}
org.apache.axis.client.Call _call = createCall();
_call.setOperation(_operations[5]);
_call.setUseSOAPAction(true);
_call.setSOAPActionURI("");
_call.setEncodingStyle(null);
_call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
_call.setProperty(org.apache.axis.AxisEngine.PROP_DOMULTIREFS, Boolean.FALSE);
_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
_call.setOperationName(new javax.xml.namespace.QName("http://core", "getFilmByID"));
setRequestHeaders(_call);
setAttachments(_call);
try {
java.lang.Object _resp = _call.invoke(new java.lang.Object[] {new java.lang.Integer(id)});
if (_resp instanceof java.rmi.RemoteException) {
throw (java.rmi.RemoteException)_resp;
}
else {
extractAttachments(_call);
try {
return (core.Film) _resp;
} catch (java.lang.Exception _exception) {
Writable
Smart Insert
357 : 14 : 18160
15:28
10/01/2020

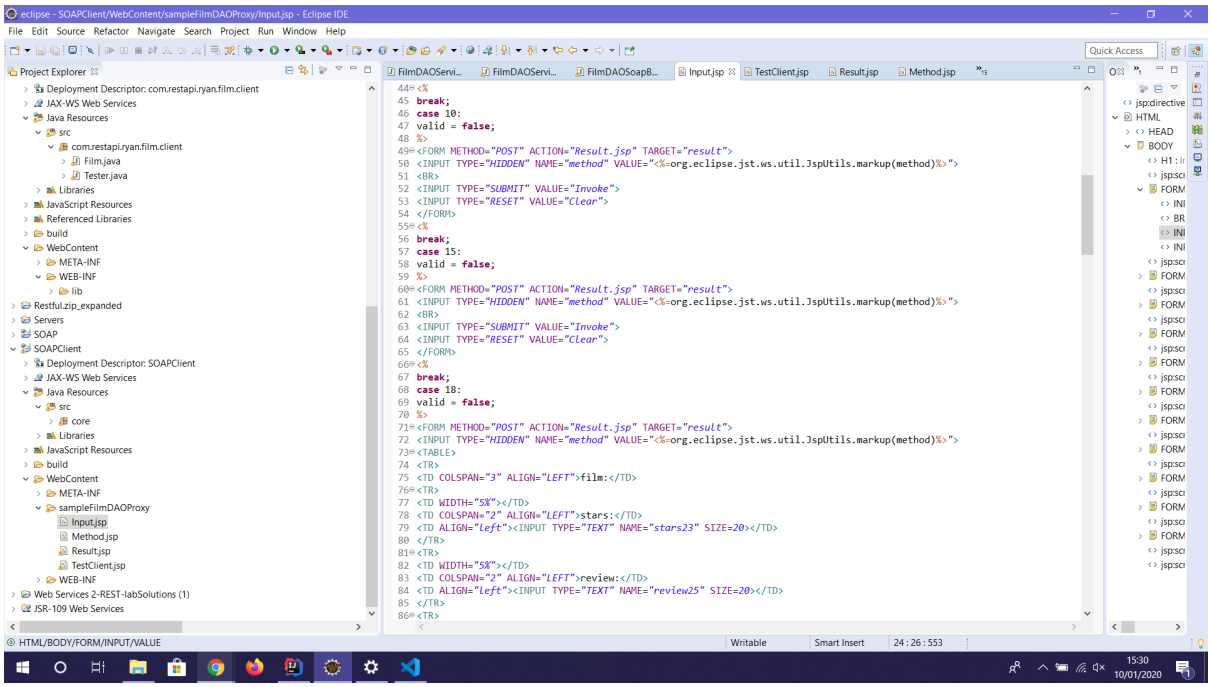

eclipse - SOAPClient/src/core/FilmDAOSoapBindingStub.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Quick Access
create_film...
FilmDAOProx...
FilmDAO.java
Film.java
FilmDAOServi...
FilmDAOSoapB...
}
} catch (org.apache.axis.AxisFault axisFaultException) {
throw axisFaultException;
}
}
public core.Film getFilmByID(int id) throws java.rmi.RemoteException {
if (super.cachedEndpoint == null) {
throw new org.apache.axis.NoEndPointException();
}
org.apache.axis.client.Call _call = createCall();
_call.setOperation(_operations[5]);
_call.setUseSOAPAction(true);
_call.setSOAPActionURI("");
_call.setEncodingStyle(null);
_call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR, Boolean.FALSE);
_call.setProperty(org.apache.axis.AxisEngine.PROP_DOMULTIREFS, Boolean.FALSE);
_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
_call.setOperationName(new javax.xml.namespace.QName("http://core", "getFilmByID"));
setRequestHeaders(_call);
setAttachments(_call);
try {
java.lang.Object _resp = _call.invoke(new java.lang.Object[] {new java.lang.Integer(id)});
if (_resp instanceof java.rmi.RemoteException) {
throw (java.rmi.RemoteException)_resp;
}
else {
extractAttachments(_call);
try {
return (core.Film) _resp;
} catch (java.lang.Exception _exception) {
return (core.Film) org.apache.axis.utils.JavaUtils.convert(_resp, core.Film.class);
}
}
} catch (org.apache.axis.AxisFault axisFaultException) {
throw axisFaultException;
}
}
}
Writable
Smart Insert
380 : 16 : 19124
15:28
10/01/2020

Input


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Input.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
<%@page contentType="text/html;charset=UTF-8"%>
<HTML>
<HEAD>
<TITLE>Inputs</TITLE>
</HEAD>
<BODY>
<H1>Inputs</H1>
<%
String method = request.getParameter("method");
int methodID = 0;
if (method == null) methodID = -1;
boolean valid = true;
if(methodID != -1) methodID = Integer.parseInt(method);
switch (methodID){
case 2:
valid = false;
%>
<FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
<INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">

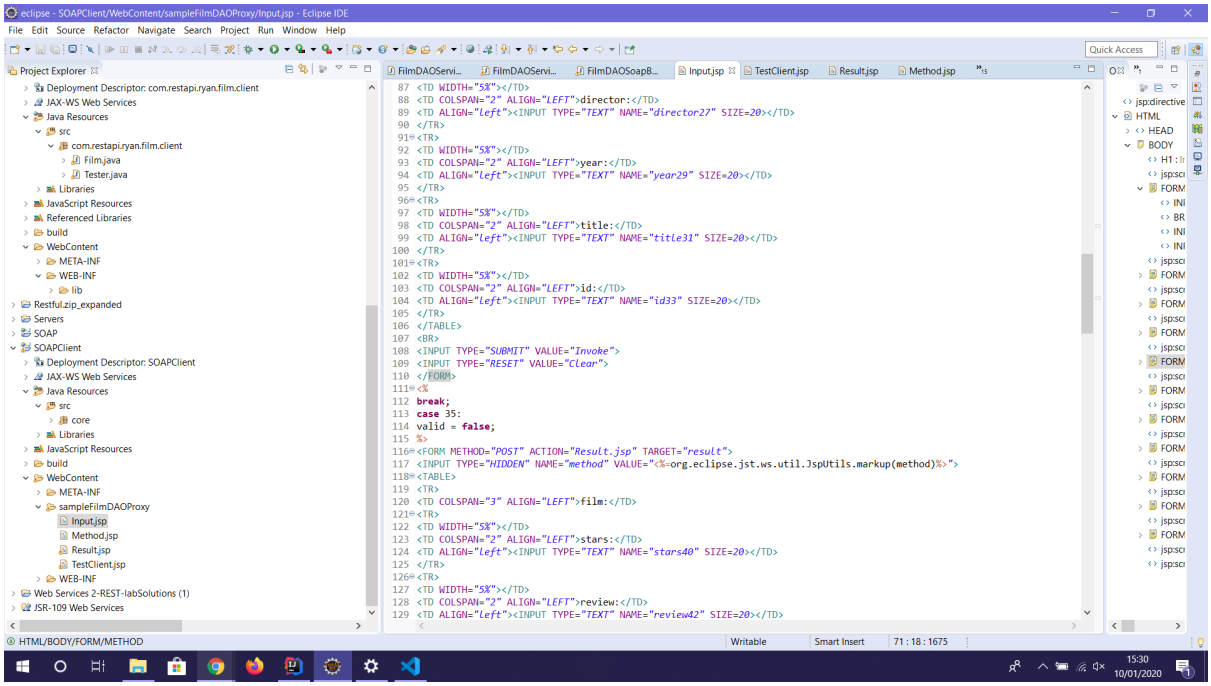
<INPUT TYPE="SUBMIT" VALUE="Invoke">
<INPUT TYPE="RESET" VALUE="Clear">
</FORM>
<%
break;
case 5:
valid = false;
%>
<FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
<INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
<TABLE>
<TR>
<TD COLSPAN="1" ALIGN="LEFT">endpoint:</TD>
<TD ALIGN="left"><INPUT TYPE="TEXT" NAME="endpoint8" SIZE=20></TD>
</TR>
</TABLE>

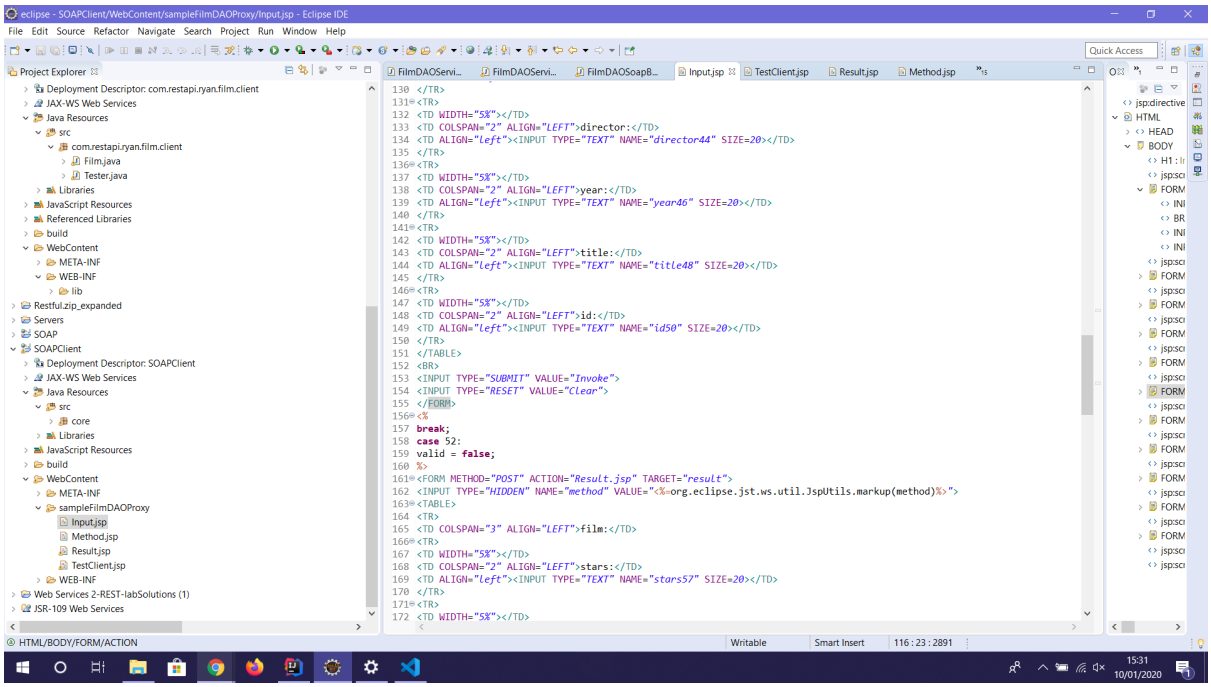
<INPUT TYPE="SUBMIT" VALUE="Invoke">
<INPUT TYPE="RESET" VALUE="Clear">
</FORM>
jsp:directive.page
Writable
Smart Insert
1 : 1 : 0
15:30
10/01/2020

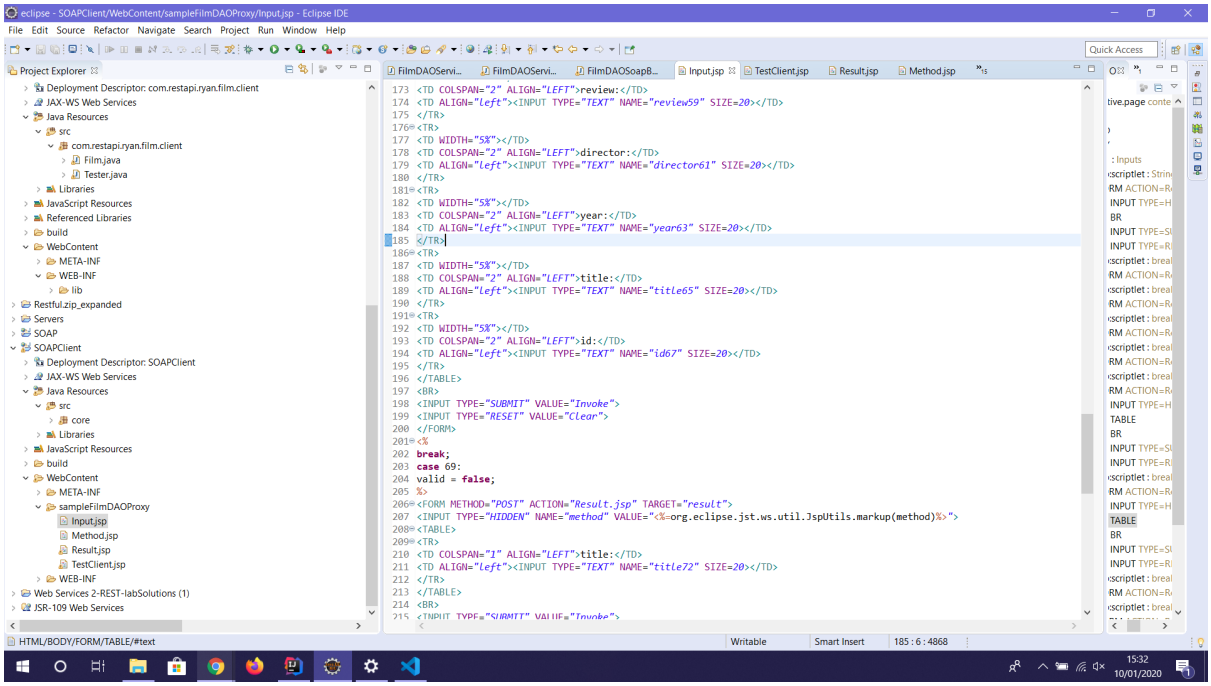

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Input.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Input.jsp
<%
break;
case 10:
valid = false;
%>
<FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
<INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">

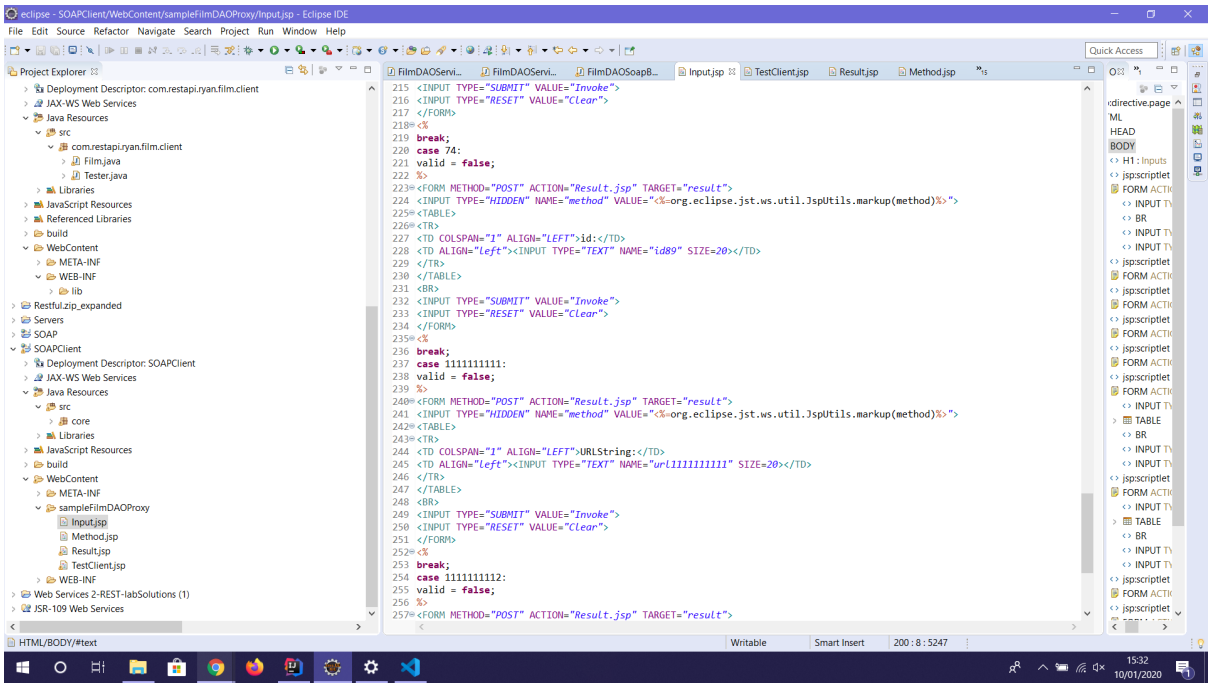
<INPUT TYPE="SUBMIT" VALUE="Invoke">
<INPUT TYPE="RESET" VALUE="Clear">
</FORM>
<%
break;
case 15:
valid = false;
%>
<FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
<INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">

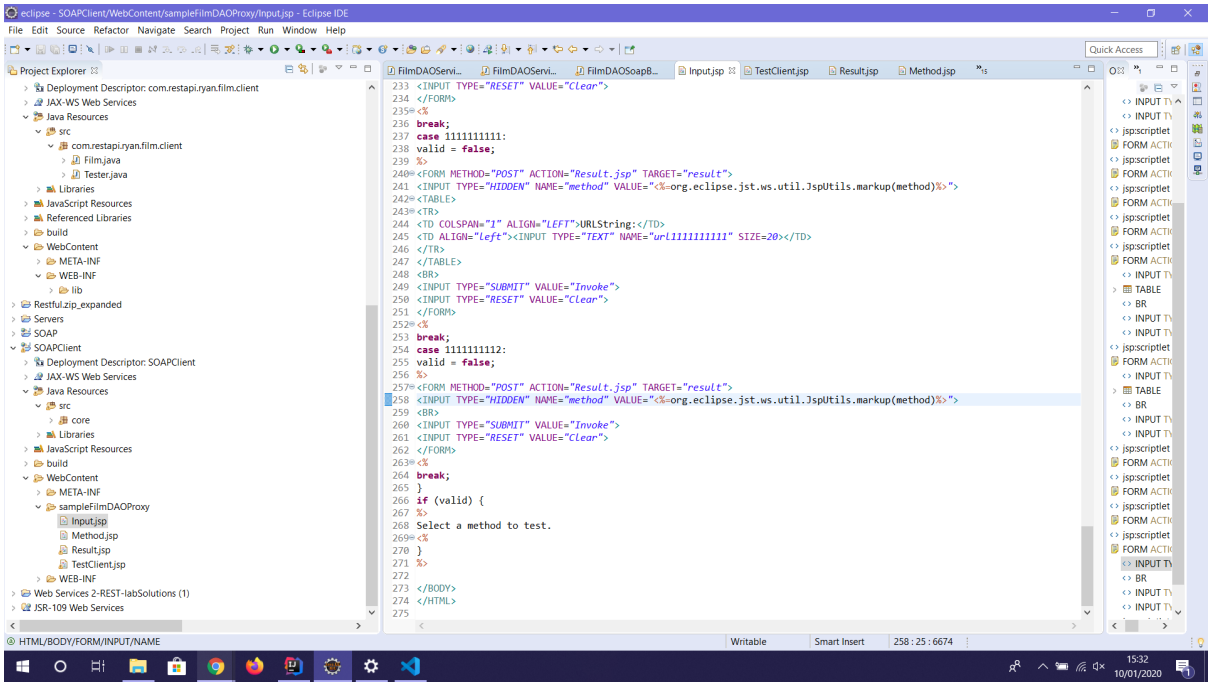
<INPUT TYPE="SUBMIT" VALUE="Invoke">
<INPUT TYPE="RESET" VALUE="Clear">
</FORM>
<%
break;
case 18:
valid = false;
%>
<FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
<INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
<TABLE>
<TR>
<TD COLSPAN="3" ALIGN="LEFT">film:</TD>
<TR>
<TD WIDTH="5%"></TD>
<TD COLSPAN="2" ALIGN="LEFT">stars:</TD>
<TD ALIGN="left"><INPUT TYPE="TEXT" NAME="stars23" SIZE=20></TD>
</TR>
<TR>
<TD WIDTH="5%"></TD>
<TD COLSPAN="2" ALIGN="LEFT">review:</TD>
<TD ALIGN="left"><INPUT TYPE="TEXT" NAME="review25" SIZE=20></TD>
</TR>
<TR>
HTML/BODY/FORM/INPUT/VALUE
Writable
Smart Insert
24 : 26 : 553
15:30
10/01/2020


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Input.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
87 <TD WIDTH="5%"></TD>
88 <TD COLSPAN="2" ALIGN="LEFT">director:</TD>
89 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="director27" SIZE=20></TD>
90 </TR>
91 <TR>
92 <TD WIDTH="5%"></TD>
93 <TD COLSPAN="2" ALIGN="LEFT">year:</TD>
94 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="year29" SIZE=20></TD>
95 </TR>
96 <TR>
97 <TD WIDTH="5%"></TD>
98 <TD COLSPAN="2" ALIGN="LEFT">title:</TD>
99 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="title31" SIZE=20></TD>
100 </TR>
101 <TR>
102 <TD WIDTH="5%"></TD>
103 <TD COLSPAN="2" ALIGN="LEFT">id:</TD>
104 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="id33" SIZE=20></TD>
105 </TR>
106 </TABLE>
107
108 <INPUT TYPE="SUBMIT" VALUE="Invoke">
109 <INPUT TYPE="RESET" VALUE="Clear">
110 </FORM>
111 <%
112 break;
113 case 35:
114 valid = false;
115 %>
116 <FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
117 <INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
118 <TABLE>
119 <TR>
120 <TD COLSPAN="3" ALIGN="LEFT">film:</TD>
121 <TR>
122 <TD WIDTH="5%"></TD>
123 <TD COLSPAN="2" ALIGN="LEFT">stars:</TD>
124 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="stars40" SIZE=20></TD>
125 </TR>
126 <TR>
127 <TD WIDTH="5%"></TD>
128 <TD COLSPAN="2" ALIGN="LEFT">review:</TD>
129 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="review42" SIZE=20></TD>
HTML/BODY/FORM/METHOD
Writable
Smart Insert
71 : 18 : 1675
15:30
10/01/2020

