**EX 3**  **IMPLEMENTATION OF SOAP**

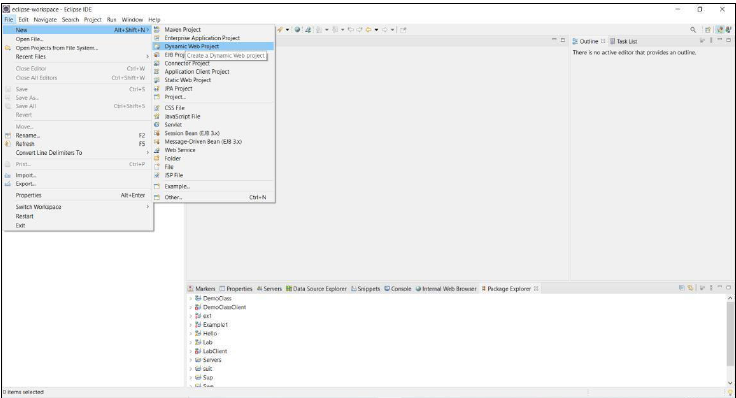
**01-04-22**  **AND REST**

**AIM :**

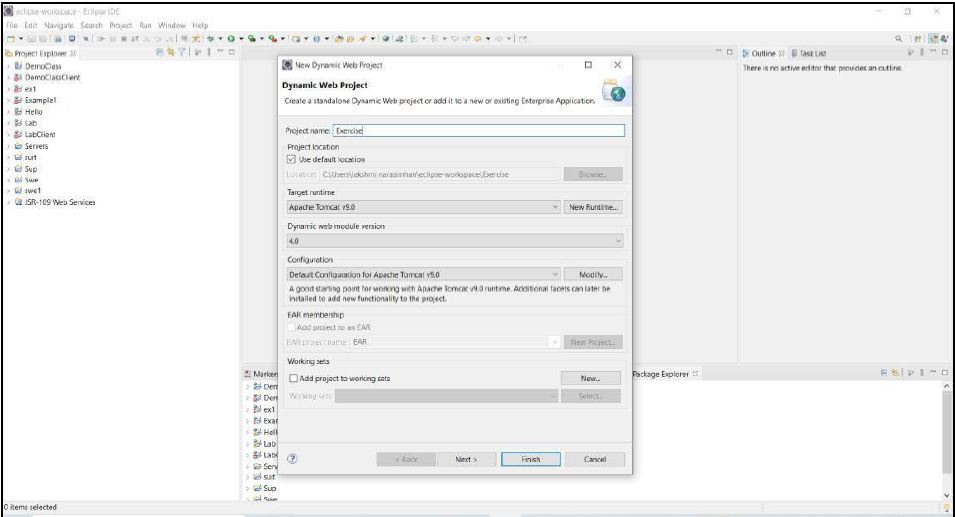
To implement SOAP and REST using java

**SOAP:**

**Create a new Dynamic Web Project in Eclipse IDE**

****

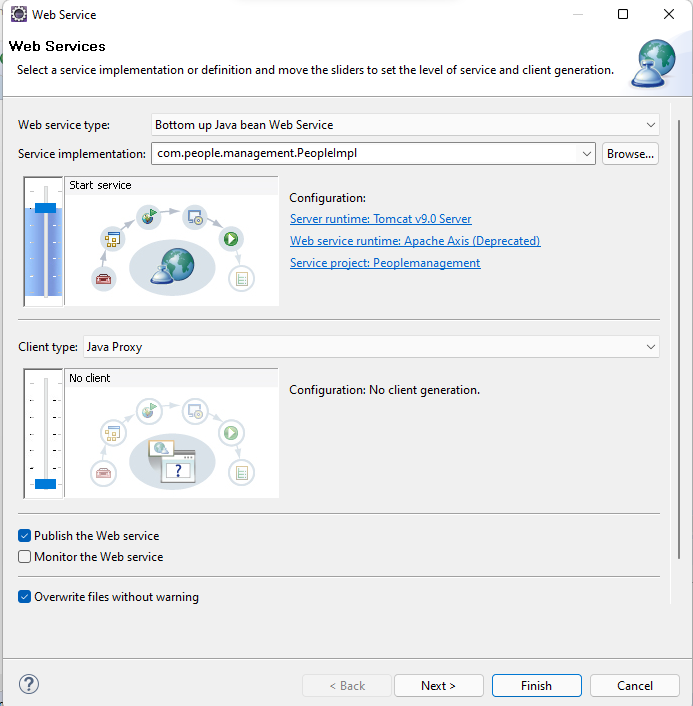
**Write a project name and check the server and runtime configurations**

