

MODULO 1:

Esercizio 1:

0:00:48 || 12:16

I namespace

- Per utilizzare un namespace, bisogna associargli un identificativo che poi viene utilizzato nei tag che si riferiscono a quel namespace

```
<?xml version="1.0" encoding="utf-8"?>
<r:rubrica xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:r="rubricaNamespace"
  xmlns:u="ufficioNamespace">
  <r:nome>Mario</r:nome>
  <r:cognome>Rossi</r:cognome>
  <r:indirizzo>Corso Italia, 30</r:indirizzo>
  <r:citta>Bari</r:citta>
  <r:lingua>it</r:lingua>
  <u:ufficio>
    <u:telefono>0803382070</u:telefono>
    <u:indirizzo>Via A. Olivetti, 11</u:indirizzo>
    <u:citta>Molfetta</u:citta>
    <u:azienda>Exprivia S.p.A.</u:azienda>
  </u:ufficio>
</r:rubrica>
```

rubrica_ufficio (XML):

```
<?xml version="1.0" encoding="utf-8"?>
<r:rubrica xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:r="rubricaNamespace"
  xmlns:u="ufficioNamespace">
  <r:nome> Mario </r:nome>
  <r:cognome> Rossi </r:cognome>
  <r:indirizzo> Corso Italia, 30 </r:indirizzo>
  <r:citta> Bari </r:citta>
  <r:lingua> it </r:lingua>
```

```
<u:ufficio>
  <u:telefono> 0803382070 </u:telefono>
  <u:indirizzo> Via A. Olivetti, 11 </u:indirizzo>
  <u:citta> Molfetta </u:citta>
  <u:azienda> Exprivia S.p.A. </u:azienda>
</u:ufficio>
</r:rubrica>
```

rubricaNamespace (XSD 1):

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="rubricaNamespace">
  <xs:element name="rubrica" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nome" type="xs:string"/>
        <xs:element name="cognome" type="xs:string"/>
        <xs:element name="indirizzo" type="xs:string"/>
        <xs:element name="citta" type="xs:string"/>
        <xs:element name="lingua" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

ufficioNamespace (XSD 2):

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="ufficioNamespace">
  <xs:element name="ufficio" maxOccurs="unbounded">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="telefono" type="xs:string"/>
        <xs:element name="indirizzo" type="xs:string"/>
        <xs:element name="citta" type="xs:string"/>
        <xs:element name="azienda" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Esercizio 2a:

Esercizi

- Tradurre in JSON uno dei file XML utilizzati come esercizio in precedenza e validare il risultato
- Modificare il programma della richiesta AJAX per:
 - Contare quante feste ci sono negli Stati Uniti (con date diverse). Assumere che le feste siano già ordinate per data.
 - trovare la data del Labor Day
 - dato un mese, trovare le feste di quel mese

prodotti (XML):

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<prodotti>
```

```
  <prodotto>
```

```
    <nome>Crackers</nome>
```

```
    <prezzo>6,49</prezzo>
```

```
    <marca>Galbusera</marca>
```

```
    <ingredienti>
```

```
      <ingrediente>farina di frumento</ingrediente>
```

```
      <ingrediente>olio di oliva</ingrediente>
```

```
      <ingrediente>sale</ingrediente>
```

```
      <!-- altro -->
```

```
    </ingredienti>
```

```
    <codice>1234</codice>
```

```
  </prodotto>
```

```
</prodotti>
```

```
<nome>Mele gialle</nome>

<prezzo>2,10</prezzo>

<marca>Melinda</marca>

<ingredienti/>

<codice>465462</codice>

</prodotto>

</prodotti>
```

prodotti (JSON):

```
{
  "prodotti":
  [
    {
      "nome": "Crackers",
      "prezzo": 6.49,
      "marca": "Galbusera",
      "ingredienti":
      [
        "farina di frumento", "olio di oliva", "sale"
      ],
      "codice": "1234"
    },
    {
      "nome": "Mele gialle",
      "prezzo": 2.10,
      "marca": "Melinda",
      "ingredienti": [],
    }
  ]
}
```

```
        "codice": "465462"
    }
]
}
```

utenti_prodotti_v1 (XML):