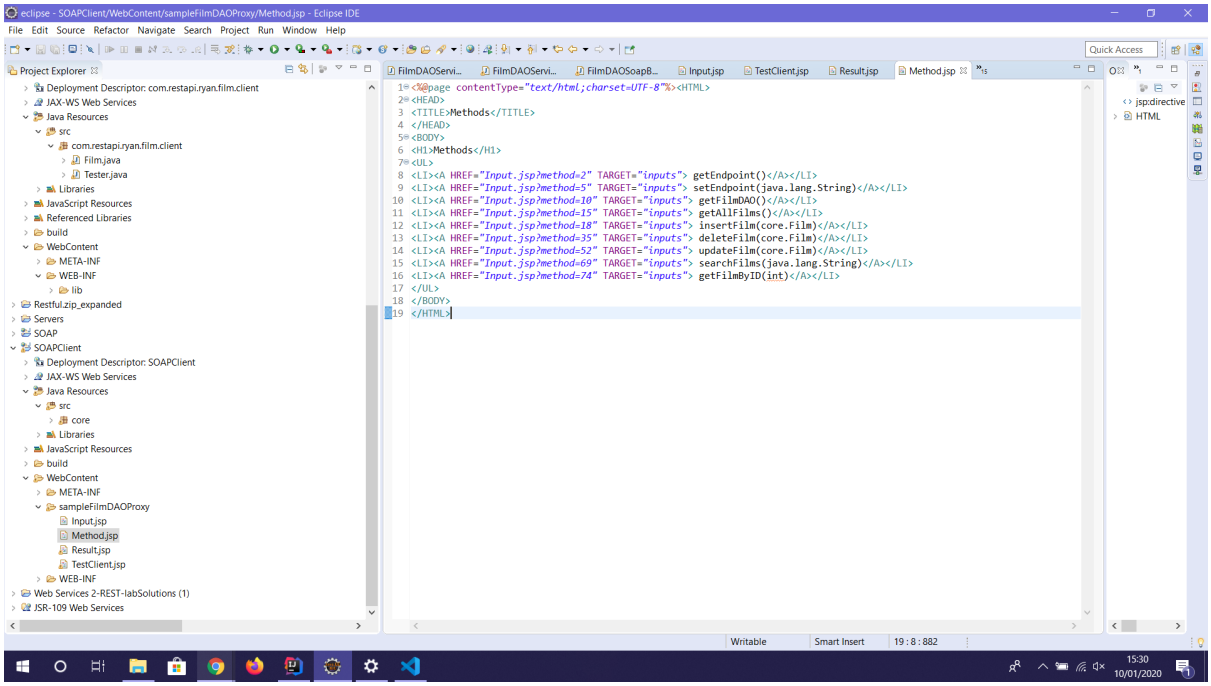

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Input.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
130 </TR>
131 <TR>
132 <TD WIDTH="5%"></TD>
133 <TD COLSPAN="2" ALIGN="LEFT">director:</TD>
134 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="director44" SIZE=20></TD>
135 </TR>
136 <TR>
137 <TD WIDTH="5%"></TD>
138 <TD COLSPAN="2" ALIGN="LEFT">year:</TD>
139 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="year46" SIZE=20></TD>
140 </TR>
141 <TR>
142 <TD WIDTH="5%"></TD>
143 <TD COLSPAN="2" ALIGN="LEFT">title:</TD>
144 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="title48" SIZE=20></TD>
145 </TR>
146 <TR>
147 <TD WIDTH="5%"></TD>
148 <TD COLSPAN="2" ALIGN="LEFT">id:</TD>
149 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="id50" SIZE=20></TD>
150 </TR>
151 </TABLE>
152
153 <INPUT TYPE="SUBMIT" VALUE="Invoke">
154 <INPUT TYPE="RESET" VALUE="Clear">
155 </FORM>
156 <%
157 break;
158 case 52:
159 valid = false;
160 %>
161 <FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
162 <INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
163 <TABLE>
164 <TR>
165 <TD COLSPAN="3" ALIGN="LEFT">film:</TD>
166 <TR>
167 <TD WIDTH="5%"></TD>
168 <TD COLSPAN="2" ALIGN="LEFT">stars:</TD>
169 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="stars57" SIZE=20></TD>
170 </TR>
171 <TR>
172 <TD WIDTH="5%"></TD>
HTML/BODY/FORM/ACTION
Writable
Smart Insert
116 : 23 : 2891
15:31
10/01/2020


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Input.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
173 <TD COLSPAN="2" ALIGN="LEFT">review:</TD>
174 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="review59" SIZE=20></TD>
175 </TR>
176 <TR>
177 <TD WIDTH="5%"></TD>
178 <TD COLSPAN="2" ALIGN="LEFT">director:</TD>
179 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="director61" SIZE=20></TD>
180 </TR>
181 <TR>
182 <TD WIDTH="5%"></TD>
183 <TD COLSPAN="2" ALIGN="LEFT">year:</TD>
184 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="year63" SIZE=20></TD>
185 </TR>
186 <TR>
187 <TD WIDTH="5%"></TD>
188 <TD COLSPAN="2" ALIGN="LEFT">title:</TD>
189 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="title65" SIZE=20></TD>
190 </TR>
191 <TR>
192 <TD WIDTH="5%"></TD>
193 <TD COLSPAN="2" ALIGN="LEFT">id:</TD>
194 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="id67" SIZE=20></TD>
195 </TR>
196 </TABLE>
197
198 <INPUT TYPE="SUBMIT" VALUE="Invoke">
199 <INPUT TYPE="RESET" VALUE="Clear">
200 </FORM>
201 <%
202 break;
203 case 69:
204 valid = false;
205 %>
206 <FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
207 <INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
208 <TABLE>
209 <TR>
210 <TD COLSPAN="1" ALIGN="LEFT">title:</TD>
211 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="title72" SIZE=20></TD>
212 </TR>
213 </TABLE>
214
HTML/BODY/FORM/TABLE/#text
Writable
Smart Insert
185 : 6 : 4868
15:32
10/01/2020

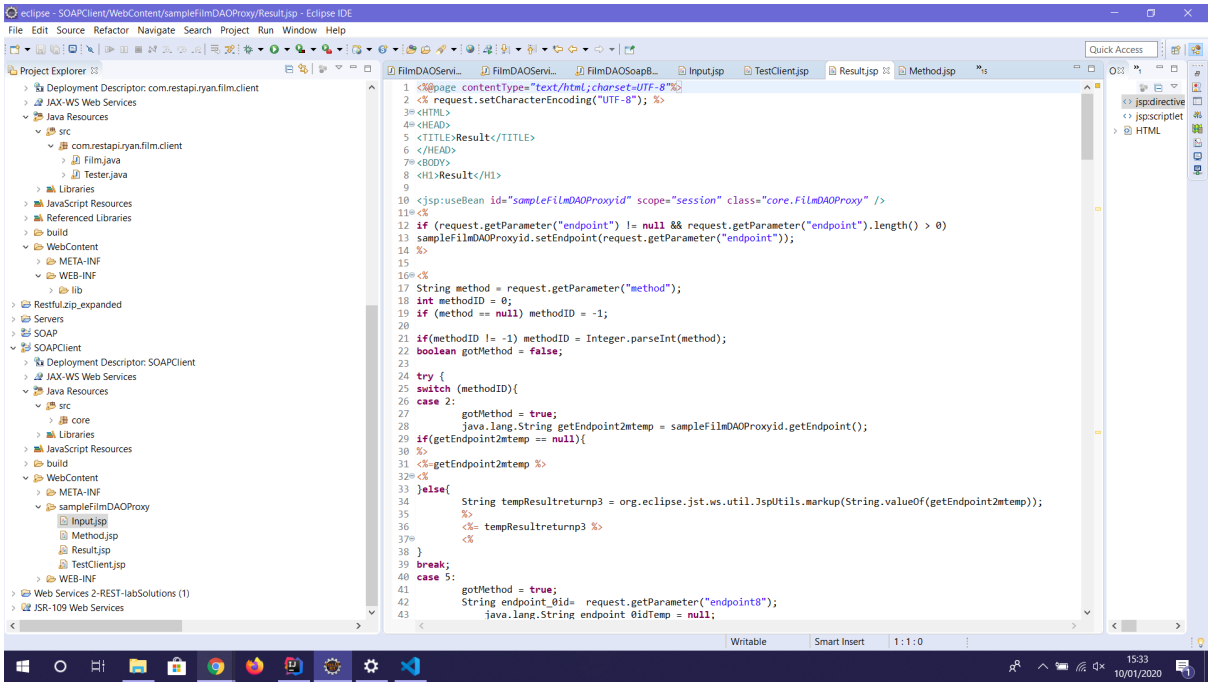

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Input.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
215 <INPUT TYPE="SUBMIT" VALUE="Invoke">
216 <INPUT TYPE="RESET" VALUE="Clear">
217 </FORM>
218 <%
219 break;
220 case 74:
221 valid = false;
222 %>
223 <FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
224 <INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
225 <TABLE>
226 <TR>
227 <TD COLSPAN="1" ALIGN="LEFT">id:</TD>
228 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="id89" SIZE=20></TD>
229 </TR>
230 </TABLE>
231
232 <INPUT TYPE="SUBMIT" VALUE="Invoke">
233 <INPUT TYPE="RESET" VALUE="Clear">
234 </FORM>
235 <%
236 break;
237 case 1111111111:
238 valid = false;
239 %>
240 <FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
241 <INPUT TYPE="HIDDEN" NAME="method" VALUE="<%=org.eclipse.jst.ws.util.JspUtils.markup(method)%>">
242 <TABLE>
243 <TR>
244 <TD COLSPAN="1" ALIGN="LEFT">URLString:</TD>
245 <TD ALIGN="left"><INPUT TYPE="TEXT" NAME="url1111111111" SIZE=20></TD>
246 </TR>
247 </TABLE>
248
249 <INPUT TYPE="SUBMIT" VALUE="Invoke">
250 <INPUT TYPE="RESET" VALUE="Clear">
251 </FORM>
252 <%
253 break;
254 case 1111111112:
255 valid = false;
256 %>
257 <FORM METHOD="POST" ACTION="Result.jsp" TARGET="result">
HTML/BODY/#text
Writable
Smart Insert
200 : 8 : 5247
15:32
10/01/2020

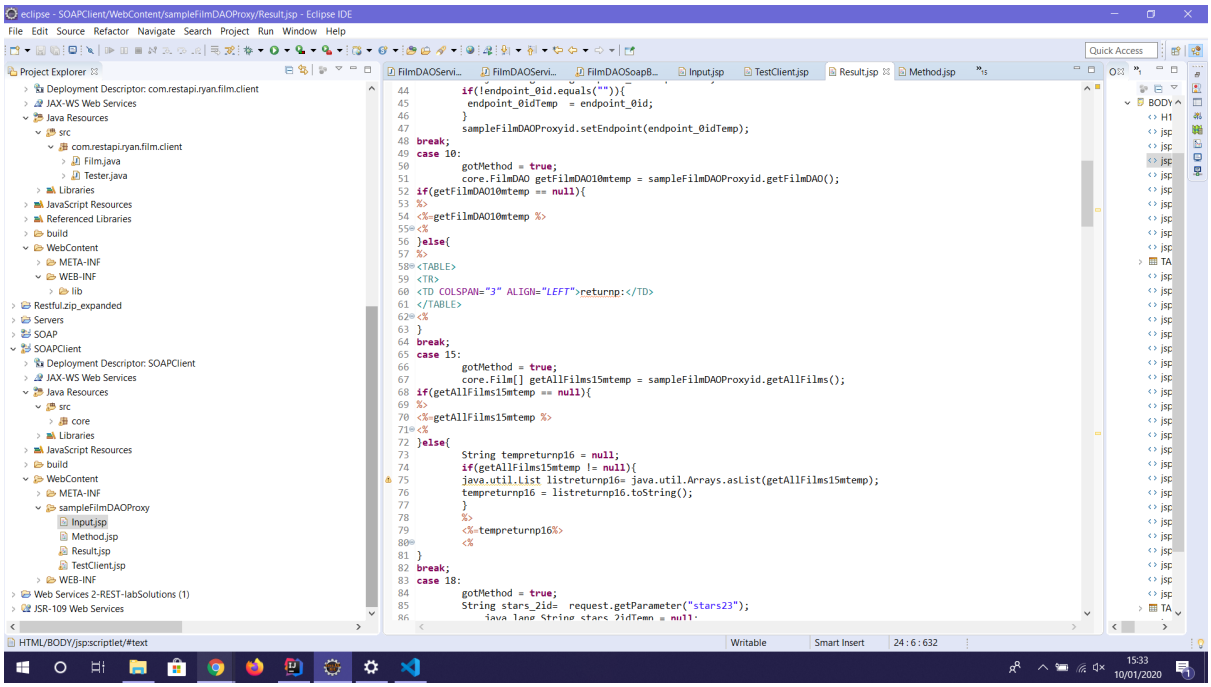


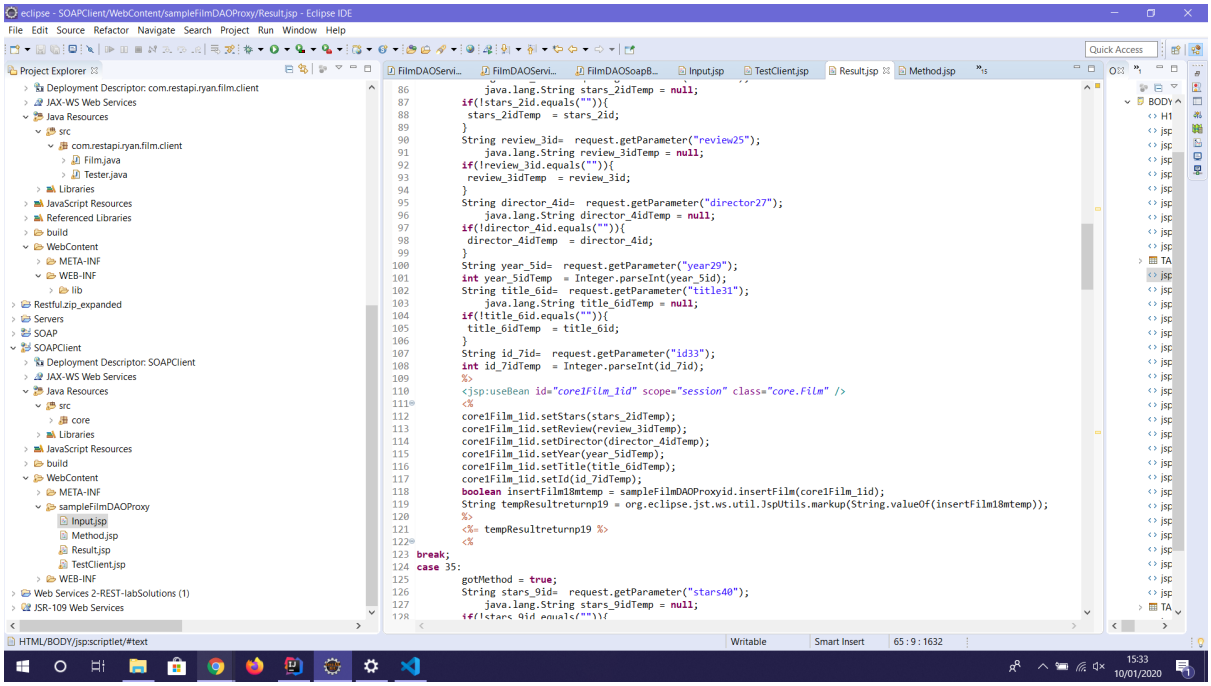
Method

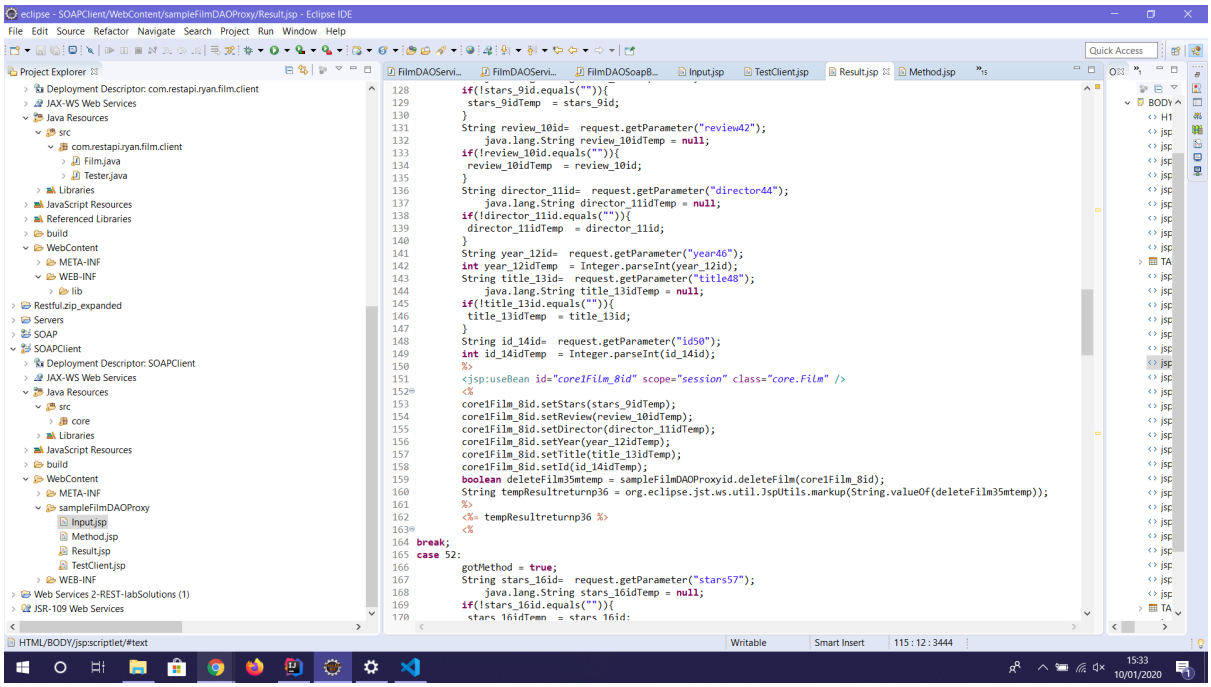


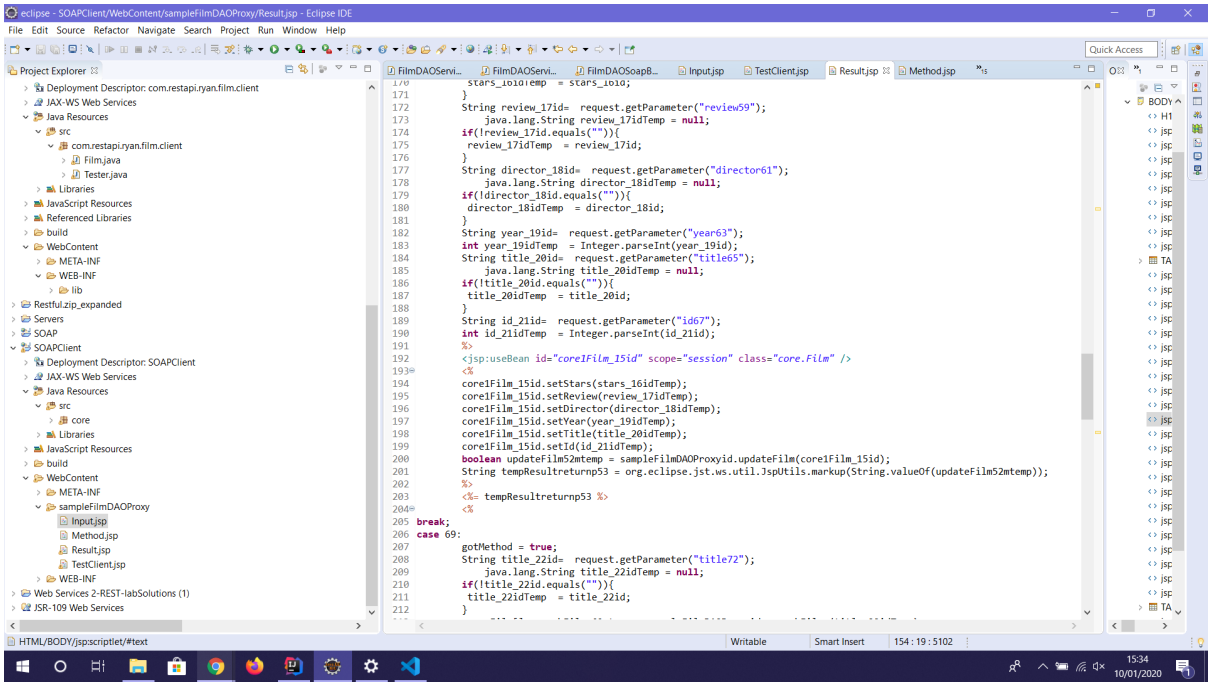
Result

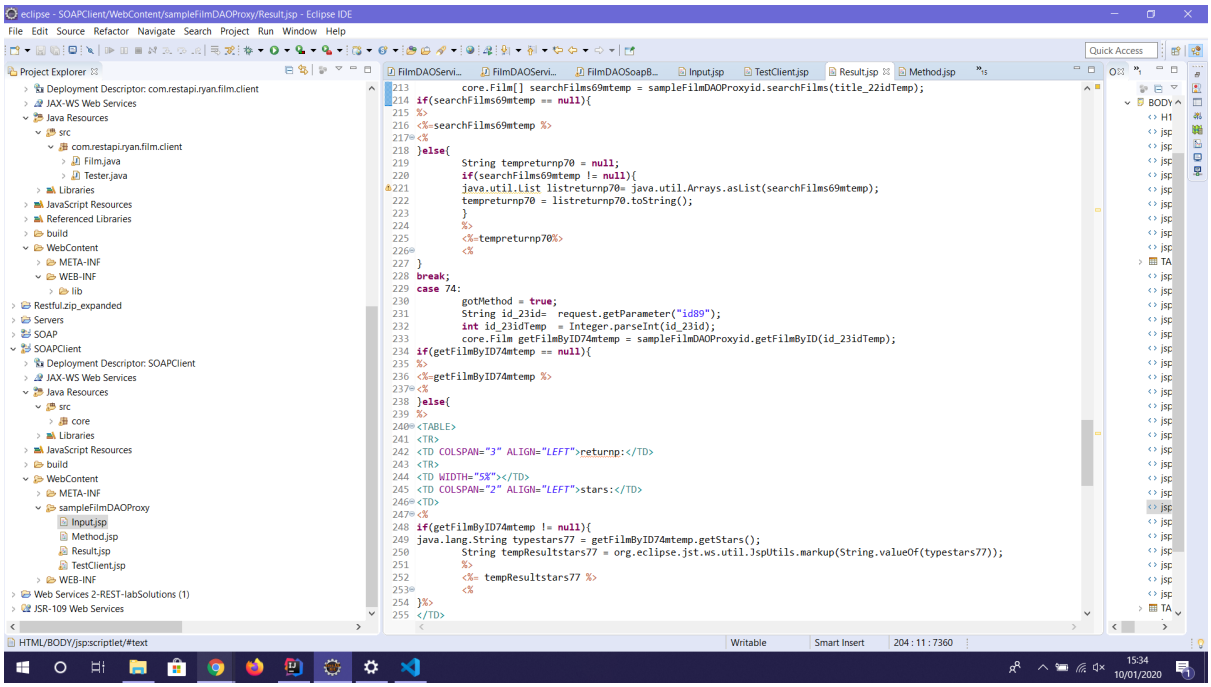

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
jsp:directive
jsp:scriptlet
HTML
<%@page contentType="text/html;charset=UTF-8"%>
<% request.setCharacterEncoding("UTF-8"); %>
<HTML>
<HEAD>
<TITLE>Result</TITLE>
</HEAD>
<BODY>
<H1>Result</H1>
<jsp:useBean id="sampleFilmDAOProxyid" scope="session" class="core.FilmDAOProxy" />
<%
if (request.getParameter("endpoint") != null && request.getParameter("endpoint").length() > 0)
sampleFilmDAOProxyid.setEndpoint(request.getParameter("endpoint"));
%>
<%
String method = request.getParameter("method");
int methodID = 0;
if (method == null) methodID = -1;
if(methodID != -1) methodID = Integer.parseInt(method);
boolean gotMethod = false;
try {
switch (methodID){
case 2:
gotMethod = true;
java.lang.String getEndpoint2mtemp = sampleFilmDAOProxyid.getEndpoint();
if(getEndpoint2mtemp == null){
%>
<%=getEndpoint2mtemp %>
<%
}else{
String tempResultreturnp3 = org.eclipse.jst.ws.util.JspUtils.markup(String.valueOf(getEndpoint2mtemp));
%>
<%= tempResultreturnp3 %>
<%
}
break;
case 5:
gotMethod = true;
String endpoint_0id= request.getParameter("endpoint8");
java.lang.String endpoint_0idTemp = null;
Writable
Smart Insert
1 : 1 : 0
15:33
10/01/2020