****

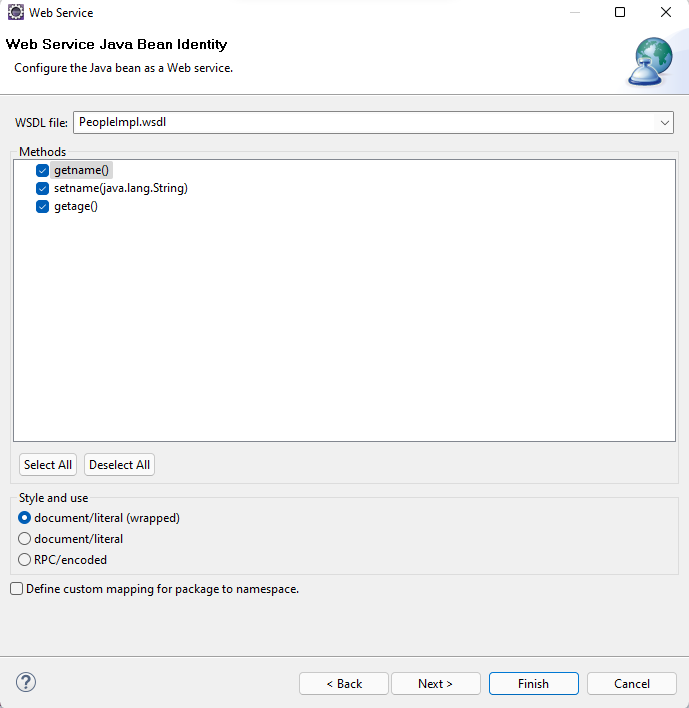
**Click Next**

**Create two files people.java and peopleimpl.java**

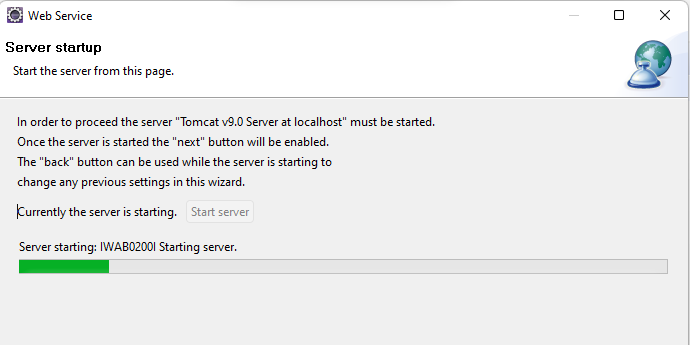
**After entering the content of the files, right click on and click web services**

****

**Click next**

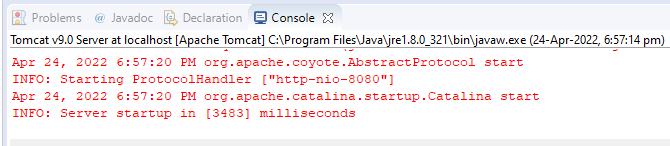
****

**Click next**

****

**Click start server**

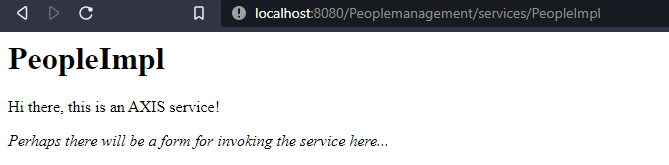
**After server started message is displayed , click finish**