```
<?xml version="1.0" encoding="UTF-8"?>
<sito>
  <utenti>
    <utente>
      <nome>Giusepep</nome>
      <cognome>Maenza</cognome>
      <data_di_nascita>01/01/2001</data_di_nascita>
      <residenza>Bitonto</residenza>
      <ruoli>
        <ruolo>Amministratore</ruolo>
      </ruoli>
      <prodotti>
        <prodotto_acquistato>
          <nome>Crakers</nome>
          <prezzo>6.49</prezzo>
          <marca>Galbusera</marca>
          <ingredienti>
            <ingrediente>farina di frumento</ingrediente>
            <ingrediente>olio di uliva</ingrediente>
          </ingredienti>
          <codice>123</codice>
        </prodotto_acquistato>
      </prodotti>
    </utente>
  </utenti>
</sito>
```

```
        </prodotto_acquistato>
    </prodotti>
</utente>

<utente>
    <nome>Davide</nome>
    <cognome>Ricci</cognome>
    <data_di_nascita>02/02/2002</data_di_nascita>
    <residenza>Molfetta</residenza>
    <ruoli>
        <ruolo>Moderatore</ruolo>
    </ruoli>
    <prodotti>
        <prodotto_acquistato>
            <nome>mele Verdi</nome>
            <prezzo>2.49</prezzo>
            <marca>Melindaelinda</marca>
            <ingredienti>
                <ingrediente></ingrediente>
            </ingredienti>
            <codice>124</codice>
        </prodotto_acquistato>
    </prodotti>
</utente>
</utenti>
</sito>
```

utenti_prodotti_v1 (JSON):

```
<?xml version="1.0" encoding="UTF-8"?>

<sito>

  <utenti>

    <utente>

      <nome>Giusepep</nome>

      <cognome>Maenza</cognome>

      <data_di_nascita>01/01/2001</data_di_nascita>

      <residenza>Bitonto</residenza>

      <ruoli>

        <ruolo>Amministratore</ruolo>

      </ruoli>

      <prodotti>

        <prodotto_acquistato>

          <nome>Crakers</nome>

          <prezzo>6.49</prezzo>

          <marca>Galbusera</marca>

          <ingredienti>

            <ingrediente>farina di frumento</ingrediente>

            <ingrediente>olio di uliva</ingrediente>

          </ingredienti>

          <codice>123</codice>

        </prodotto_acquistato>

      </prodotti>

    </utente>

    <utente>

      <nome>Davide</nome>

      <cognome>Ricci</cognome>
```



```
<data_di_nascita>02/02/2002</data_di_nascita>

<residenza>Molfetta</residenza>

<ruoli>
  <ruolo>Moderatore</ruolo>
</ruoli>

<prodotti>
  <prodotto_acquistato>
    <nome>mele Verdi</nome>
    <prezzo>2.49</prezzo>
    <marca>Melindaelinda</marca>
    <ingredienti>
      <ingrediente></ingrediente>
    </ingredienti>
    <codice>124</codice>
  </prodotto_acquistato>
</prodotti>

</utente>

</utenti>

</sito>
```

Esercizio 2b:

Esercizi

- Tradurre in JSON uno dei file XML utilizzati come esercizio in precedenza e validare il risultato
- Modificare il programma della richiesta AJAX per:
 - contare quante feste ci sono negli Stati Uniti (con date diverse). Assumere che le feste siano già ordinate per data.
 - trovare la data del Labor Day
 - dato un mese, trovare le feste di quel mese

cercaLaborDay (HTML 1):

```
<html>

<head>

  <script type="text/javascript">

    function cercaLaborDay(lista)

    {

      for (var i = 0; i < lista.length; i++)

      {

        if (lista[i].localName === "Labor Day")

        {

          return lista[i].date;

        }

      }

      return "Labor Day non trovato nella lista delle festività.";

    }

  </script>

</head>

</html>
```

```
var httpRequest = new XMLHttpRequest();
```

```
httpRequest.addEventListener('load', function()
```

```
{
```

```
    if (httpRequest.status === 200)
```

```
    {
```

```
        var listaFeste = JSON.parse(httpRequest.responseText);
```

```
        var dataLaborDay = cercaLaborDay(listaFeste);
```

```
        console.log("Data del Labor Day:", dataLaborDay);
```

```
    }
```

```
    else
```

```
    {
```

```
        console.error("Errore nella richiesta HTTP.");
```

```
    }
```

```
});
```

```
httpRequest.open('GET', 'https://date.nager.at/api/v2/publicholidays/2023/US');
```