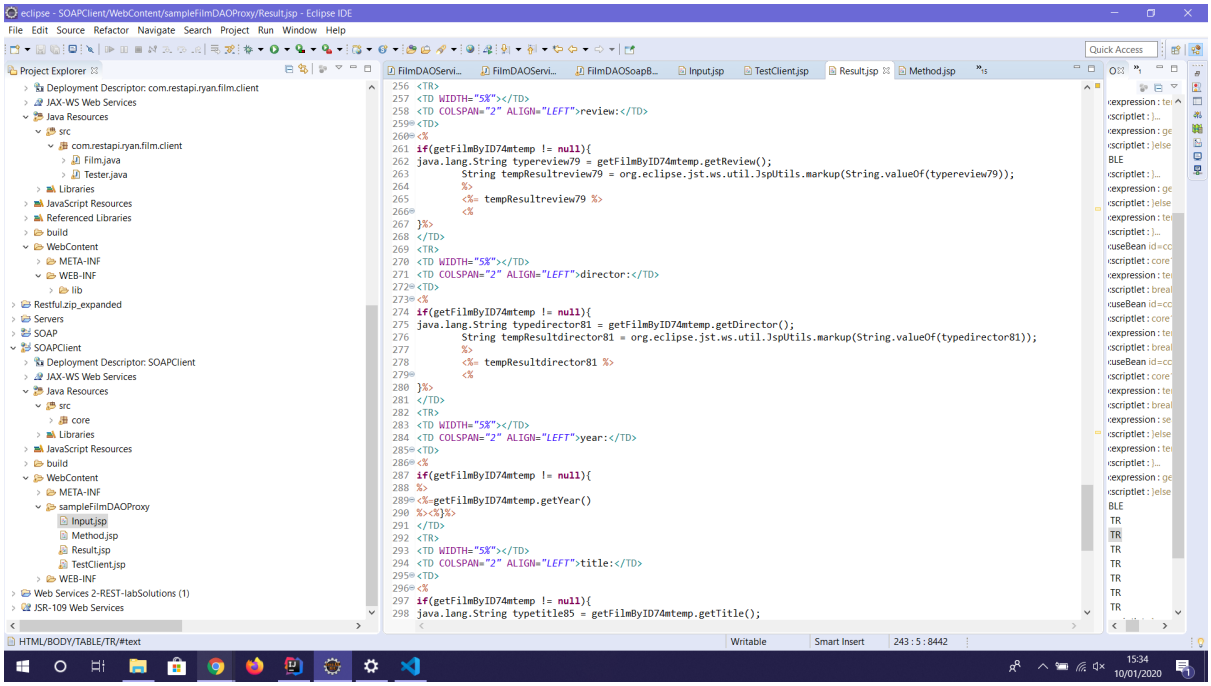

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
BODY
H1
TA
if(!endpoint_0id.equals("")){
endpoint_0idTemp = endpoint_0id;
}
sampleFilmDAOProxyid.setEndpoint(endpoint_0idTemp);
break;
case 10:
gotMethod = true;
core.FilmDAO getFilmDAO10mtemp = sampleFilmDAOProxyid.getFilmDAO();
if(getFilmDAO10mtemp == null){
%>
<%=getFilmDAO10mtemp %>
<%
}else{
%>
<TABLE>
<TR>
<TD COLSPAN="3" ALIGN="LEFT">returnp:</TD>
</TABLE>
<%
}
break;
case 15:
gotMethod = true;
core.Film[] getAllFilms15mtemp = sampleFilmDAOProxyid.getAllFilms();
if(getAllFilms15mtemp == null){
%>
<%=getAllFilms15mtemp %>
<%
}else{
String tempreturnp16 = null;
if(getAllFilms15mtemp != null){
java.util.List listreturnp16= java.util.Arrays.asList(getAllFilms15mtemp);
tempreturnp16 = listreturnp16.toString();
}
%>
<%=tempreturnp16%>
<%
}
break;
case 18:
gotMethod = true;
String stars_2id= request.getParameter("stars23");
HTML/BODY/jsp:scriptlet/#text
Writable
Smart Insert
24 : 6 : 632
15:33
10/01/2020

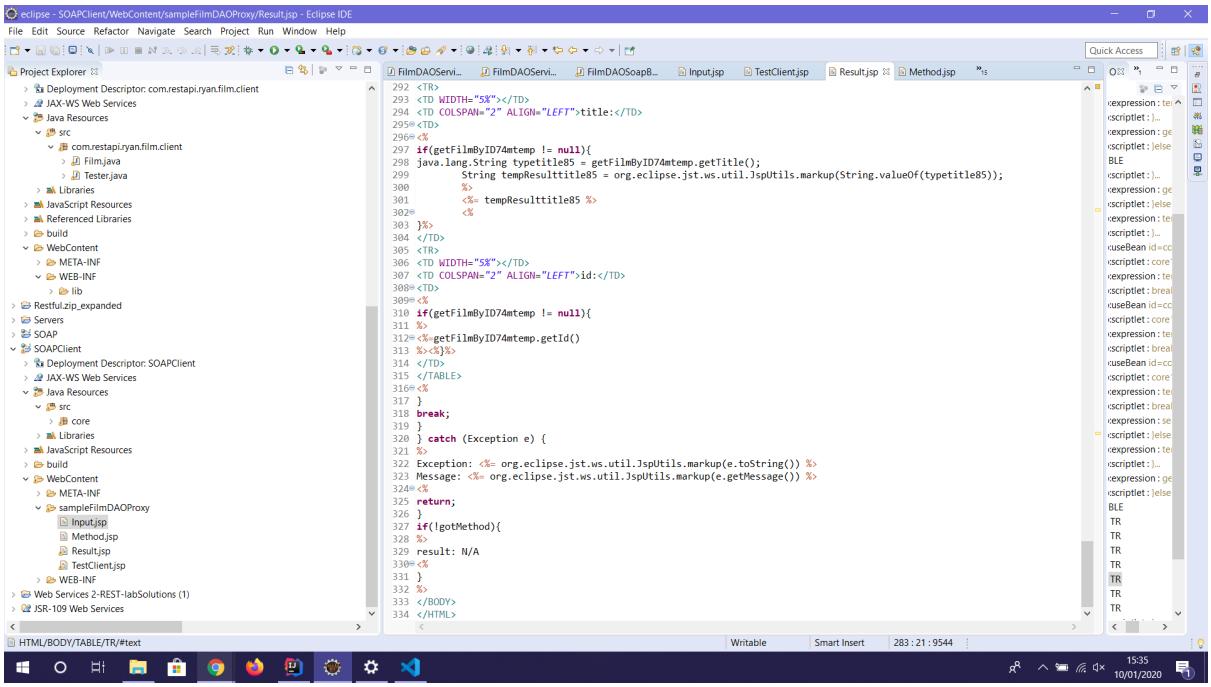

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
HTML/BODY/jsp:scriptlet/#text
Writable
Smart Insert
65 : 9 : 1632
15:33
10/01/2020


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
HTML/BODY/jsp:scriptlet/#text
Writable
Smart Insert
115 : 12 : 3444
15:33
10/01/2020

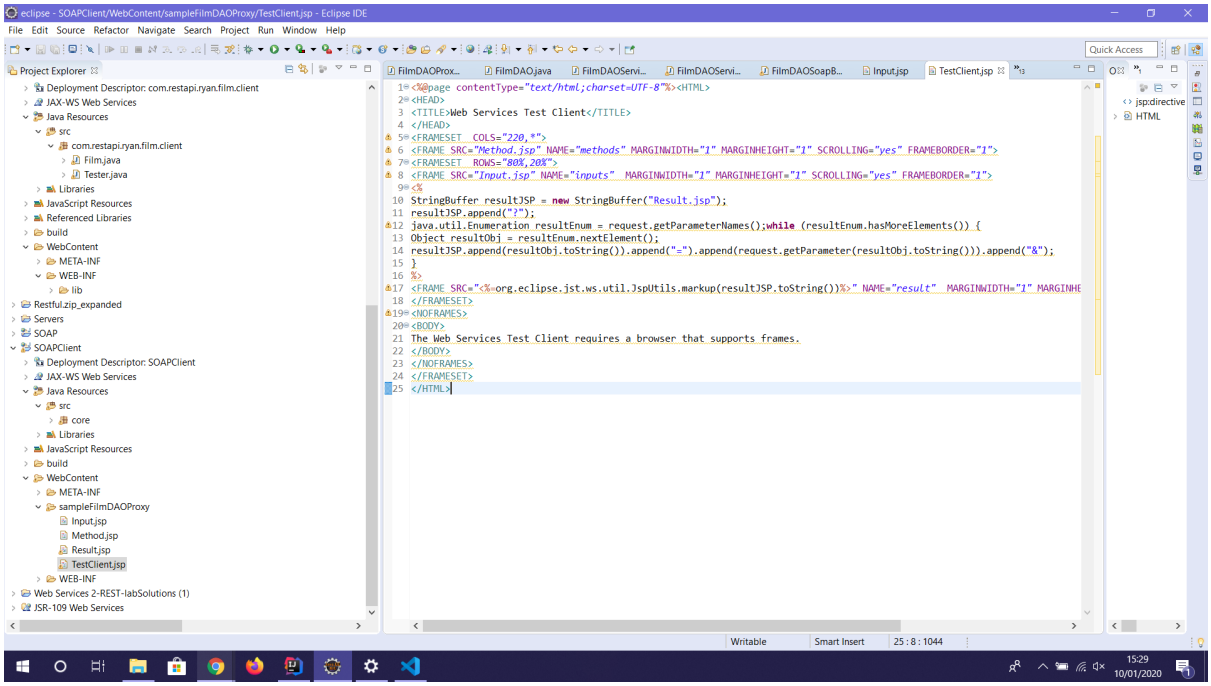

eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
HTML/BODY/jsp:scriptlet/#text
Writable
Smart Insert
154 : 19 : 5102
15:34
10/01/2020


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
HTML/BODY/jsp:scriptlet/#text
Writable
Smart Insert
204 : 11 : 7360
15:34
10/01/2020


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
HTML/BODY/TABLE/TR/#text
Writable
Smart Insert
243 : 5 : 8442
15:34
10/01/2020


eclipse - SOAPClient/WebContent/sampleFilmDAOProxy/Result.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Services 2-REST-labSolutions (1)
JSR-109 Web Services
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
Quick Access
HTML/BODY/TABLE/TR/#text
Writable
Smart Insert
283 : 21 : 9544
15:35
10/01/2020

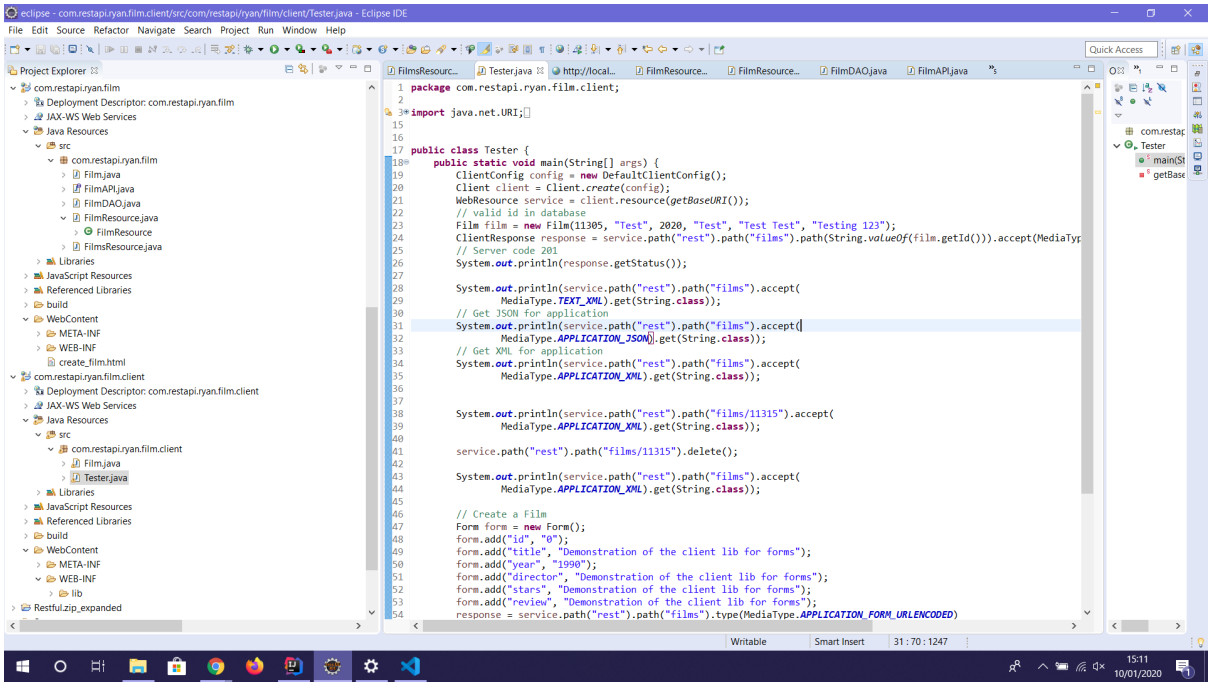
TestClient

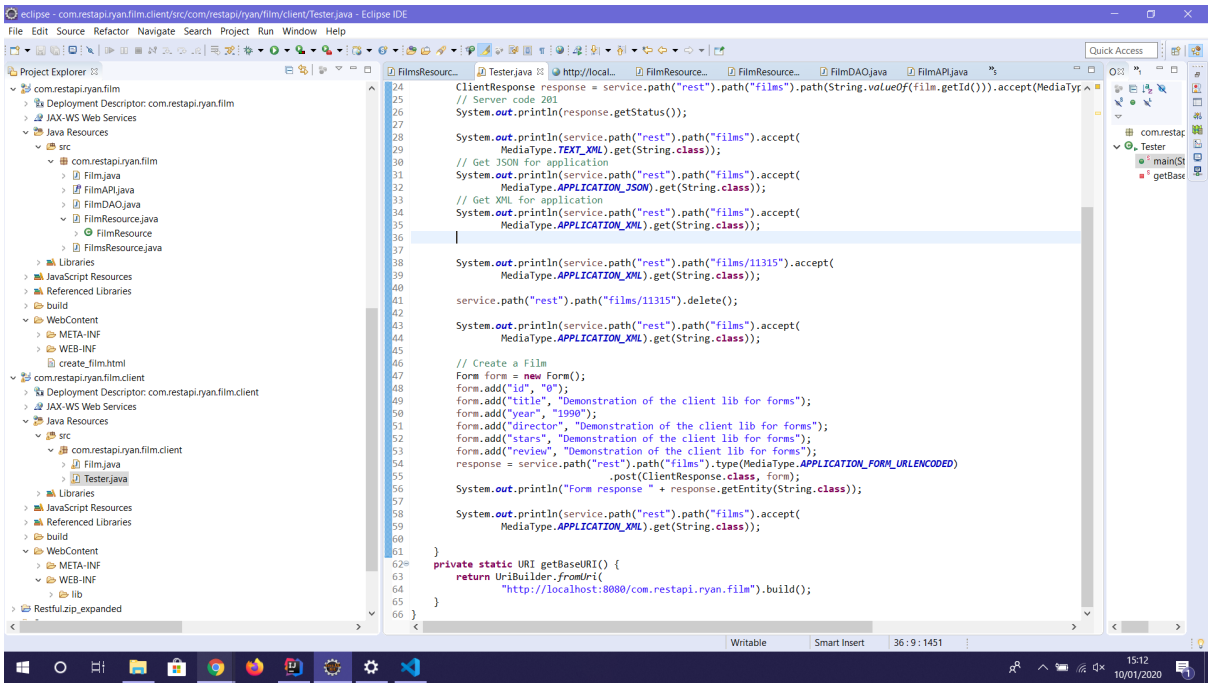


Rest

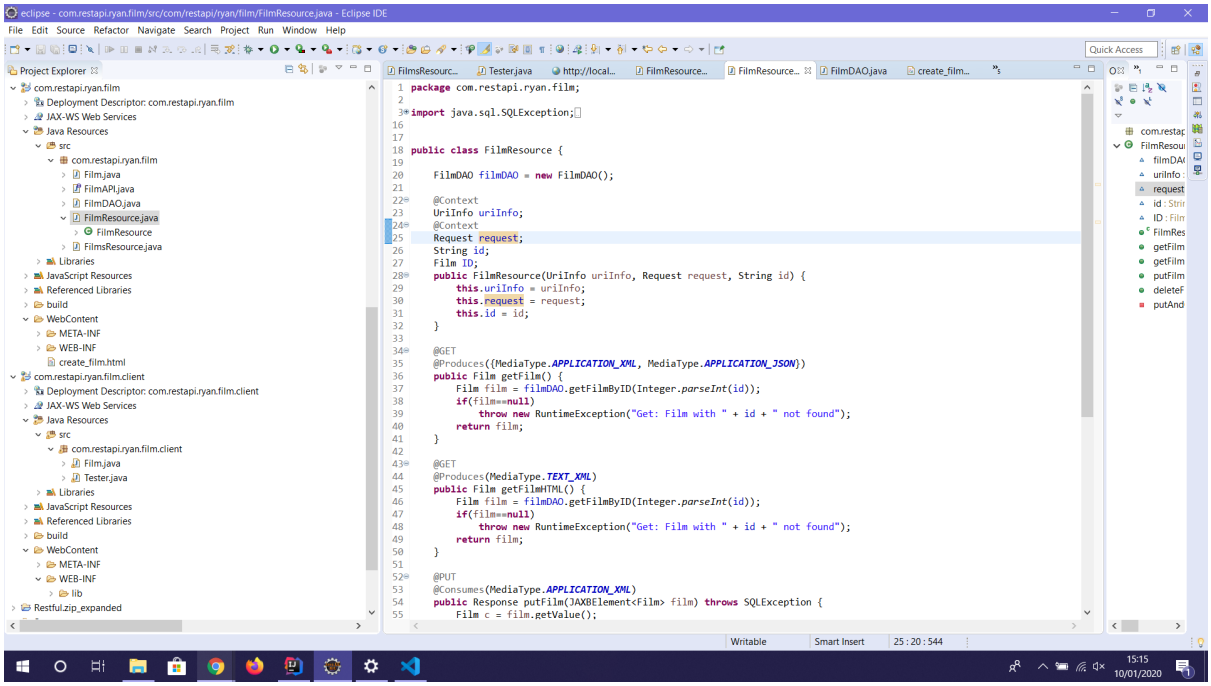
In the rest application that i have created it is able to communicate on the REST API layer being able to the put get delete and post data from my already exciting application the only difference is that rather than using a connection pool it is using a static connect method inside the main film dao application and is called upon each time a request to called this applies to opening the connection and the closing of the connections this may seem like a odd approach but based on the complexity and general waste of implementation for the application it is only fitting that the application uses elements like in the soap to give it a more like for like comparison in its construction for the application so there are only three new files in this application one is a tester file which will run some java code to test the application endpoints then other two are the actual REST API implementation so one of the files is a film resource this will handle getting the relevant information like the URI and build information for the rest acting as a constructor for the rest of the files to work around it and provide the service while the other file is films resource which will handle the actual request and deal with the implementation from the filmdao to be used in the application to ensure that the application is picked up at each endpoint with crud operations in full effect that correspond to the appropriate action in rest then another file which is just a html file for creating new films in the database so that the tester file can be used to get the application working the rest aspect is in a separate project and the client side ie the tester is in another to ensure aspect are modular and separate to give the back end the ability to work in other project without requiring any alterations.