****

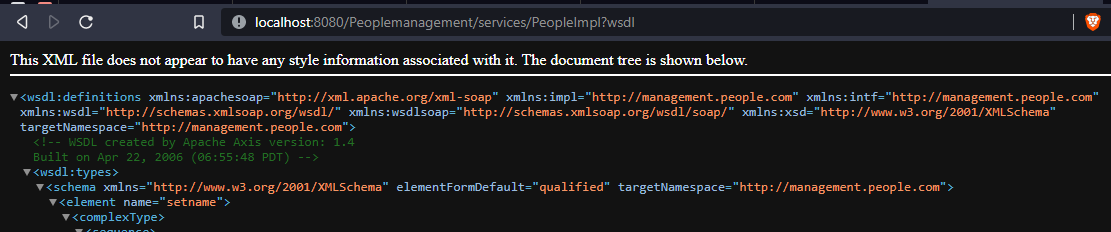
**If the server is started successfully, this above message will appear**

**And a wsdl file will be displayed**

**And web server will host the page in the address**

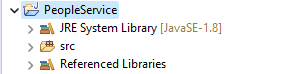
****

**Which can be invoked via a service file, and will be created as a separate project**

****

**This shows the variety of services available**

**On creating new service project, the explorer will display it like this**

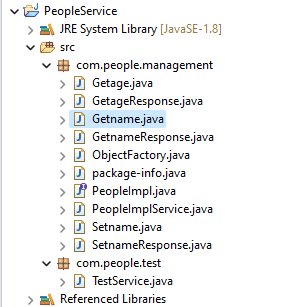
****

**In command prompt , we should type the following command**

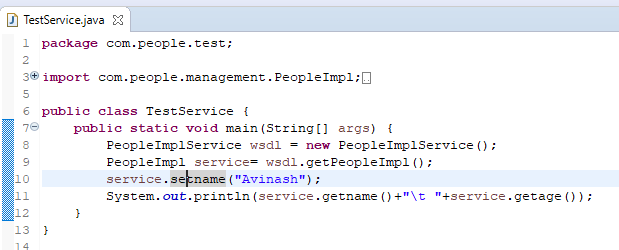
**Wsimport -keep** [**http://localhost:8080/Peoplemanagement/services/PeopleImpl**](http://localhost:8080/Peoplemanagement/services/PeopleImpl)

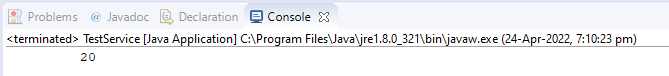
**Which would parse the wsdl file and we have to delete the class files in the directory where we have executed these commands**

**Then we should create a test project with new package**

****

**Here we have used getname and getage methods**

****

****

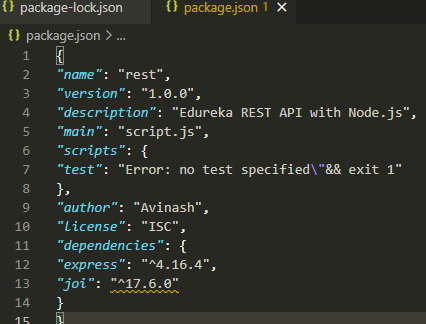
**This is the output of the request**

**REST:**

**First, you need to create your project directory. Next, open the command prompt and navigate to your project directory. Once there, you need to call npm using the below command:**

|  | **npm init**  **Node -i express**  **Node -i joi** |
| --- | --- |

**When you hit enter, Node.js will ask you to enter some details to build the package.json**

****

**Next the server.js file should be written and it should be executed**

const express = require("express");

const Joi = require("Joi");

const app = express();

app.use(express.json());

const books = [

{ title: "Harry Potter", id: 1 },

{ title: "Twilight", id: 2 },

{ title: "Lorien Legacies", id: 3 },

];

//READ Request Handlers

app.get("/", (req, res) => {

res.send("Welcome to Avinash REST API with Node.js ");

});

app.get("/api/books", (req, res) => {

res.send(books);

});

app.get("/api/books/:id", (req, res) => {

const book = books.find((c) => c.id === parseInt(req.params.id));

if (!book)

res

.status(404)

.send(

'<h2 style="font-family: Malgun Gothic; color: darkred;">Ooops... Cant find what you are looking for!</h2>'

);

res.send(book);

});

//CREATE Request Handler

app.post("/api/books", (req, res) => {

const { error } = validateBook(req.body);

const book = {

id: books.length + 1,

title: req.body.title,

};

books.push(book);

res.send(book);