```
console.log("Invio la richiesta");
```

```
httpRequest.send();
```

```
console.log("Richiesta inviata");
```

```
</script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

conteggioFeste (HTML 2):

```
<html>
```

```
<head>
```

```
<script type="text/javascript">
```

```
    //funzione che conta le feste con data diversa nella lista ricevuta ordinata per data
```

```
    //restituisce il conteggio delle feste con data diversa
```

```
    function contaFeste(lista){
```

```
        var dataPrecedente = null;
```

```
        var conteggio = 0;
```

```
        for(var i = 0; i < lista.length; i++){
```

```
            if(lista[i].date != dataPrecedente){
```

```
                conteggio++;
```

```
            }
```

```
            dataPrecedente = lista[i].date;
```

```
        }
```

```
        return conteggio;
```

```
    }
```

```
    function callback(){
```

```
        console.log(httpRequest.responseText);
```

```
        listaFeste = JSON.parse(httpRequest.responseText);
```

```
        var conteggioFeste = contaFeste(listaFeste);
```

```
        alert(conteggioFeste);
```

```
    }
```

```
    var httpRequest = new XMLHttpRequest();
```

```
    /*
```

```
    httpRequest.addEventListener('load', function(){
```

```
        console.log(httpRequest.responseText);
```

```
        console.log(JSON.parse(httpRequest.responseText));
```

```
    });
```

```

        */
        httpRequest.addEventListener('load', callback);
        httpRequest.open('GET', 'https://date.nager.at/api/v2/publicholidays/2023/US');
        console.log("invio la richiesta");
        httpRequest.send();
        console.log("richiesta inviata");

    </script>
</head>
<body>
</body>
</html>

```

trovaFesteMese (HTML 3):

```

<html>
<head>
    <script type="text/javascript">
        function trovaFesteDelMese(lista, meseDesiderato)
        {
            var festivit DelMese = [];
            for (var i = 0; i < lista.length; i++)
            {
                var dataFesta = new Date(lista[i].date);
                // +1 perch  i mesi sono 0-11
                var meseFesta = dataFesta.getMonth() + 1;
            }
        }
    </script>