Tester

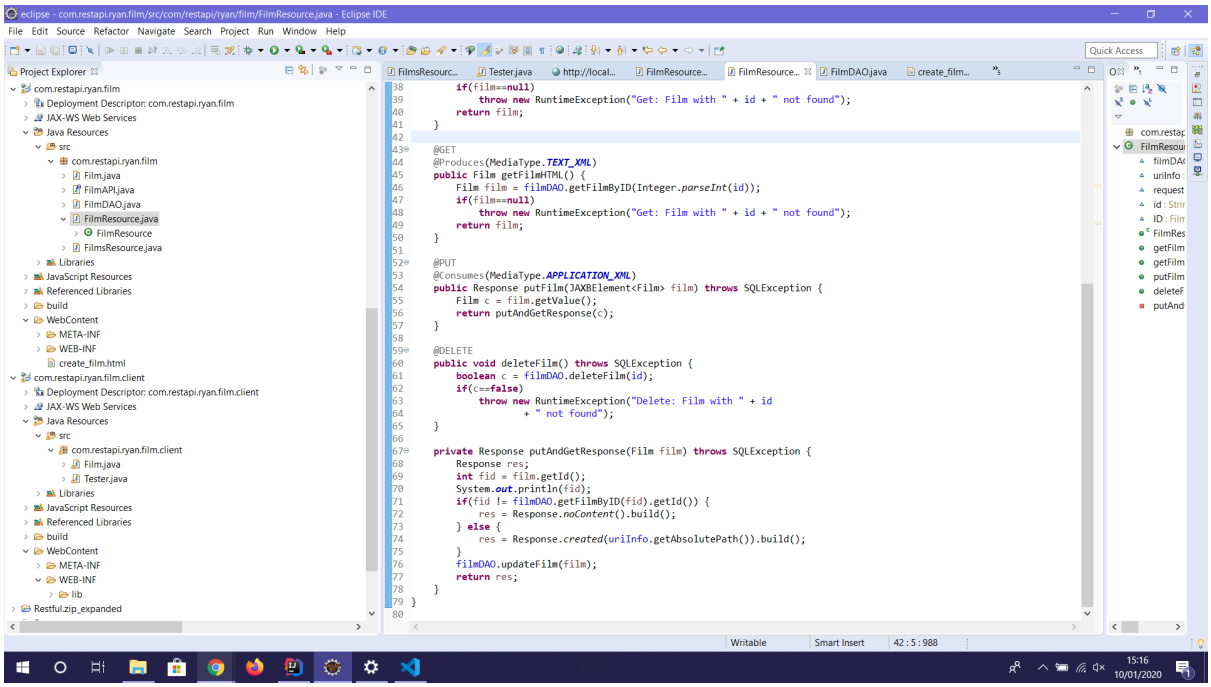




FilmResource

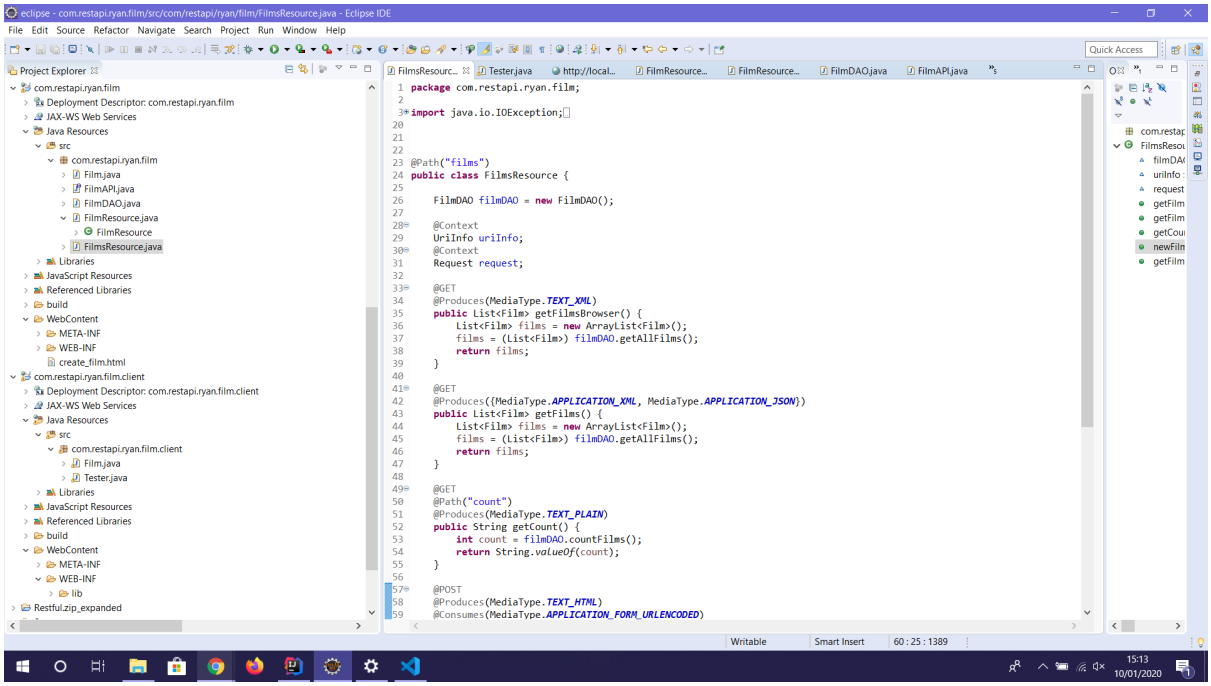


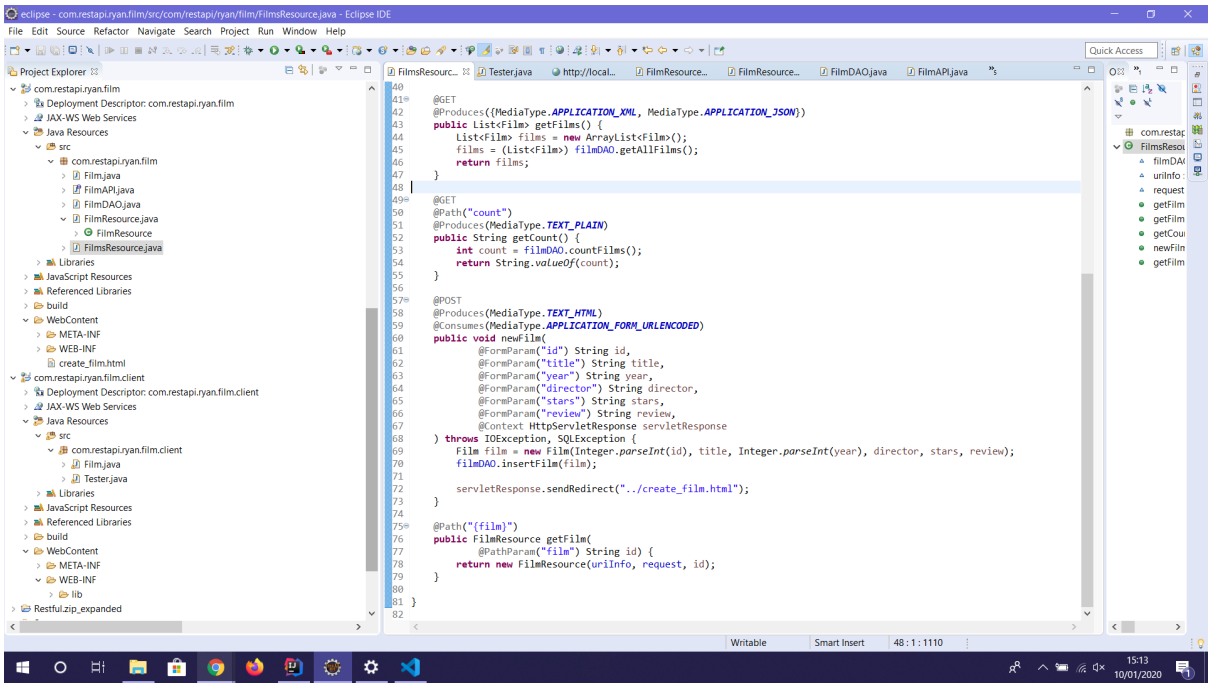
```
package com.restapi.ryan.film;

import java.sql.SQLException;

public class FilmResource {

    FilmDAO filmDAO = new FilmDAO();

    @Context
    UriInfo uriInfo;
    @Context
    Request request;
    String id;
    Film ID;
    public FilmResource(UriInfo uriInfo, Request request, String id) {
        this.uriInfo = uriInfo;
        this.request = request;
        this.id = id;
    }

    @GET
    @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
    public Film getFilm() {
        Film film = filmDAO.getFilmByID(Integer.parseInt(id));
        if(film==null)
            throw new RuntimeException("Get: Film with " + id + " not found");
        return film;
    }

    @GET
    @Produces(MediaType.TEXT_XML)
    public Film getFilmHTML() {
        Film film = filmDAO.getFilmByID(Integer.parseInt(id));
        if(film==null)
            throw new RuntimeException("Get: Film with " + id + " not found");
        return film;
    }

    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public Response putFilm(JAXBElement<Film> film) throws SQLException {
        Film c = film.getValue();
```



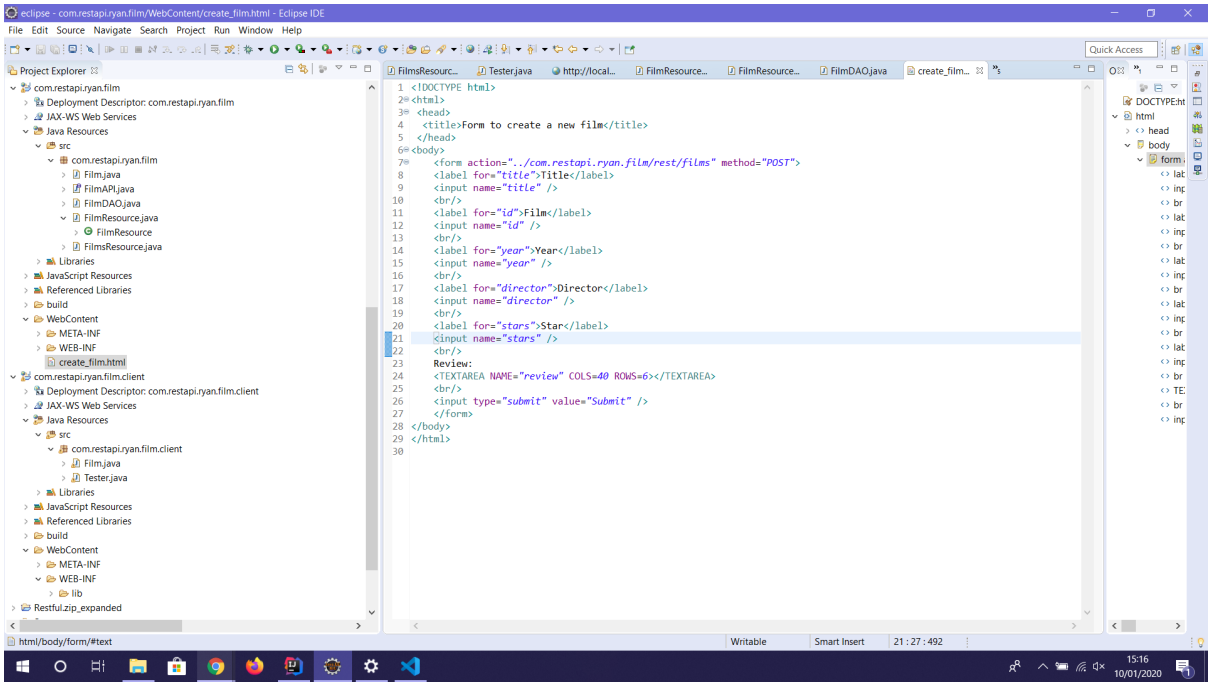
```
        if(film==null)
            throw new RuntimeException("Get: Film with " + id + " not found");
        return film;
    }

    @GET
    @Produces(MediaType.TEXT_XML)
    public Film getFilmHTML() {
        Film film = filmDAO.getFilmByID(Integer.parseInt(id));
        if(film==null)
            throw new RuntimeException("Get: Film with " + id + " not found");
        return film;
    }

    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public Response putFilm(JAXBElement<Film> film) throws SQLException {
        Film c = film.getValue();
        return putAndGetResponse(c);
    }

    @DELETE
    public void deleteFilm() throws SQLException {
        boolean c = filmDAO.deleteFilm(id);
        if(c==false)
            throw new RuntimeException("Delete: Film with " + id
                    + " not found");
    }

    private Response putAndGetResponse(Film film) throws SQLException {
        Response res;
        int fid = film.getId();
        System.out.println(fid);
        if(fid != filmDAO.getFilmByID(fid).getId()) {
            res = Response.noContent().build();
        } else {
            res = Response.created(uriInfo.getAbsolutePath()).build();
        }
        filmDAO.updateFilm(film);
        return res;
    }
}
```

FilmsResource


eclipse - com.restapi.ryan.film/src/com/restapi/ryan/film/FilmsResource.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
com.restapi.ryan.film
Deployment Descriptor: com.restapi.ryan.film
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film
Film.java
FilmAPI.java
FilmDAO.java
FilmResource.java
FilmResource
FilmsResource.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
create_film.html
com.restapi.ryan.film.client
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
FilmsResourc...
Tester.java
http://local...
FilmResource...
FilmResource...
FilmDAO.java
FilmAPI.java
Quick Access
1 package com.restapi.ryan.film;
2
3* import java.io.IOException;
20
21
22
23 @Path("films")
24 public class FilmsResource {
25
26 FilmDAO filmDAO = new FilmDAO();
27
28 @Context
29 UriInfo uriInfo;
30 @Context
31 Request request;
32
33 @GET
34 @Produces(MediaType.TEXT_XML)
35 public List<Film> getFilmsBrowser() {
36 List<Film> films = new ArrayList<Film>();
37 films = (List<Film>) filmDAO.getAllFilms();
38 return films;
39 }
40
41 @GET
42 @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
43 public List<Film> getFilms() {
44 List<Film> films = new ArrayList<Film>();
45 films = (List<Film>) filmDAO.getAllFilms();
46 return films;
47 }
48
49 @GET
50 @Path("count")
51 @Produces(MediaType.TEXT_PLAIN)
52 public String getCount() {
53 int count = filmDAO.countFilms();
54 return String.valueOf(count);
55 }
56
57 @POST
58 @Produces(MediaType.TEXT_HTML)
59 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
Writable
Smart Insert
60 : 25 : 1389
15:13
10/01/2020


eclipse - com.restapi.ryan.film/src/com/restapi/ryan/film/FilmsResource.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
com.restapi.ryan.film
Deployment Descriptor: com.restapi.ryan.film
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film
Film.java
FilmAPI.java
FilmDAO.java
FilmResource.java
FilmResource
FilmsResource.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
create_film.html
com.restapi.ryan.film.client
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
FilmsResourc...
Tester.java
http://local...
FilmResource...
FilmResource...
FilmDAO.java
FilmAPI.java
Quick Access
41 @GET
42 @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
43 public List<Film> getFilms() {
44 List<Film> films = new ArrayList<Film>();
45 films = (List<Film>) filmDAO.getAllFilms();
46 return films;
47 }
48
49 @GET
50 @Path("count")
51 @Produces(MediaType.TEXT_PLAIN)
52 public String getCount() {
53 int count = filmDAO.countFilms();
54 return String.valueOf(count);
55 }
56
57 @POST
58 @Produces(MediaType.TEXT_HTML)
59 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
60 public void newFilm(
61 @FormParam("id") String id,
62 @FormParam("title") String title,
63 @FormParam("year") String year,
64 @FormParam("director") String director,
65 @FormParam("stars") String stars,
66 @FormParam("review") String review,
67 @Context HttpServletResponse servletResponse
68) throws IOException, SQLException {
69 Film film = new Film(Integer.parseInt(id), title, Integer.parseInt(year), director, stars, review);
70 filmDAO.insertFilm(film);
71
72 servletResponse.sendRedirect("../create_film.html");
73 }
74
75 @Path("{film}")
76 public FilmResource getFilm(
77 @PathParam("film") String id) {
78 return new FilmResource(uriInfo, request, id);
79 }
80
81 }
82
Writable
Smart Insert
48 : 1 : 1110
15:13
10/01/2020

Create Film



Comparison

when looking at all three implementation for the applications it is quite easy to see which one is more featured rich being the HTTP project as this has elements like cloud functionality as well as an interface that has been made by the developer for the action required whereas something like rest and soap will use generated or console applications to fulfill this role as HTTP also has the leg up in newer technologies with JavaScript es6 and connection pooling for caching the queries which lead to an overall better experience in my opinion

but soap has its benefits as well compared to the others that being the generation aspect meaning an application can be up and running as long as you have your methods and data constructors making it a less painful ordeal unlike the other two methods which required hours of tinkering to get it fully operation where as soap did this is a couple of buttons with things like the back end and the front end made in a way that can show off each function in a web browser for that endpoint

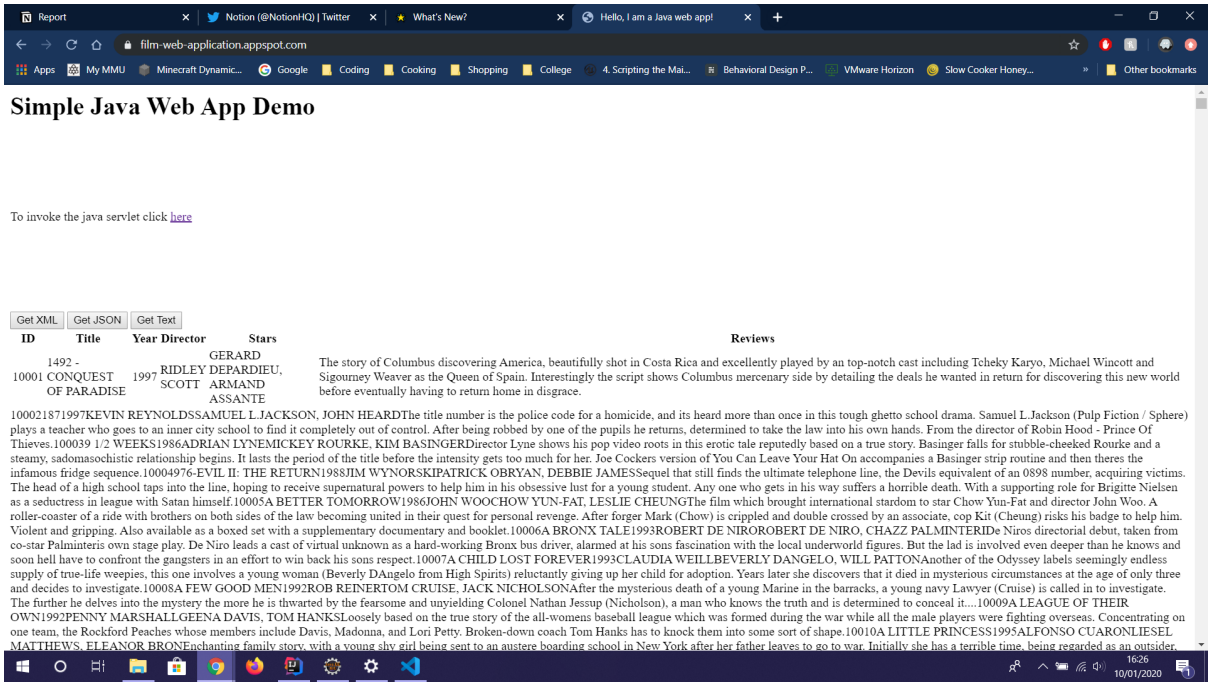
but rest also had a leg up on the other projects in that it can be easily adaptable to any project due to it providing an endpoint in which you provide a client to and with the separation of the view and back end this makes it much more will integrate with other application whereas soap would require tinkering with the generation which could cause more issues and the HTTP required more code to get the data from the back end to the front end with the servlets and jsp

Conclusion

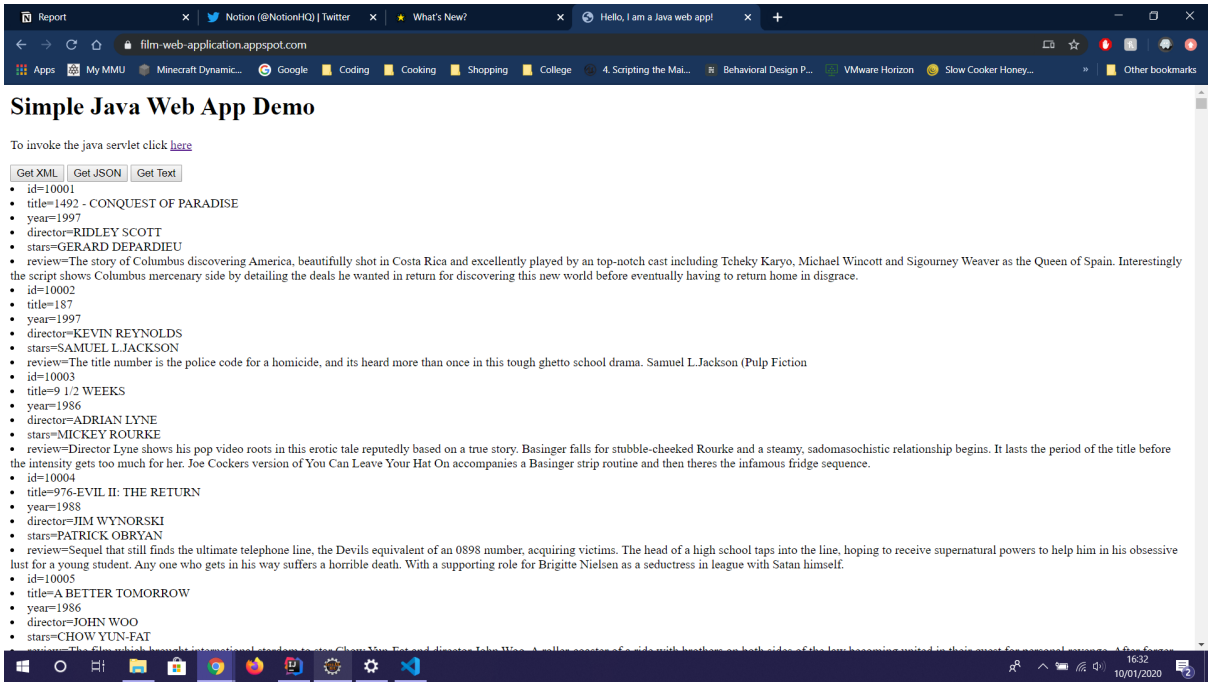
in terms of which is the better application there is an argument for all three but in my opinion the HTTP is the better choice as granted it required more work but the end result turned out a professional product that is being delivered on the internet with the ability to add more on to future proof the application like login system or animations as points like modular from the rest side can be accomplished with HTTP as the MVC model allows for this process and the generation aspect can also be created for HTTP using crud generators so although all three deliver a purpose HTTP is the better option for the application implementation for this project