});

//UPDATE Request Handler

app.put("/api/books/:id", (req, res) => {

const book = books.find((c) => c.id === parseInt(req.params.id));

if (!book)

res

.status(404)

.send(

'<h2 style="font-family: Malgun Gothic; color:darkred;">Not Found!! </h2>'

);

const { error } = validateBook(req.body);

if (error) {

res.status(400).send(error.details[0].message);

return;

}

book.title = req.body.title;

res.send(book);

});

//DELETE Request Handler

app.delete("/api/books/:id", (req, res) => {

const book = books.find((c) => c.id === parseInt(req.params.id));

if (!book)

res

.status(404)

.send(

'<h2 style="font-family: Malgun Gothic; color: darkred;"> Not Found!! </h2>'

);

const index = books.indexOf(book);

books.splice(index, 1);

res.send(book);

});

function validateBook(book) {

const schema = {

title: Joi.string().min(3).required(),

};

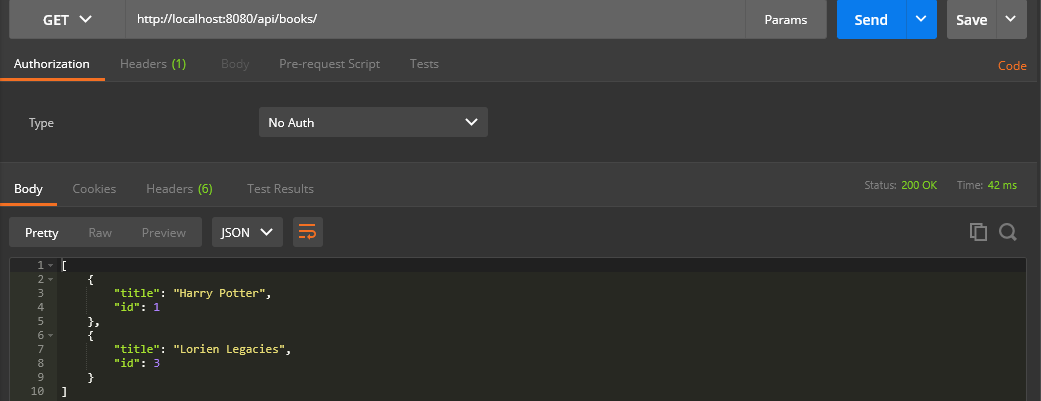
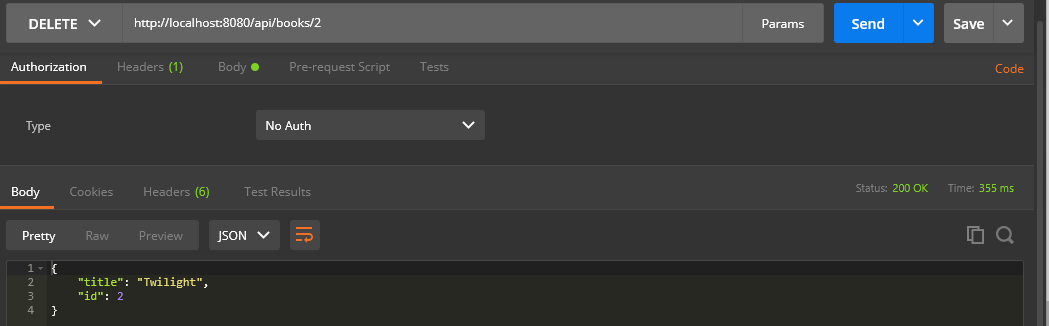
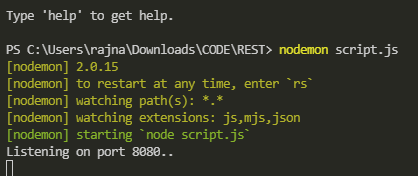
return Joi.validate(book, schema);

}

//PORT ENVIRONMENT VARIABLE

const port = process.env.PORT || 8080;

app.listen(port, () => console.log(`Listening on port ${port}..`));

****

**RESULTS:**

**Thus, the installation or configuration of SOAP and RESTFUL web services are implemented.**