Results

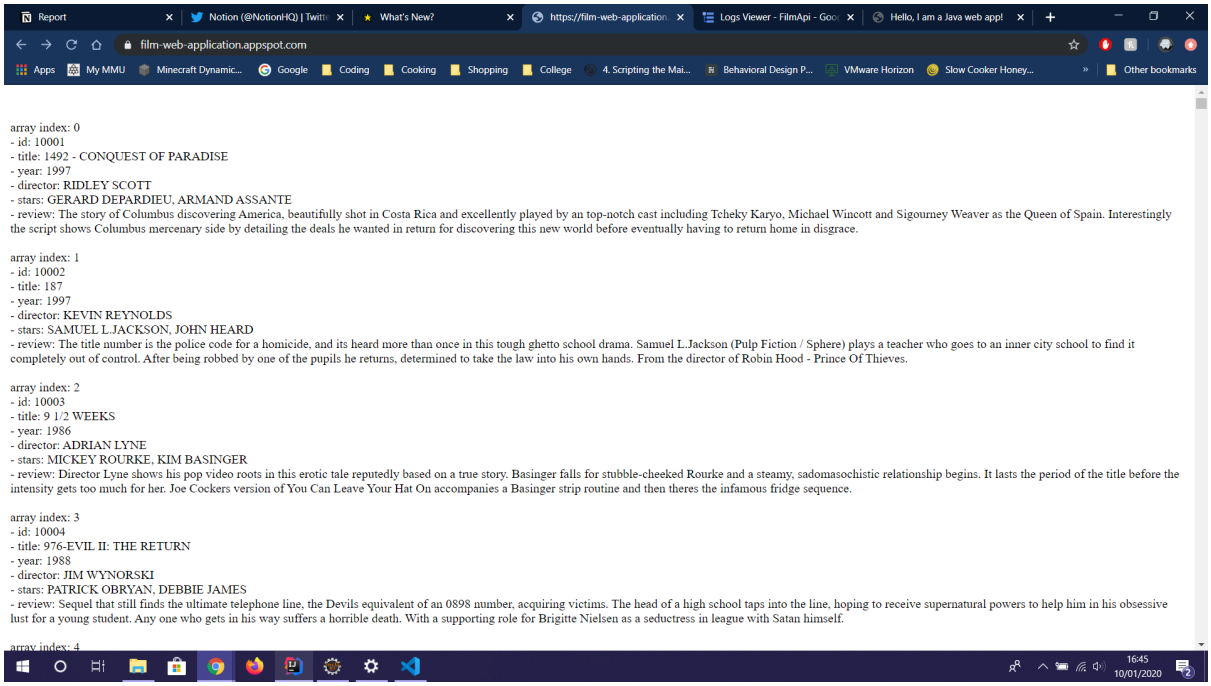
Ajax XML



Ajax Text



Ajax JSON

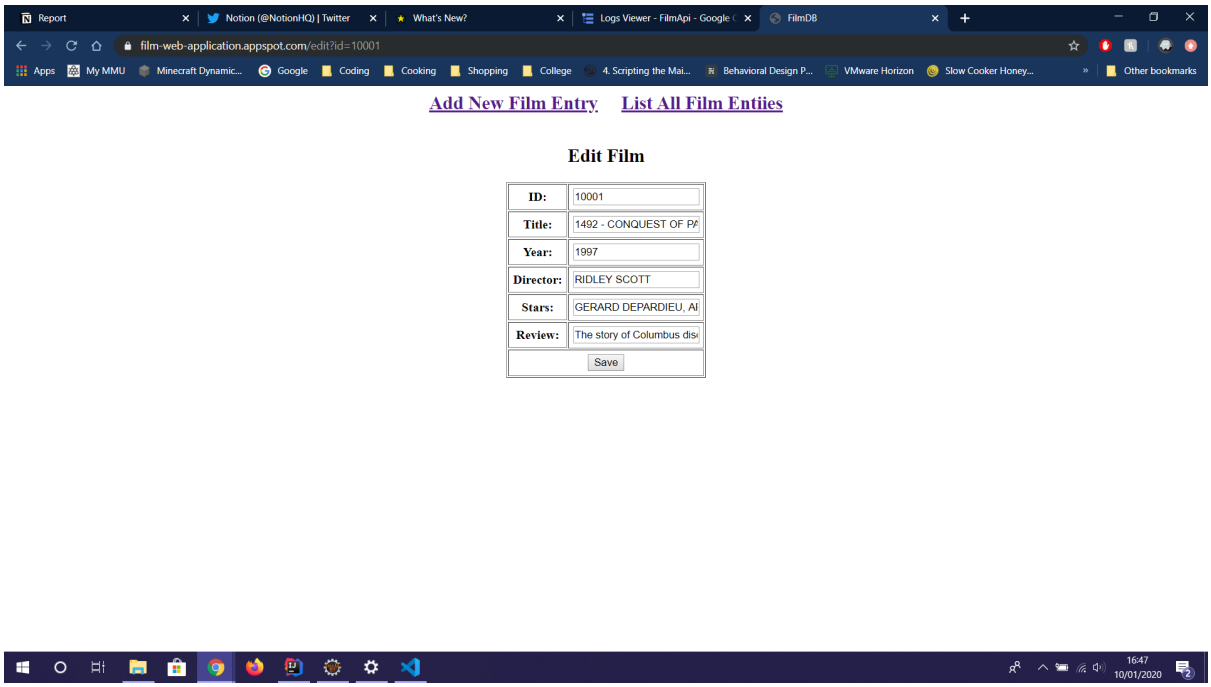


GUI

Read

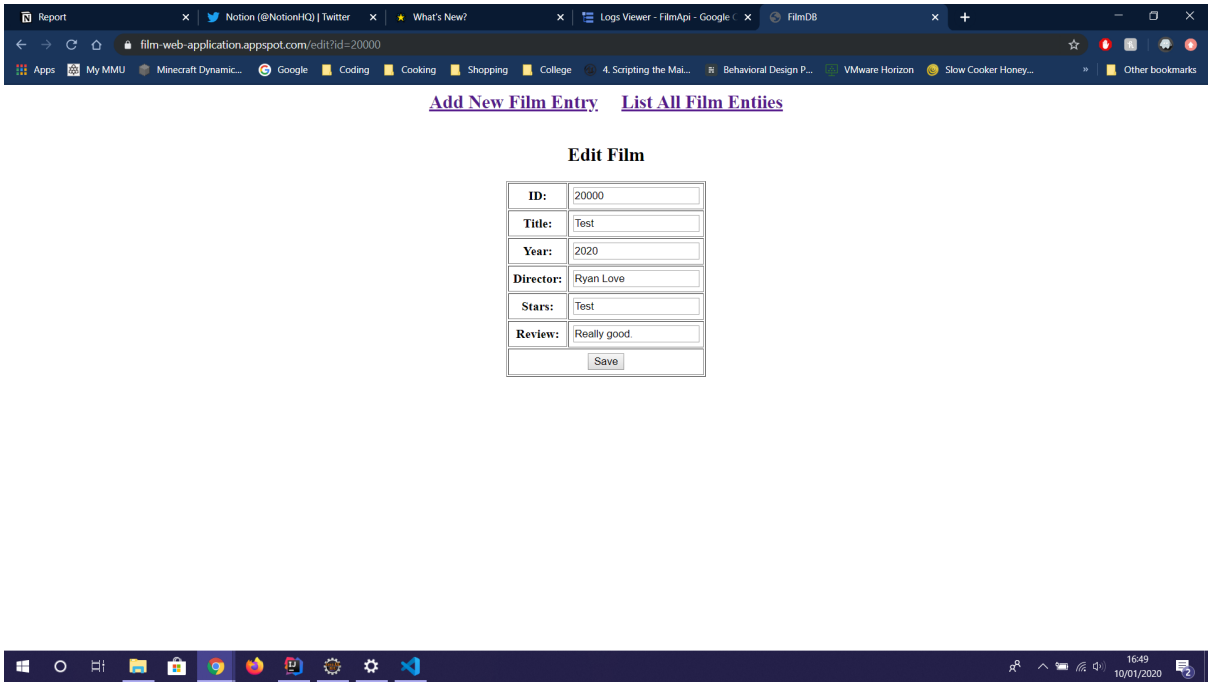
Update

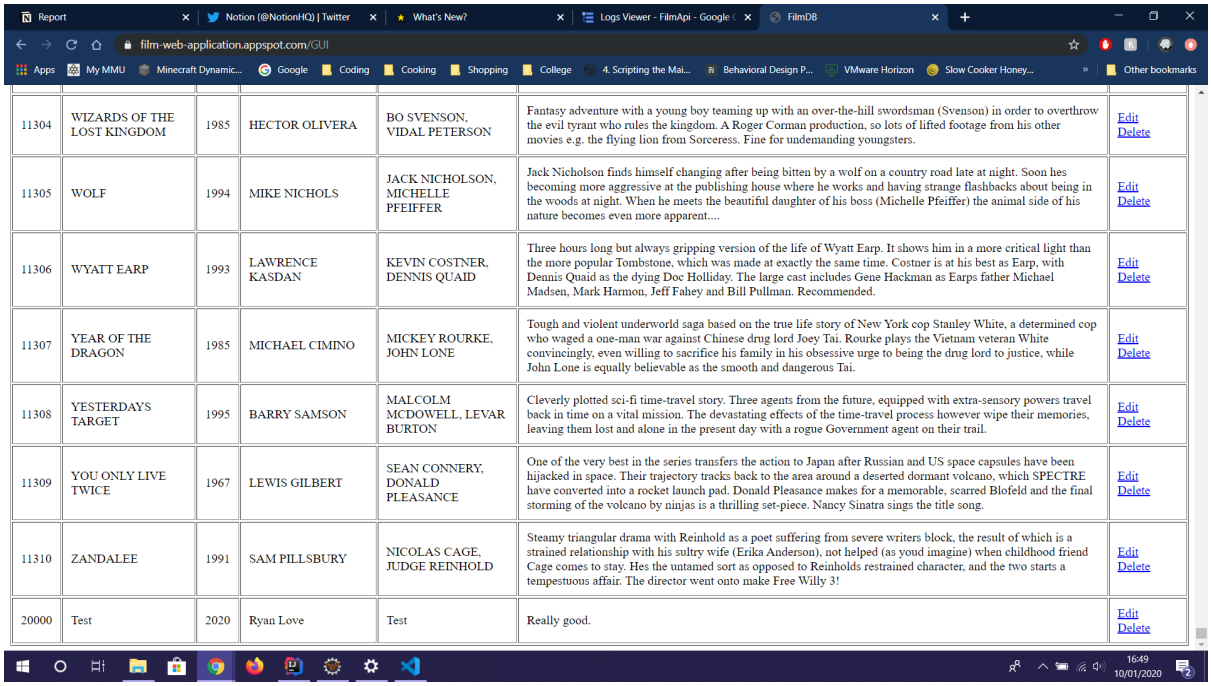

Report
Notion (@NotionHQ) | Twitter
What's New?
Logs Viewer - FilmApi - Google
FilmDB
film-web-application.appspot.com/GUI
Apps
My MMU
Minecraft Dynamic...
Google
Coding
Cooking
Shopping
College
4. Scripting the Mai...
Behavioral Design P...
VMware Horizon
Slow Cooker Honey...
Other bookmarks
Film
Add New Film Entry
List All Film Entiies
List of All Films
ID
Title
Year
Director
Stars
Review
Operations
10001
1492 - CONQUEST OF PARADISE
1997
RIDLEY SCOTT
GERARD DEPARDIEU, ARMAND ASSANTE
The story of Columbus discovering America, beautifully shot in Costa Rica and excellently played by an top-notch cast including Tcheky Karyo, Michael Wincott and Sigourney Weaver as the Queen of Spain. Interestingly the script shows Columbus mercenary side by detailing the deals he wanted in return for discovering this new world before eventually having to return home in disgrace.
Edit
Delete
10002
187
1997
KEVIN REYNOLDS
SAMUEL L.JACKSON, JOHN HEARD
The title number is the police code for a homicide, and its heard more than once in this tough ghetto school drama. Samuel L.Jackson (Pulp Fiction / Sphere) plays a teacher who goes to an inner city school to find it completely out of control. After being robbed by one of the pupils he returns, determined to take the law into his own hands. From the director of Robin Hood - Prince Of Thieves.
Edit
Delete
10003
9 1/2 WEEKS
1986
ADRIAN LYNE
MICKEY ROURKE, KIM BASINGER
Director Lyne shows his pop video roots in this erotic tale reputedly based on a true story. Basinger falls for stubble-cheeked Rourke and a steamy, sadomasochistic relationship begins. It lasts the period of the title before the intensity gets too much for her. Joe Cockers version of You Can Leave Your Hat On accompanies a Basinger strip routine and then theres the infamous fridge sequence.
Edit
Delete
10004
976-EVIL II: THE RETURN
1988
JIM WYNORSKI
PATRICK OBRYAN, DEBBIE JAMES
Sequel that still finds the ultimate telephone line, the Devils equivalent of an 0898 number, acquiring victims. The head of a high school taps into the line, hoping to receive supernatural powers to help him in his obsessive lust for a young student. Any one who gets in his way suffers a horrible death. With a supporting role for Brigitte Nielsen as a seductress in league with Satan himself.
Edit
Delete
10005
A BETTER TOMORROW
1986
JOHN WOO
CHOW YUN-FAT, LESLIE CHEUNG
The film which brought international stardom to star Chow Yun-Fat and director John Woo. A roller-coaster of a ride with brothers on both sides of the law becoming united in their quest for personal revenge. After forger Mark (Chow) is crippled and double crossed by an associate, cop Kit (Cheung) risks his badge to help him. Violent and gripping. Also available as a boxed set with a supplementary documentary and booklet.
Edit
Delete
De Niros directorial debut, taken from co-star Palminteris own stage play. De Niro leads a cast of virtual
16:47
10/01/2020


Report
Notion (@NotionHQ) | Twitter
What's New?
Logs Viewer - FilmApi - Google
FilmDB
film-web-application.appspot.com/edit?id=10001
Apps
My MMU
Minecraft Dynamic...
Google
Coding
Cooking
Shopping
College
4. Scripting the Mai...
Behavioral Design P...
VMware Horizon
Slow Cooker Honey...
Other bookmarks
Add New Film Entry
List All Film Entiies
Edit Film
ID:
10001
Title:
1492 - CONQUEST OF PA
Year:
1997
Director:
RIDLEY SCOTT
Stars:
GERARD DEPARDIEU, AI
Review:
The story of Columbus dis
Save
16:47
10/01/2020


Report
Notion (@NotionHQ) | Twitter
What's New?
Logs Viewer - FilmApi - Google
FilmDB
film-web-application.appspot.com/GUI
Apps
My MMU
Minecraft Dynamic...
Google
Coding
Cooking
Shopping
College
4. Scripting the Mai...
Behavioral Design P...
VMware Horizon
Slow Cooker Honey...
Other bookmarks
Film
Add New Film Entry
List All Film Entiies
List of All Films
ID
Title
Year
Director
Stars
Review
Operations
10001
1492 - CONQUEST OF PARADISE
2020
RIDLEY SCOTT
GERARD DEPARDIEU, ARMAND ASSANTE
The story of Columbus discovering America, beautifully shot in Costa Rica and excellently played by an top-notch cast including Tcheky Karyo, Michael Wincott and Sigourney Weaver as the Queen of Spain. Interestingly the script shows Columbus mercenary side by detailing the deals he wanted in return for discovering this new world before eventually having to return home in disgrace.
Edit
Delete
10002
187
1997
KEVIN REYNOLDS
SAMUEL L.JACKSON, JOHN HEARD
The title number is the police code for a homicide, and its heard more than once in this tough ghetto school drama. Samuel L.Jackson (Pulp Fiction / Sphere) plays a teacher who goes to an inner city school to find it completely out of control. After being robbed by one of the pupils he returns, determined to take the law into his own hands. From the director of Robin Hood - Prince Of Thieves.
Edit
Delete
10003
9 1/2 WEEKS
1986
ADRIAN LYNE
MICKEY ROURKE, KIM BASINGER
Director Lyne shows his pop video roots in this erotic tale reputedly based on a true story. Basinger falls for stubble-cheeked Rourke and a steamy, sadomasochistic relationship begins. It lasts the period of the title before the intensity gets too much for her. Joe Cockers version of You Can Leave Your Hat On accompanies a Basinger strip routine and then theres the infamous fridge sequence.
Edit
Delete
10004
976-EVIL II: THE RETURN
1988
JIM WYNORSKI
PATRICK OBRYAN, DEBBIE JAMES
Sequel that still finds the ultimate telephone line, the Devils equivalent of an 0898 number, acquiring victims. The head of a high school taps into the line, hoping to receive supernatural powers to help him in his obsessive lust for a young student. Any one who gets in his way suffers a horrible death. With a supporting role for Brigitte Nielsen as a seductress in league with Satan himself.
Edit
Delete
10005
A BETTER TOMORROW
1986
JOHN WOO
CHOW YUN-FAT, LESLIE CHEUNG
The film which brought international stardom to star Chow Yun-Fat and director John Woo. A roller-coaster of a ride with brothers on both sides of the law becoming united in their quest for personal revenge. After forger Mark (Chow) is crippled and double crossed by an associate, cop Kit (Cheung) risks his badge to help him. Violent and gripping. Also available as a boxed set with a supplementary documentary and booklet.
Edit
Delete
De Niros directorial debut, taken from co-star Palminteris own stage play. De Niro leads a cast of virtual
16:48
10/01/2020

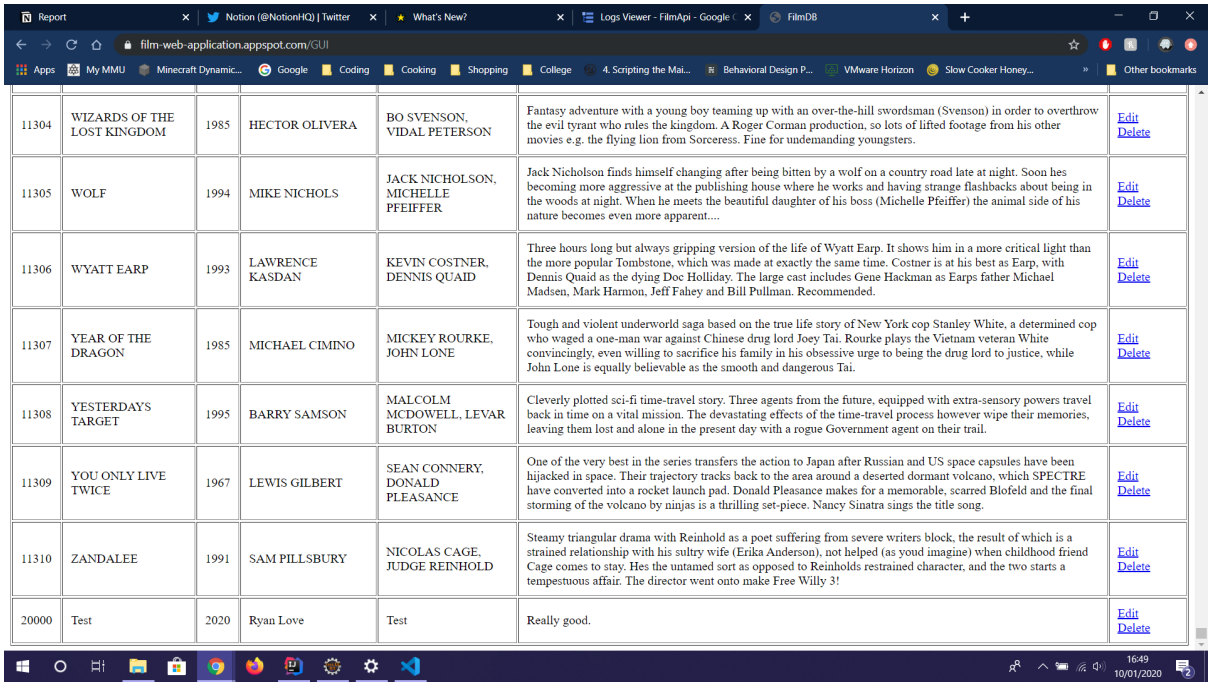
Create


Report
Notion (@NotionHQ) | Twitter
What's New?
Logs Viewer - FilmApi - Google
FilmDB
film-web-application.appspot.com/edit?id=20000
Apps
My MMU
Minecraft Dynamic...
Google
Coding
Cooking
Shopping
College
4. Scripting the Mai...
Behavioral Design P...
VMware Horizon
Slow Cooker Honey...
Other bookmarks
Add New Film Entry
List All Film Entiies
Edit Film
ID:
20000
Title:
Test
Year:
2020
Director:
Ryan Love
Stars:
Test
Review:
Really good.
Save
16:49
10/01/2020



11304	WIZARDS OF THE LOST KINGDOM	1985	HECTOR OLIVERA	BO SVENSON, VIDAL PETERSON	Fantasy adventure with a young boy teaming up with an over-the-hill swordsman (Svenson) in order to overthrow the evil tyrant who rules the kingdom. A Roger Corman production, so lots of lifted footage from his other movies e.g. the flying lion from Sorceress. Fine for undemanding youngsters.	Edit Delete
11305	WOLF	1994	MIKE NICHOLS	JACK NICHOLSON, MICHELLE PFEIFFER	Jack Nicholson finds himself changing after being bitten by a wolf on a country road late at night. Soon hes becoming more aggressive at the publishing house where he works and having strange flashbacks about being in the woods at night. When he meets the beautiful daughter of his boss (Michelle Pfeiffer) the animal side of his nature becomes even more apparent....	Edit Delete
11306	WYATT EARP	1993	LAWRENCE KASDAN	KEVIN COSTNER, DENNIS QUAID	Three hours long but always gripping version of the life of Wyatt Earp. It shows him in a more critical light than the more popular Tombstone, which was made at exactly the same time. Costner is at his best as Earp, with Dennis Quaid as the dying Doc Holliday. The large cast includes Gene Hackman as Earps father Michael Madsen, Mark Harmon, Jeff Fahey and Bill Pullman. Recommended.	Edit Delete
11307	YEAR OF THE DRAGON	1985	MICHAEL CIMINO	MICKEY ROURKE, JOHN LONE	Tough and violent underworld saga based on the true life story of New York cop Stanley White, a determined cop who waged a one-man war against Chinese drug lord Joey Tai. Rourke plays the Vietnam veteran White convincingly, even willing to sacrifice his family in his obsessive urge to being the drug lord to justice, while John Lone is equally believable as the smooth and dangerous Tai.	Edit Delete
11308	YESTERDAYS TARGET	1995	BARRY SAMSON	MALCOLM MCDOWELL, LEVAR BURTON	Cleverly plotted sci-fi time-travel story. Three agents from the future, equipped with extra-sensory powers travel back in time on a vital mission. The devastating effects of the time-travel process however wipe their memories, leaving them lost and alone in the present day with a rogue Government agent on their trail.	Edit Delete
11309	YOU ONLY LIVE TWICE	1967	LEWIS GILBERT	SEAN CONNERY, DONALD PLEASANCE	One of the very best in the series transfers the action to Japan after Russian and US space capsules have been hijacked in space. Their trajectory tracks back to the area around a deserted dormant volcano, which SPECTRE have converted into a rocket launch pad. Donald Pleasance makes for a memorable, scarred Blofeld and the final storming of the volcano by ninjas is a thrilling set-piece. Nancy Sinatra sings the title song.	Edit Delete
11310	ZANDALEE	1991	SAM PILLSBURY	NICOLAS CAGE, JUDGE REINHOLD	Steamy triangular drama with Reinhold as a poet suffering from severe writers block, the result of which is a strained relationship with his sultry wife (Erika Anderson), not helped (as youd imagine) when childhood friend Cage comes to stay. Hes the untamed sort as opposed to Reinholds restrained character, and the two starts a tempestuous affair. The director went onto make Free Willy 3!	Edit Delete
20000	Test	2020	Ryan Love	Test	Really good.	Edit Delete

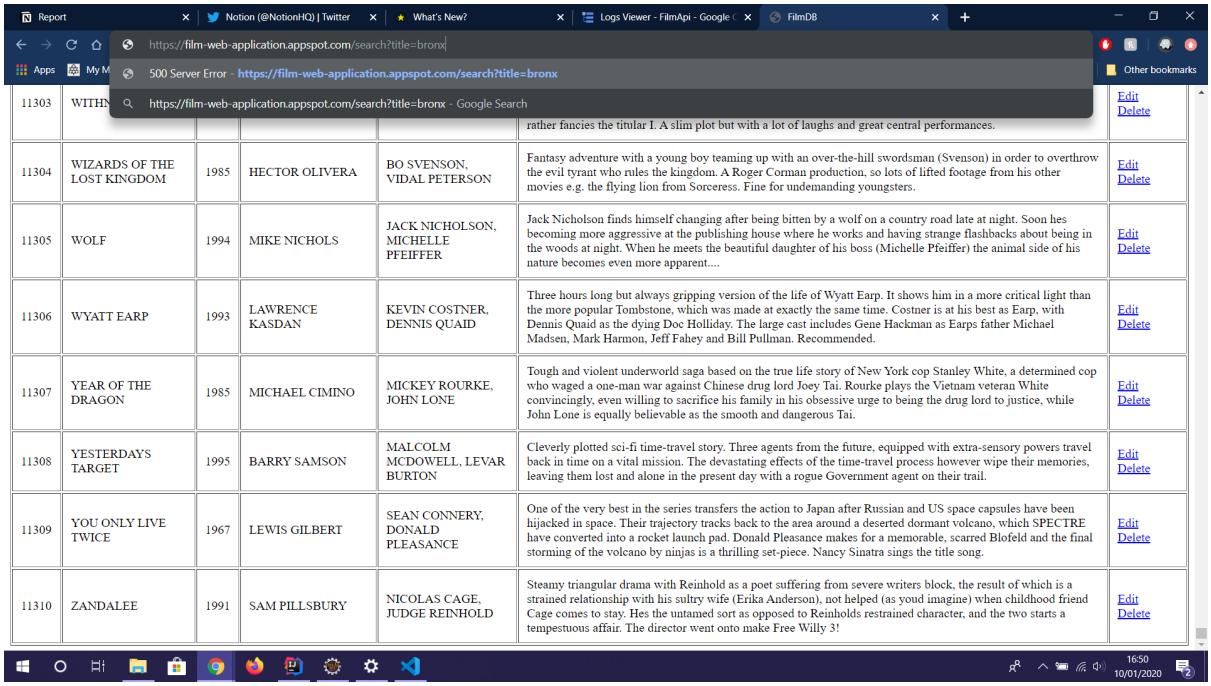
Delete



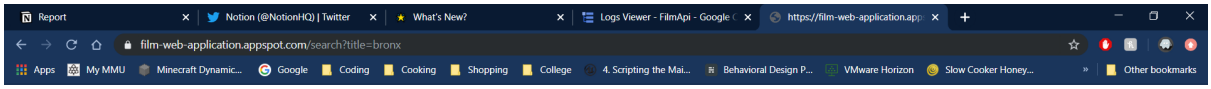
11304	WIZARDS OF THE LOST KINGDOM	1985	HECTOR OLIVERA	BO SVENSON, VIDAL PETERSON	Fantasy adventure with a young boy teaming up with an over-the-hill swordsman (Svenson) in order to overthrow the evil tyrant who rules the kingdom. A Roger Corman production, so lots of lifted footage from his other movies e.g. the flying lion from Sorceress. Fine for undemanding youngsters.	Edit Delete
11305	WOLF	1994	MIKE NICHOLS	JACK NICHOLSON, MICHELLE PFEIFFER	Jack Nicholson finds himself changing after being bitten by a wolf on a country road late at night. Soon hes becoming more aggressive at the publishing house where he works and having strange flashbacks about being in the woods at night. When he meets the beautiful daughter of his boss (Michelle Pfeiffer) the animal side of his nature becomes even more apparent....	Edit Delete
11306	WYATT EARP	1993	LAWRENCE KASDAN	KEVIN COSTNER, DENNIS QUAID	Three hours long but always gripping version of the life of Wyatt Earp. It shows him in a more critical light than the more popular Tombstone, which was made at exactly the same time. Costner is at his best as Earp, with Dennis Quaid as the dying Doc Holliday. The large cast includes Gene Hackman as Earps father Michael Madsen, Mark Harmon, Jeff Fahey and Bill Pullman. Recommended.	Edit Delete
11307	YEAR OF THE DRAGON	1985	MICHAEL CIMINO	MICKEY ROURKE, JOHN LONE	Tough and violent underworld saga based on the true life story of New York cop Stanley White, a determined cop who waged a one-man war against Chinese drug lord Joey Tai. Rourke plays the Vietnam veteran White convincingly, even willing to sacrifice his family in his obsessive urge to being the drug lord to justice, while John Lone is equally believable as the smooth and dangerous Tai.	Edit Delete
11308	YESTERDAYS TARGET	1995	BARRY SAMSON	MALCOLM MCDOWELL, LEVAR BURTON	Cleverly plotted sci-fi time-travel story. Three agents from the future, equipped with extra-sensory powers travel back in time on a vital mission. The devastating effects of the time-travel process however wipe their memories, leaving them lost and alone in the present day with a rogue Government agent on their trail.	Edit Delete
11309	YOU ONLY LIVE TWICE	1967	LEWIS GILBERT	SEAN CONNERY, DONALD PLEASANCE	One of the very best in the series transfers the action to Japan after Russian and US space capsules have been hijacked in space. Their trajectory tracks back to the area around a deserted dormant volcano, which SPECTRE have converted into a rocket launch pad. Donald Pleasance makes for a memorable, scarred Blofeld and the final storming of the volcano by ninjas is a thrilling set-piece. Nancy Sinatra sings the title song.	Edit Delete
11310	ZANDALEE	1991	SAM PILLSBURY	NICOLAS CAGE, JUDGE REINHOLD	Steamy triangular drama with Reinhold as a poet suffering from severe writers block, the result of which is a strained relationship with his sultry wife (Erika Anderson), not helped (as youd imagine) when childhood friend Cage comes to stay. Hes the untamed sort as opposed to Reinholds restrained character, and the two starts a tempestuous affair. The director went onto make Free Willy 3!	Edit Delete
20000	Test	2020	Ryan Love	Test	Really good.	Edit Delete

11303	WITHNAIL & I	1986	BRUCE ROBINSON	RICHARD E. GRANT, PAUL MC GANN	struggling actors in the 60s, lurching from one drunken or drug-fuelled revel to the next. Fancying a break they decide to stay at Withnails Uncle Montys cottage in the country, not expecting him to be there. But he is, and he rather fancies the titular I. A slim plot but with a lot of laughs and great central performances.	Edit Delete
11304	WIZARDS OF THE LOST KINGDOM	1985	HECTOR OLIVERA	BO SVENSON, VIDAL PETERSON	Fantasy adventure with a young boy teaming up with an over-the-hill swordsman (Svenson) in order to overthrow the evil tyrant who rules the kingdom. A Roger Corman production, so lots of lifted footage from his other movies e.g. the flying lion from Sorceress. Fine for undemanding youngsters.	Edit Delete
11305	WOLF	1994	MIKE NICHOLS	JACK NICHOLSON, MICHELLE PFEIFFER	Jack Nicholson finds himself changing after being bitten by a wolf on a country road late at night. Soon hes becoming more aggressive at the publishing house where he works and having strange flashbacks about being in the woods at night. When he meets the beautiful daughter of his boss (Michelle Pfeiffer) the animal side of his nature becomes even more apparent....	Edit Delete
11306	WYATT EARP	1993	LAWRENCE KASDAN	KEVIN COSTNER, DENNIS QUAID	Three hours long but always gripping version of the life of Wyatt Earp. It shows him in a more critical light than the more popular Tombstone, which was made at exactly the same time. Costner is at his best as Earp, with Dennis Quaid as the dying Doc Holliday. The large cast includes Gene Hackman as Earps father Michael Madsen, Mark Harmon, Jeff Fahey and Bill Pullman. Recommended.	Edit Delete
11307	YEAR OF THE DRAGON	1985	MICHAEL CIMINO	MICKEY ROURKE, JOHN LONE	Tough and violent underworld saga based on the true life story of New York cop Stanley White, a determined cop who waged a one-man war against Chinese drug lord Joey Tai. Rourke plays the Vietnam veteran White convincingly, even willing to sacrifice his family in his obsessive urge to being the drug lord to justice, while John Lone is equally believable as the smooth and dangerous Tai.	Edit Delete
11308	YESTERDAYS TARGET	1995	BARRY SAMSON	MALCOLM MCDOWELL, LEVAR BURTON	Cleverly plotted sci-fi time-travel story. Three agents from the future, equipped with extra-sensory powers travel back in time on a vital mission. The devastating effects of the time-travel process however wipe their memories, leaving them lost and alone in the present day with a rogue Government agent on their trail.	Edit Delete
11309	YOU ONLY LIVE TWICE	1967	LEWIS GILBERT	SEAN CONNERY, DONALD PLEASANCE	One of the very best in the series transfers the action to Japan after Russian and US space capsules have been hijacked in space. Their trajectory tracks back to the area around a deserted dormant volcano, which SPECTRE have converted into a rocket launch pad. Donald Pleasance makes for a memorable, scarred Blofeld and the final storming of the volcano by ninjas is a thrilling set-piece. Nancy Sinatra sings the title song.	Edit Delete
11310	ZANDALEE	1991	SAM PILLSBURY	NICOLAS CAGE, JUDGE REINHOLD	Steamy triangular drama with Reinhold as a poet suffering from severe writers block, the result of which is a strained relationship with his sultry wife (Erika Anderson), not helped (as youd imagine) when childhood friend Cage comes to stay. Hes the untamed sort as opposed to Reinholds restrained character, and the two starts a tempestuous affair. The director went onto make Free Willy 3!	Edit Delete

Search



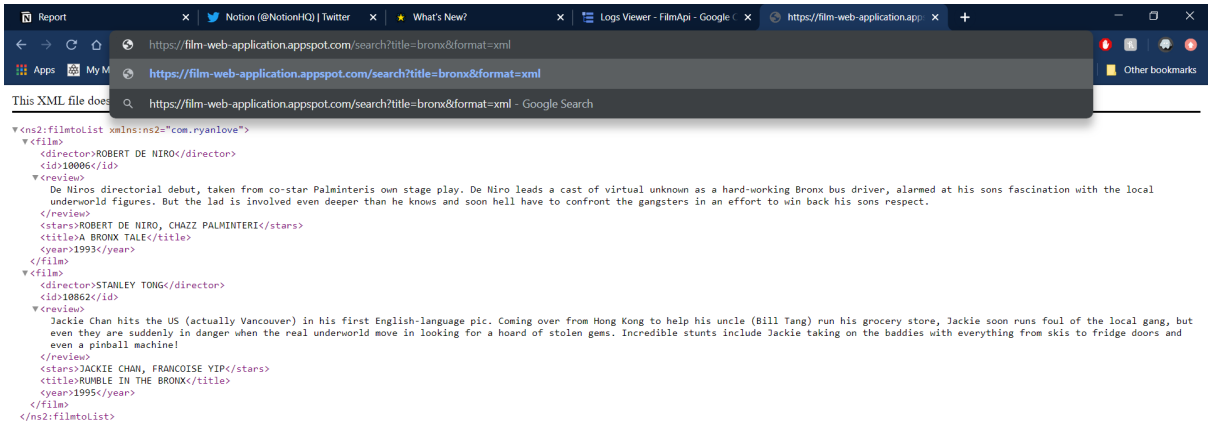
11303	WITH				rather fancies the titular I. A slim plot but with a lot of laughs and great central performances.	Edit Delete
11304	WIZARDS OF THE LOST KINGDOM	1985	HECTOR OLIVERA	BO SVENSON, VIDAL PETERSON	Fantasy adventure with a young boy teaming up with an over-the-hill swordsman (Svenson) in order to overthrow the evil tyrant who rules the kingdom. A Roger Corman production, so lots of lifted footage from his other movies e.g. the flying lion from Sorceress. Fine for undemanding youngsters.	Edit Delete
11305	WOLF	1994	MIKE NICHOLS	JACK NICHOLSON, MICHELLE PFEIFFER	Jack Nicholson finds himself changing after being bitten by a wolf on a country road late at night. Soon hes becoming more aggressive at the publishing house where he works and having strange flashbacks about being in the woods at night. When he meets the beautiful daughter of his boss (Michelle Pfeiffer) the animal side of his nature becomes even more apparent....	Edit Delete
11306	WYATT EARP	1993	LAWRENCE KASDAN	KEVIN COSTNER, DENNIS QUAID	Three hours long but always gripping version of the life of Wyatt Earp. It shows him in a more critical light than the more popular Tombstone, which was made at exactly the same time. Costner is at his best as Earp, with Dennis Quaid as the dying Doc Holliday. The large cast includes Gene Hackman as Earps father Michael Madsen, Mark Harmon, Jeff Fahey and Bill Pullman. Recommended.	Edit Delete
11307	YEAR OF THE DRAGON	1985	MICHAEL CIMINO	MICKEY ROURKE, JOHN LONE	Tough and violent underworld saga based on the true life story of New York cop Stanley White, a determined cop who waged a one-man war against Chinese drug lord Joey Tai. Rourke plays the Vietnam veteran White convincingly, even willing to sacrifice his family in his obsessive urge to being the drug lord to justice, while John Lone is equally believable as the smooth and dangerous Tai.	Edit Delete
11308	YESTERDAYS TARGET	1995	BARRY SAMSON	MALCOLM MCDOWELL, LEVAR BURTON	Cleverly plotted sci-fi time-travel story. Three agents from the future, equipped with extra-sensory powers travel back in time on a vital mission. The devastating effects of the time-travel process however wipe their memories, leaving them lost and alone in the present day with a rogue Government agent on their trail.	Edit Delete
11309	YOU ONLY LIVE TWICE	1967	LEWIS GILBERT	SEAN CONNERY, DONALD PLEASANCE	One of the very best in the series transfers the action to Japan after Russian and US space capsules have been hijacked in space. Their trajectory tracks back to the area around a deserted dormant volcano, which SPECTRE have converted into a rocket launch pad. Donald Pleasance makes for a memorable, scarred Blofeld and the final storming of the volcano by ninjas is a thrilling set-piece. Nancy Sinatra sings the title song.	Edit Delete
11310	ZANDALEE	1991	SAM PILLSBURY	NICOLAS CAGE, JUDGE REINHOLD	Steamy triangular drama with Reinhold as a poet suffering from severe writers block, the result of which is a strained relationship with his sultry wife (Erika Anderson), not helped (as youd imagine) when childhood friend Cage comes to stay. Hes the untamed sort as opposed to Reinholds restrained character, and the two starts a tempestuous affair. The director went onto make Free Willy 3!	Edit Delete



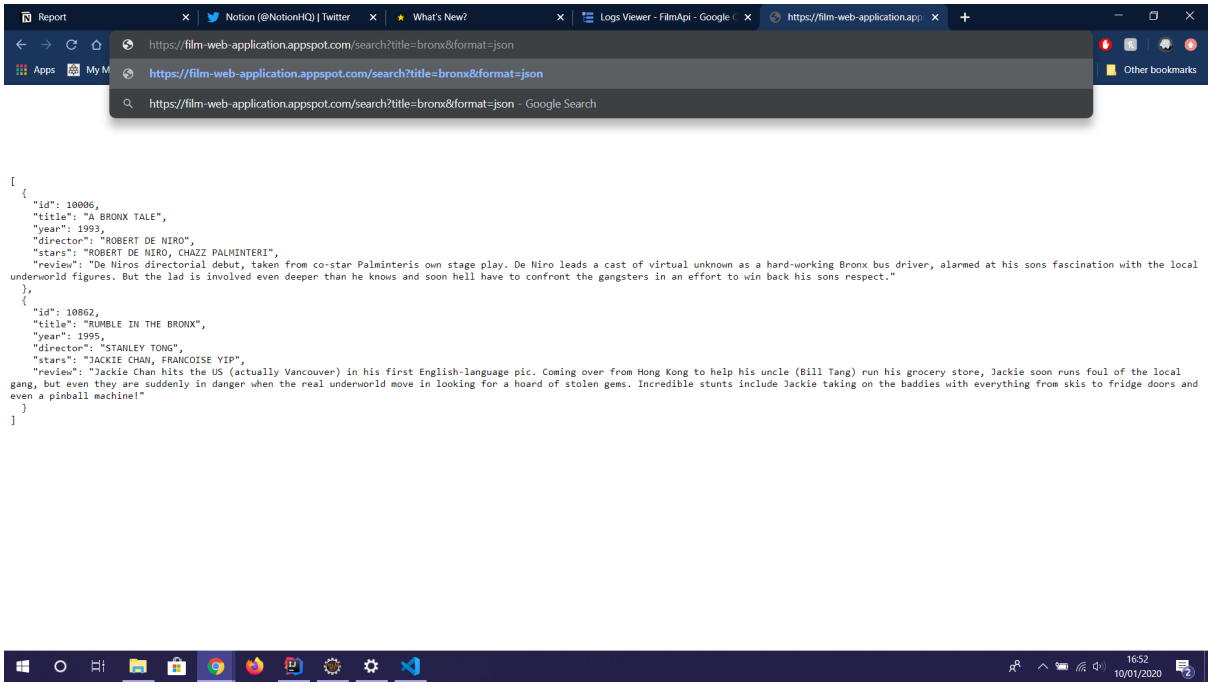

```
[
  {
    "id": 10006,
    "title": "A BRONX TALE",
    "year": 1993,
    "director": "ROBERT DE NIRO",
    "stars": "ROBERT DE NIRO, CHAZZ PALMINTERI",
    "review": "De Niros directorial debut, taken from co-star Palminteris own stage play. De Niro leads a cast of virtual unknown as a hard-working Bronx bus driver, alarmed at his sons fascination with the local underworld figures. But the lad is involved even deeper than he knows and soon hell have to confront the gangsters in an effort to win back his sons respect."
  },
  {
    "id": 10862,
    "title": "RUMBLE IN THE BRONX",
    "year": 1995,
    "director": "STANLEY TONG",
    "stars": "JACKIE CHAN, FRANCOISE YIP",
    "review": "Jackie Chan hits the US (actually Vancouver) in his first English-language pic. Coming over from Hong Kong to help his uncle (Bill Tang) run his grocery store, Jackie soon runs foul of the local gang, but even they are suddenly in danger when the real underworld move in looking for a hoard of stolen gems. Incredible stunts include Jackie taking on the baddies with everything from skis to fridge doors and even a pinball machine!"
  }
]
```


Search Formats

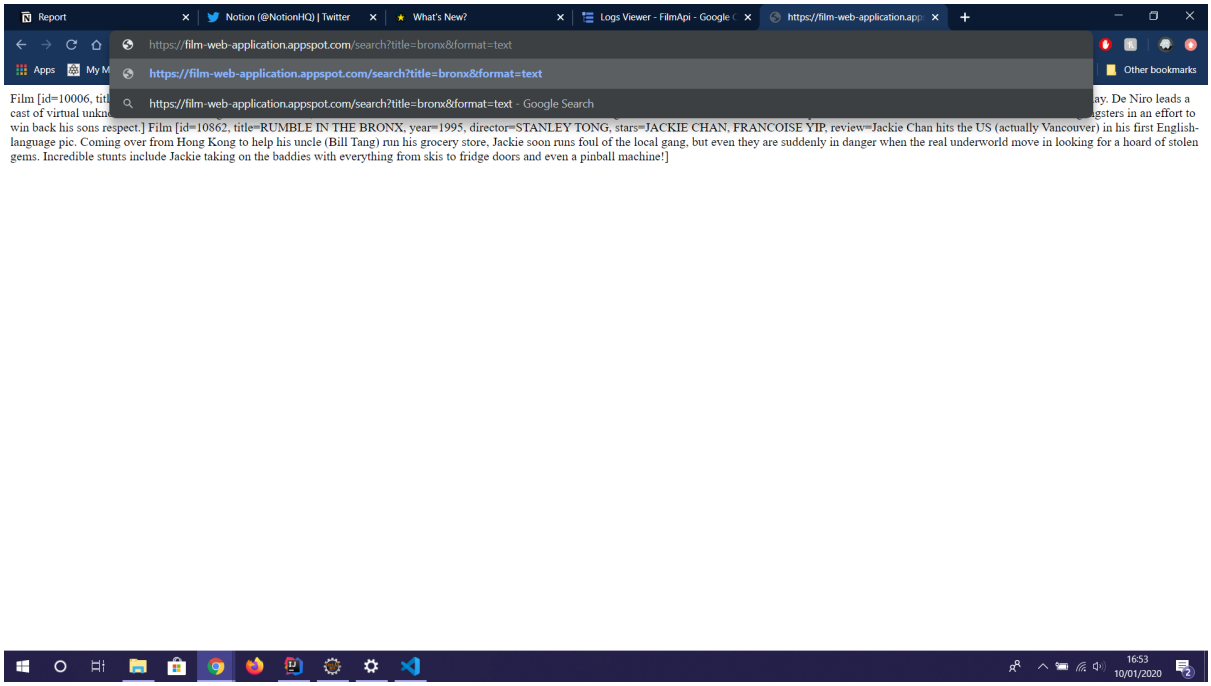
XML



JSON

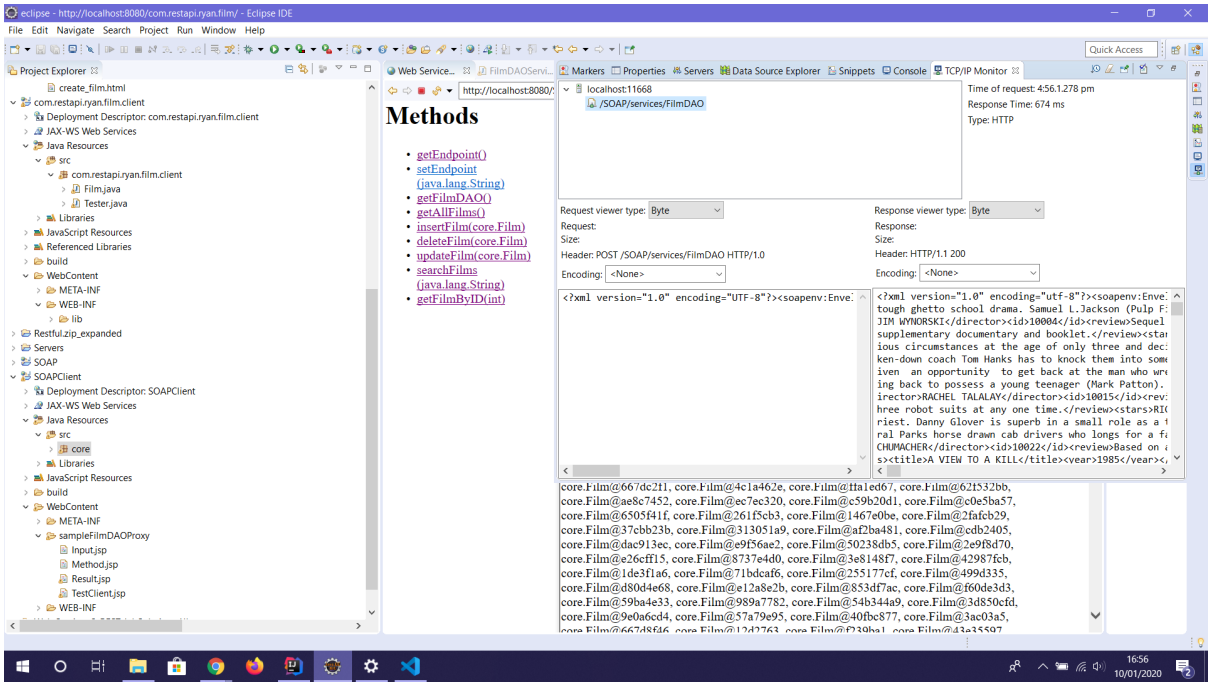


Text

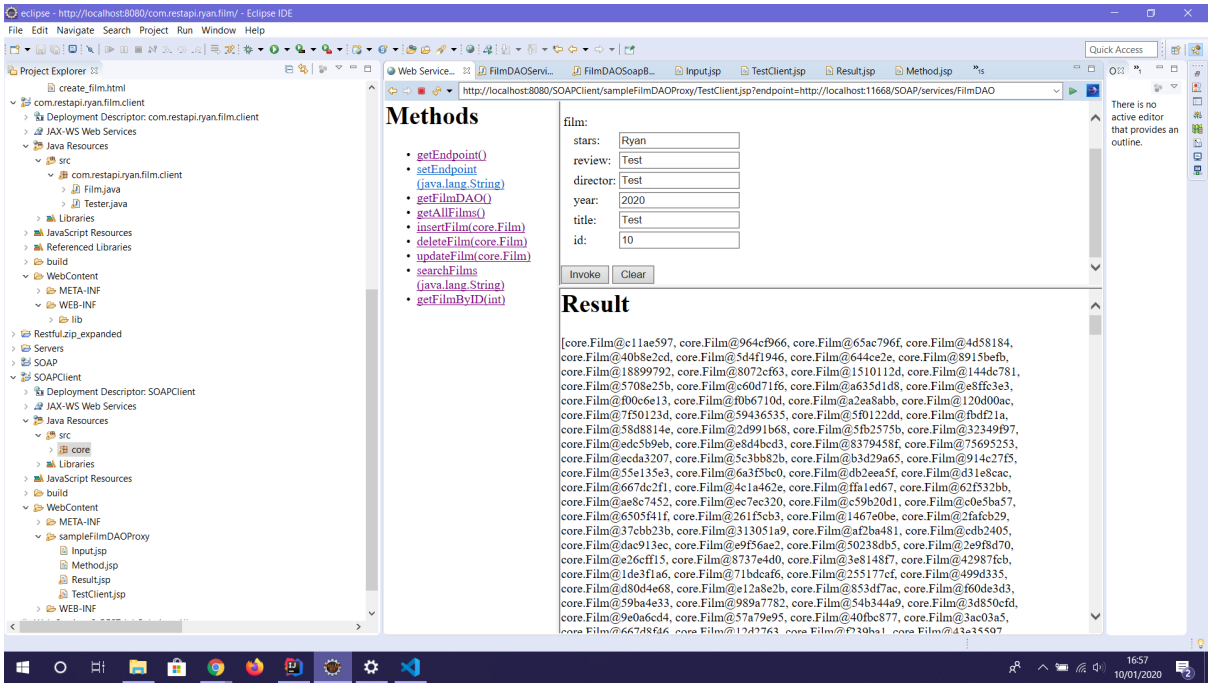


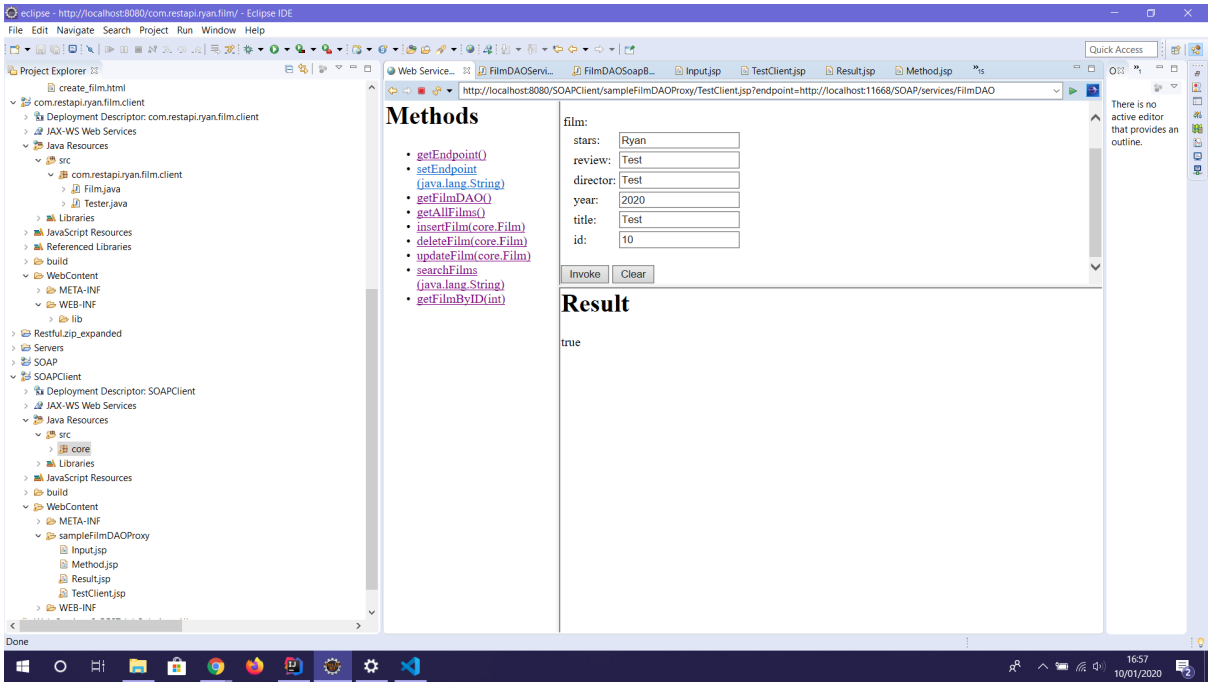
SOAP

Read

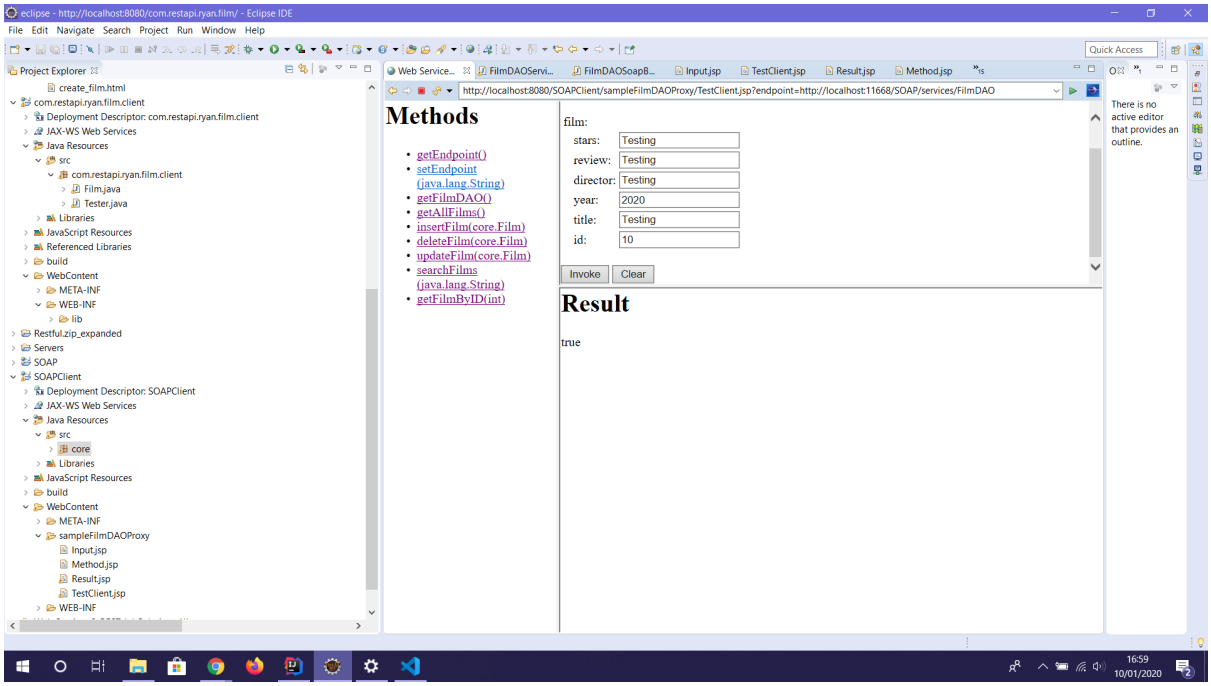

eclipse - http://localhost:8080/com.restapi.ryan.film/ - Eclipse IDE
Methods
getEndpoint()
setEndpoint (java.lang.String)
getFilmDAO()
getAllFilms()
insertFilm(core.Film)
deleteFilm(core.Film)
updateFilm(core.Film)
searchFilms (java.lang.String)
getFilmByID(int)

Create

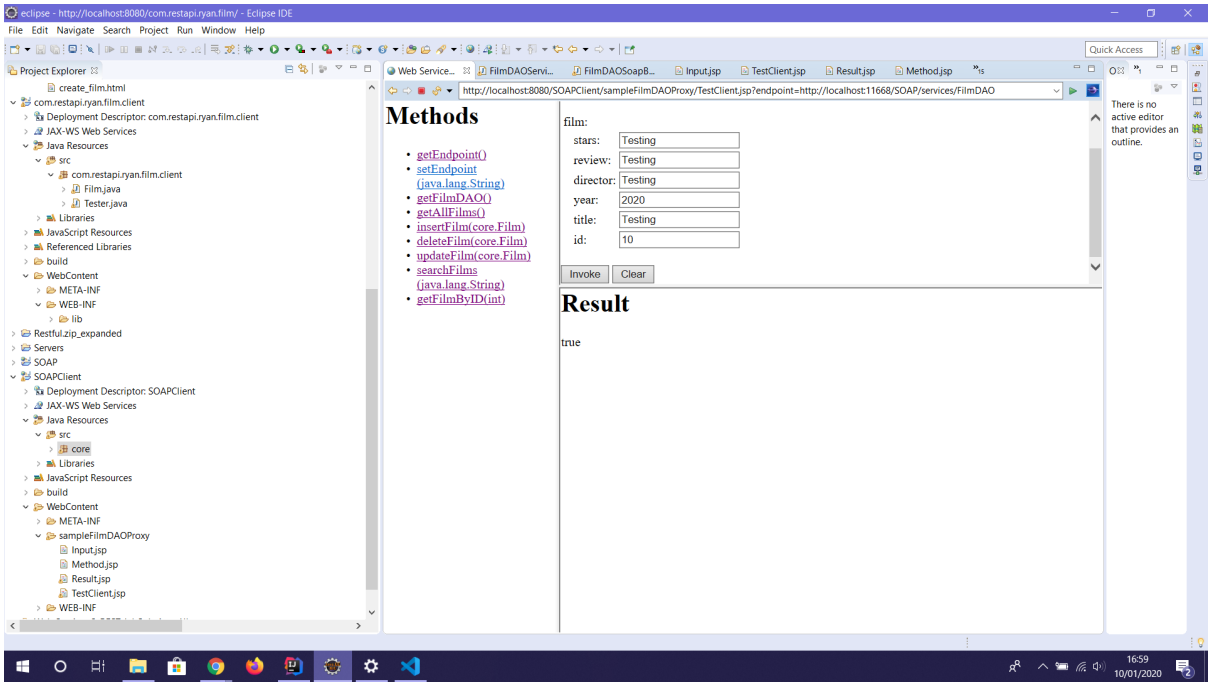

eclipse - http://localhost:8080/com.restapi.ryan.film/ - Eclipse IDE
Methods
getEndpoint()
setEndpoint (java.lang.String)
getFilmDAO()
getAllFilms()
insertFilm(core.Film)
deleteFilm(core.Film)
updateFilm(core.Film)
searchFilms (java.lang.String)
getFilmByID(int)
film:
stars: Ryan
review: Test
director: Test
year: 2020
title: Test
id: 10
Invoke
Clear
Result



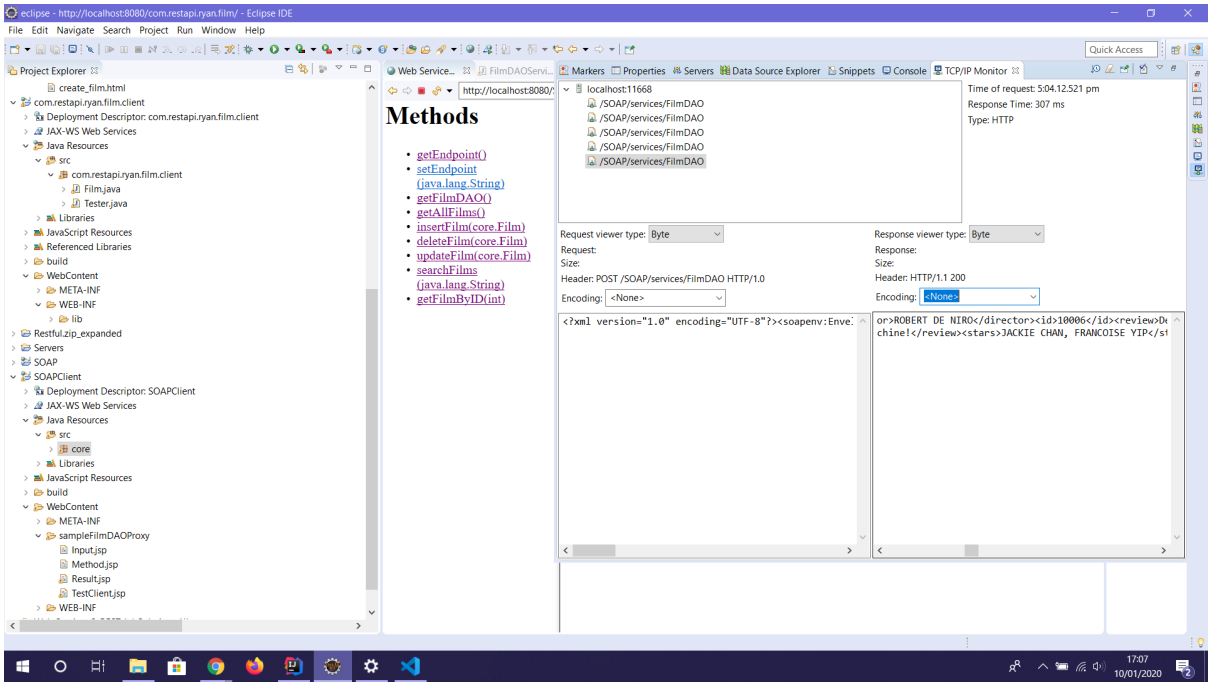
Update



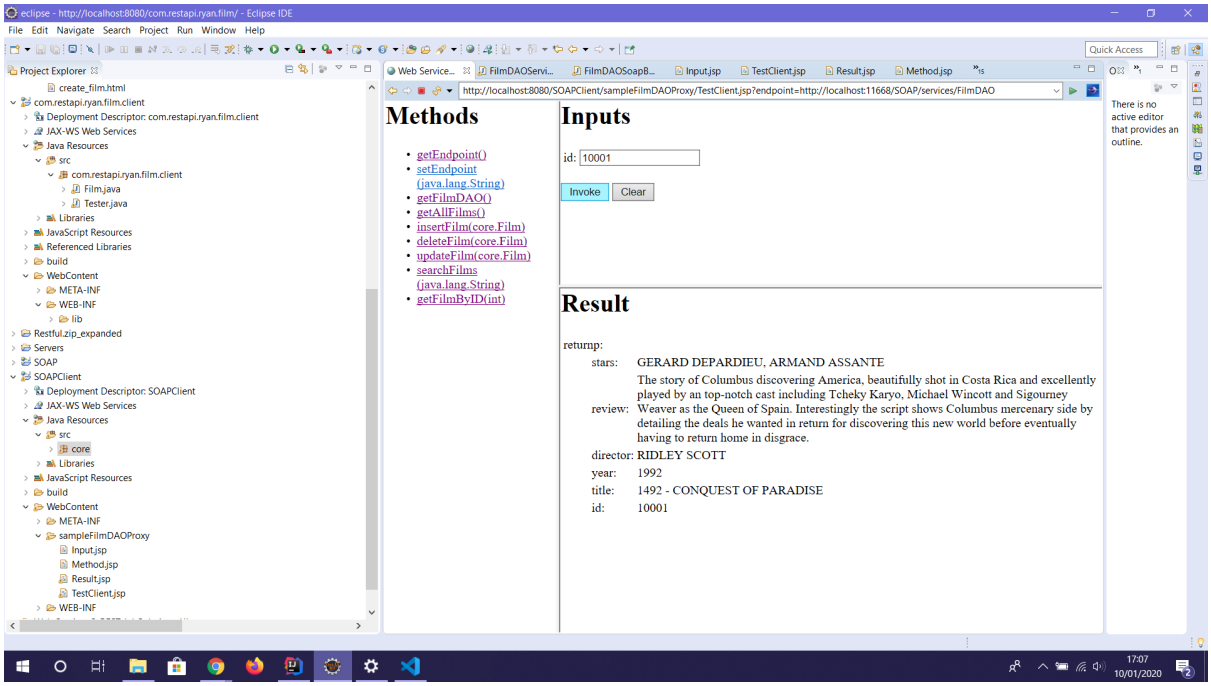
Delete


eclipse - http://localhost:8080/com.restapi.ryan.film/ - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
create_film.html
com.restapi.ryan.film.client
Deployment Descriptor: com.restapi.ryan.film.client
JAX-WS Web Services
Java Resources
src
com.restapi.ryan.film.client
Film.java
Tester.java
Libraries
JavaScript Resources
Referenced Libraries
build
WebContent
META-INF
WEB-INF
lib
Restful.zip_expanded
Servers
SOAP
SOAPClient
Deployment Descriptor: SOAPClient
JAX-WS Web Services
Java Resources
src
core
Libraries
JavaScript Resources
build
WebContent
META-INF
sampleFilmDAOProxy
Input.jsp
Method.jsp
Result.jsp
TestClient.jsp
WEB-INF
Web Service...
FilmDAOServi...
FilmDAOSoapB...
Input.jsp
TestClient.jsp
Result.jsp
Method.jsp
http://localhost:8080/SOAPClient/sampleFilmDAOProxy/TestClient.jsp?endpoint=http://localhost:11668/SOAP/services/FilmDAO
Methods
getEndpoint()
setEndpoint (java.lang.String)
getFilmDAO()
getAllFilms()
insertFilm(core.Film)
deleteFilm(core.Film)
updateFilm(core.Film)
searchFilms (java.lang.String)
getFilmByID(int)
film:
stars: Testing
review: Testing
director: Testing
year: 2020
title: Testing
id: 10
Invoke
Clear
Result
true
There is no active editor that provides an outline.
16:59
10/01/2020

Search

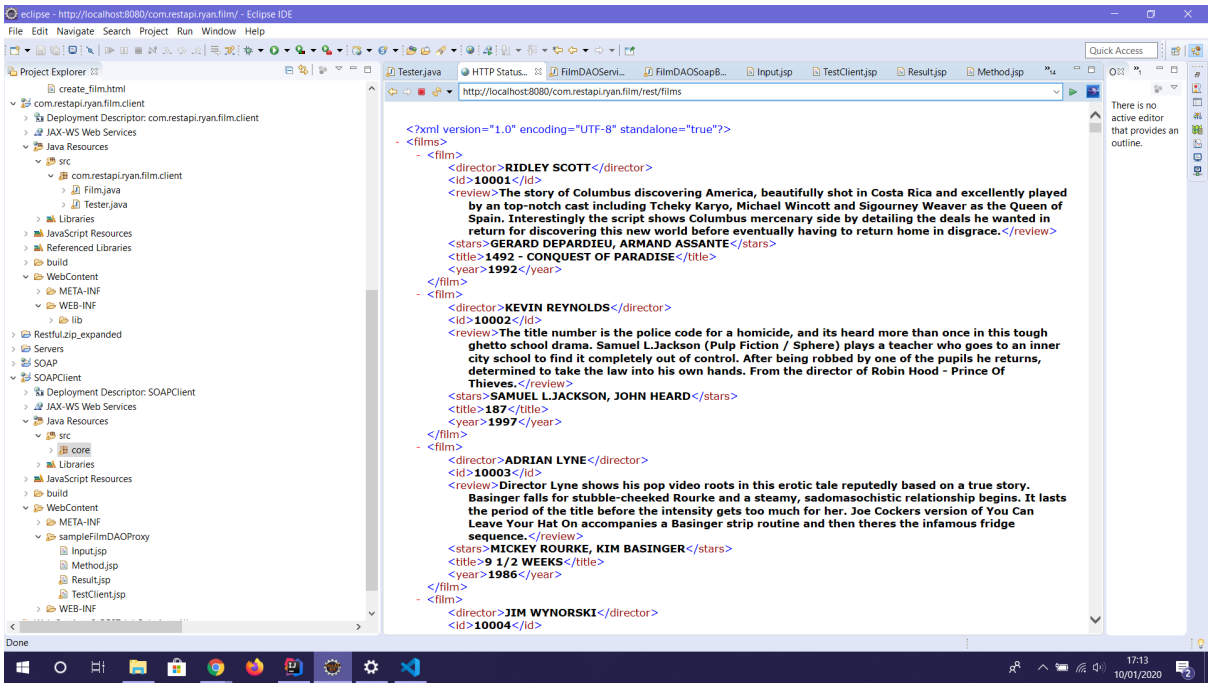

eclipse - http://localhost:8080/com.restapi.ryan.film/ - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Web Service...
FilmDAOServi...
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
TCP/IP Monitor
http://localhost:8080/
Methods
getEndpoint()
setEndpoint (java.lang.String)
getFilmDAO()
getAllFilms()
insertFilm(core.Film)
deleteFilm(core.Film)
updateFilm(core.Film)
searchFilms (java.lang.String)
getFilmByID(int)
localhost:11668
/SOAP/services/FilmDAO
/SOAP/services/FilmDAO
/SOAP/services/FilmDAO
/SOAP/services/FilmDAO
/SOAP/services/FilmDAO
Time of request: 5:04.12.521 pm
Response Time: 307 ms
Type: HTTP
Request viewer type: Byte
Request:
Size:
Header: POST /SOAP/services/FilmDAO HTTP/1.0
Encoding: <None>
Response viewer type: Byte
Response:
Size:
Header: HTTP/1.1 200
Encoding: <None>
<?xml version="1.0" encoding="UTF-8"?><soapenv:Enve:
or>ROBERT DE NIRO</director><id>10006</id><review>De
chine!</review><stars>JACKIE CHAN, FRANCOISE YIP</st
17:07
10/01/2020

ID

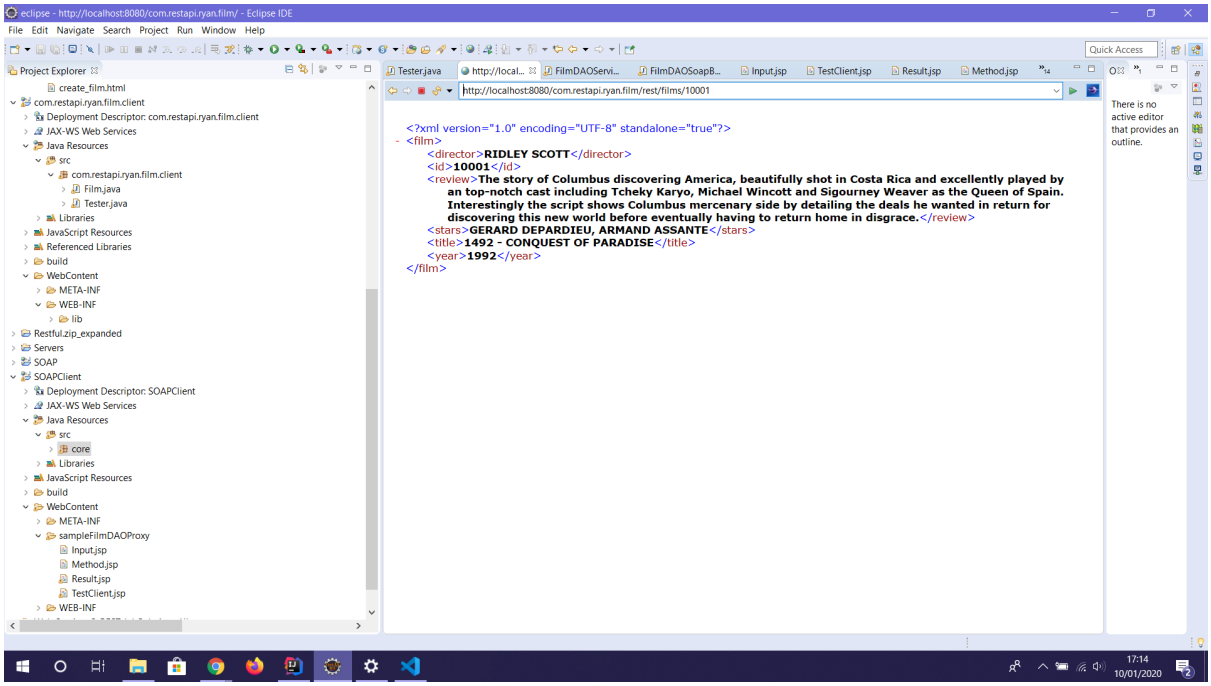


REST

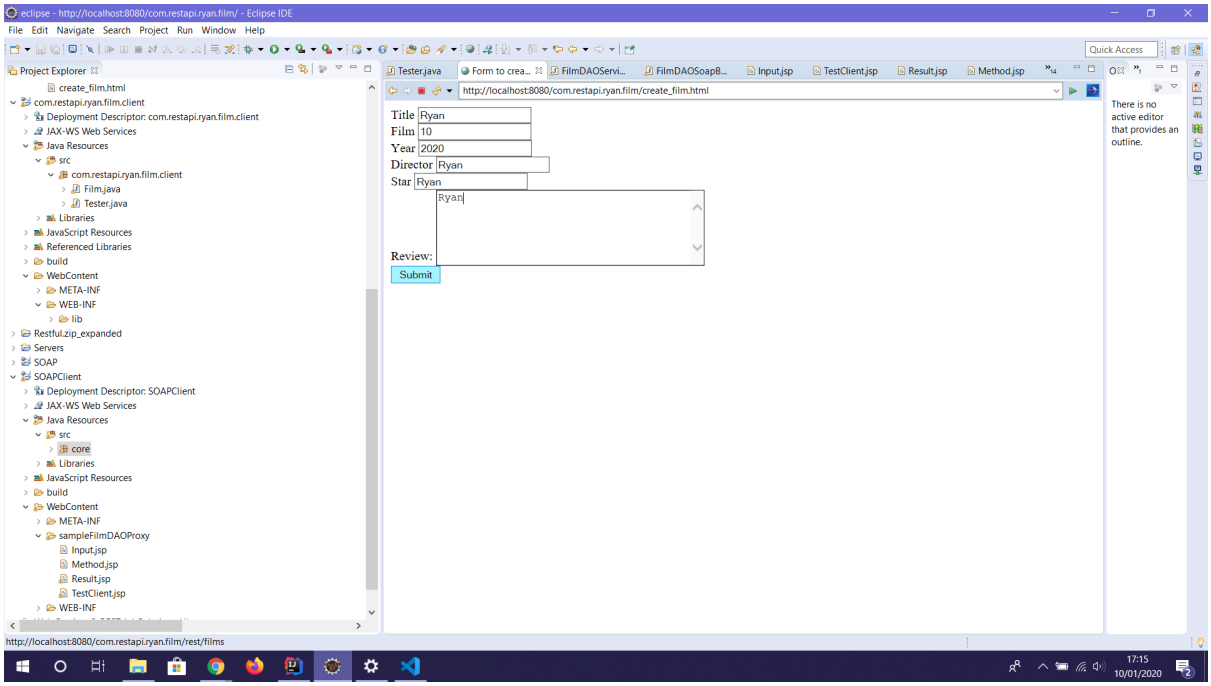
Get

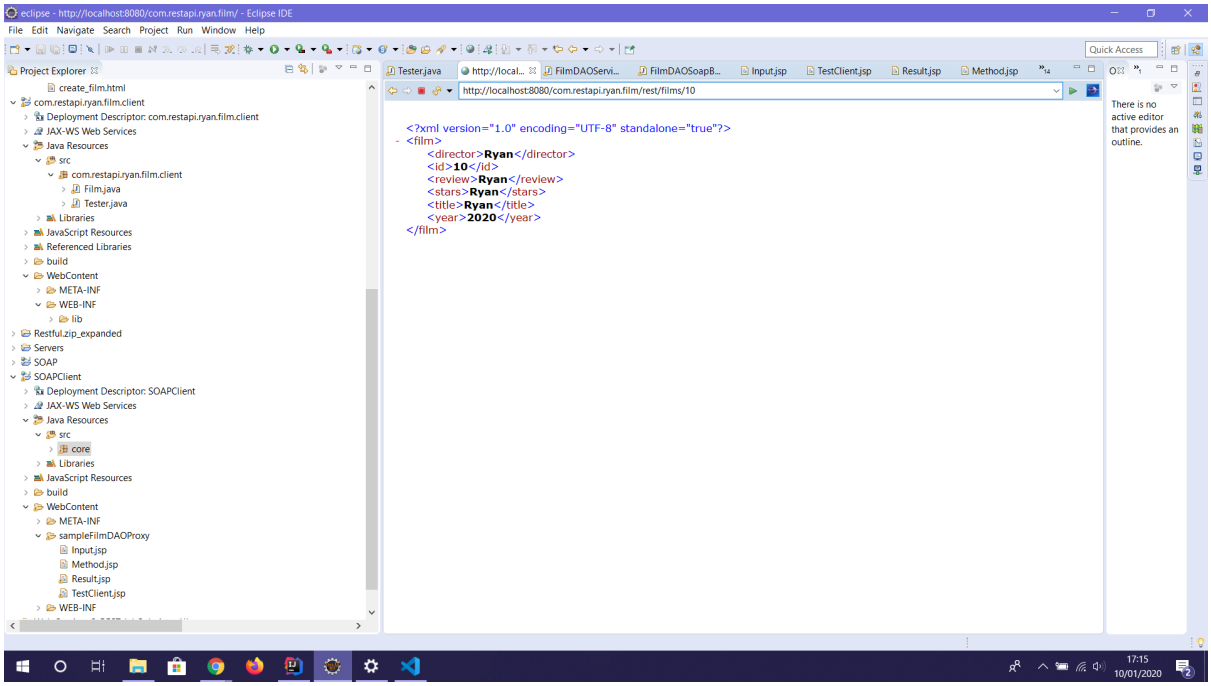


Get ID/Search


eclipse - http://localhost:8080/com.restapi.ryan.film/ - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
http://localhost:8080/com.restapi.ryan.film/rest/films/10001
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<film>
<director>RIDLEY SCOTT</director>
<id>10001</id>
<review>The story of Columbus discovering America, beautifully shot in Costa Rica and excellently played by an top-notch cast including Tcheky Karyo, Michael Wincott and Sigourney Weaver as the Queen of Spain. Interestingly the script shows Columbus mercenary side by detailing the deals he wanted in return for discovering this new world before eventually having to return home in disgrace.</review>
<stars>GERARD DEPARDIEU, ARMAND ASSANTE</stars>
<title>1492 - CONQUEST OF PARADISE</title>
<year>1992</year>
</film>
There is no active editor that provides an outline.
17:14
10/01/2020

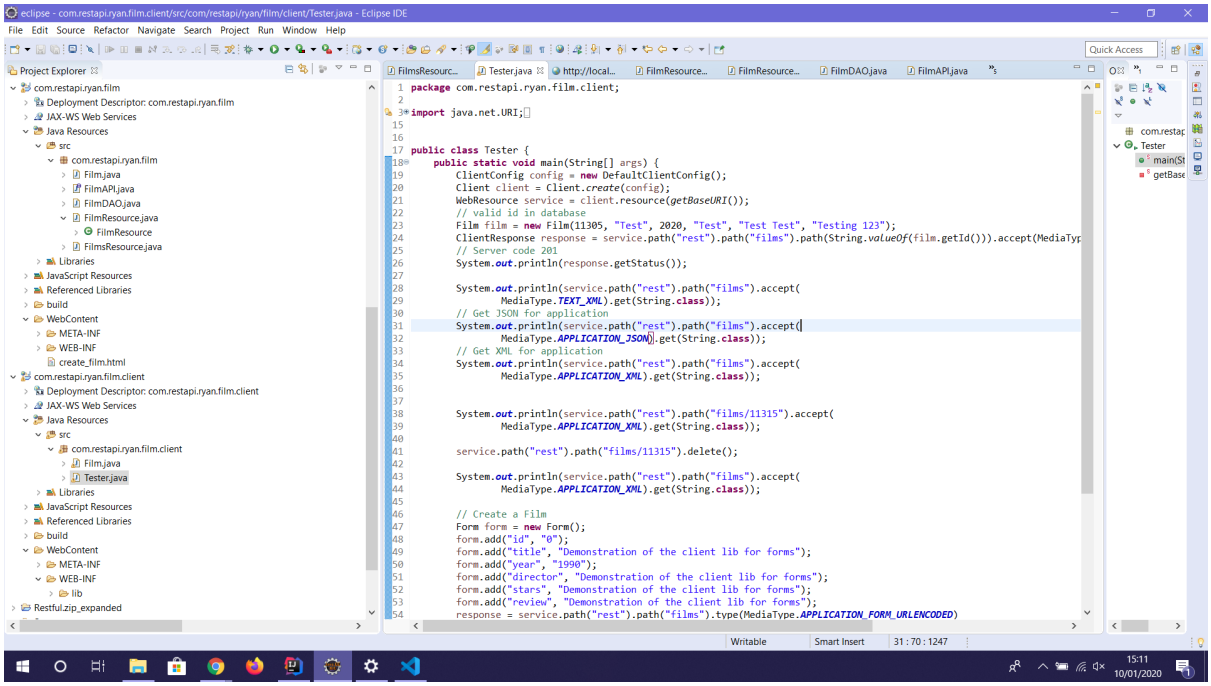
Create

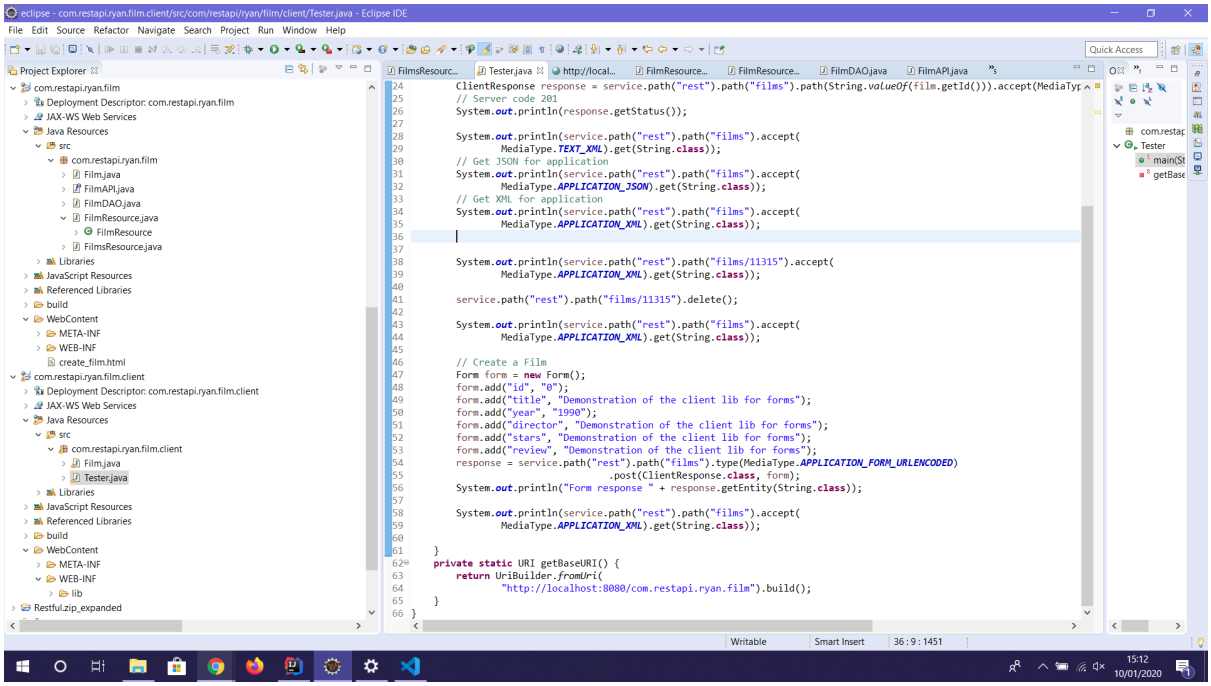

eclipse - http://localhost:8080/com.restapi.ryan.film/ - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
http://localhost:8080/com.restapi.ryan.film/create_film.html
Title Ryan
Film 10
Year 2020
Director Ryan
Star Ryan
Ryan
Review:
Submit
There is no active editor that provides an outline.
http://localhost:8080/com.restapi.ryan.film/rest/films
17:15
10/01/2020



Update and Delete

This test is preformed in the tester file that is shown below that is not showing this request in the console due to the speed and actions it is doing with the update and delete working in this tester class but once this file has been ran the id is no longer visible to the user hence when rerunning the application an error will appear if a valid id is not provided in the tester file for the film constructor and the endpoints for update and delete.





References

A Simple Guide to Connection Pooling in Java | Baeldung. (2019) Baeldung. [Online] [Accessed on 10 January 2020] https://www.baeldung.com/java-connection-pooling.

Denning, S. (2016) What Is Agile?. Forbes.com. [Online] [Accessed on 10 January 2020] https://www.forbes.com/sites/stevedenning/2016/08/13/what-is-agile/.

Google Cloud Computing, Hosting Services & APIs | Google Cloud. (2020) Google Cloud. [Online] [Accessed on 10 January 2020] https://cloud.google.com/gcp/?utm_source=google&utm_medium=cpc&utm_campaign=emea-gb-all-en-dr-bkws-all-all-trial-e-gcp-1008073&utm_content=text-ad-none-any-DEV_c-CRE_166514091269-ADGP_Hybrid+%7C+AW+SEM+%7C+BKWS+~+EXA_M:1_GB_EN_General_Cloud_google+in+the+cloud-KWID_43700045189079410-aud-606988878854:kwd-6458750523-userloc_1006904&utm_term=KW_google%20cloudg&ds_rl=1242853&ds_rl=1245734&ds_rl=1245734&gclid=CjwKCAiA3

IBM Cloud. (2020) Ibm.com. [Online] [Accessed on 10 January 2020] https://www.ibm.com/uk-en/cloud.

Malik, D. (2019) A Brief Guide to Clean Code: Naming Conventions - DEV Community 👭. Dev.to. [Online] [Accessed on 10 January 2020] https://dev.to/danialmalik/a-beginner-s-guide-to-clean-code-part1-naming-conventions-139l.

Mardan, A. (2020) Top 10 ES6 Features Every Busy JavaScript Developer Must Know. Webapplog.com. [Online] [Accessed on 10 January 2020] https://webapplog.com/es6/.

Pal, K. (2020) Importance of code indentation. Mrbool.com. [Online] [Accessed on 10 January 2020] http://mrbool.com/importance-of-code-indentation/29079.

SOAP Vs. REST: Difference between Web API Services. (2020) Guru99.com. [Online] [Accessed on 10 January 2020] https://www.guru99.com/comparison-between-web-services.html.

Sourour, B. (2017) Putting comments in code: the good, the bad, and the ugly.. freeCodeCamp.org. [Online] [Accessed on 10 January 2020] https://www.freecodecamp.org/news/code-comments-the-good-the-bad-and-the-ugly-be9cc65fbf83/.

Using a JavaScript library versus building the functionality yourself. (2018) JavaScript Code Readability. [Online] [Accessed on 10 January 2020] https://www.codereadability.com/javascript-library-vs-building-yourself/.

Wattis, A. (2017) What is MVC, and how is it like a sandwich shop?. freeCodeCamp.org. [Online] [Accessed on 10 January 2020] https://www.freecodecamp.org/news/simplified-explanation-to-mvc-5d307796df30/.

What is Scrum?. (2020) Scrum.org. [Online] [Accessed on 10 January 2020] https://www.scrum.org/resources/what-is-scrum.

Yeri, P. (2019) under_scores, camelCase and PascalCase - The three naming conventions every programmer should be aware of - DEV Community 👩‍👧‍👦. Dev.to. [Online] [Accessed on 10 January 2020] https://dev.to/prahladyeri/underscores-camelcasing-and-pascalcasing-the-three-naming-conventions-every-programmer-should-be-aware-of-3aed.