```

```
        if (meseFesta === meseDesiderato)
        {
            festivitàDelMese.push(lista[i]);
        }
    }
    return festivitàDelMese;
}
```

```
// Definisci la variabile listaFeste
```

```
var listaFeste;
```

```
var httpRequest = new XMLHttpRequest();
```

```
httpRequest.addEventListener('load', function()
```

```
{
    console.log(httpRequest.responseText);
    listaFeste = JSON.parse(httpRequest.responseText);
```

```
    // Cambia il numero del mese desiderato
```

```
    var meseDesiderato = 9
```

```
    var festivitàDelMeseDesiderato = trovaFesteDelMese(listaFeste, meseDesiderato);
```

```
    console.log(festivitàDelMeseDesiderato);
```

```
});
```

```
httpRequest.open('GET', 'https://date.nager.at/api/v2/publicholidays/2023/US');
```

```
console.log("invio la richiesta");
```

```
httpRequest.send();
```

```
console.log("richiesta inviata");
```

```
</script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

Esercizio 3:

Traccia:

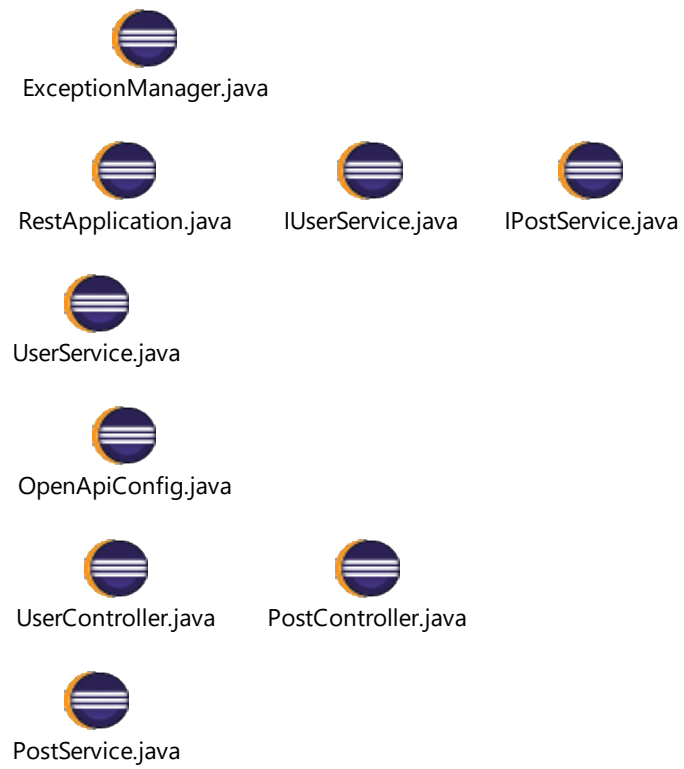
Utilizzando Eclipse IDE:

Implementare e documentare con Swagger questi servizi:

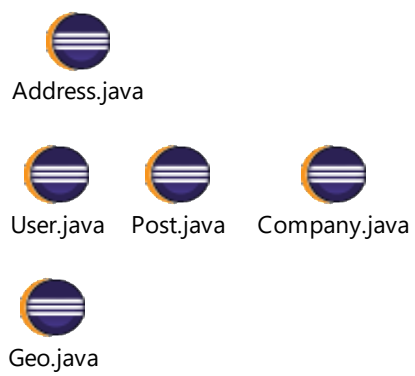
- Modifica di un post (PUT /posts/{id}) sollevando un'eccezione se l'id non è valido
- Cancellazione di un post (DELETE /posts/{id}) sollevando un'eccezione se l'id non è valido
- Leggere l'elenco degli utenti da JSON Placeholder con la classe RestTemplate (come abbiamo fatto per i post)
- Implementare il servizio /posts?userId={userId}

Utilizzare l'annotation @RequestParam <https://www.baeldung.com/spring-request-param>

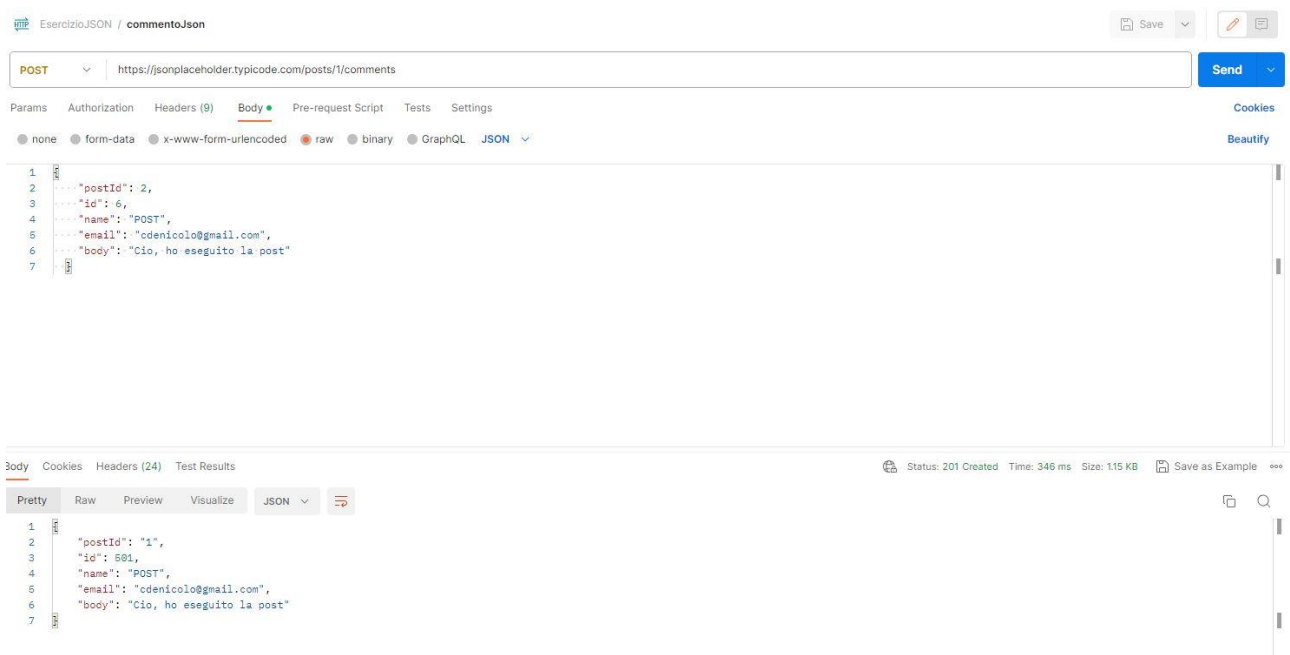
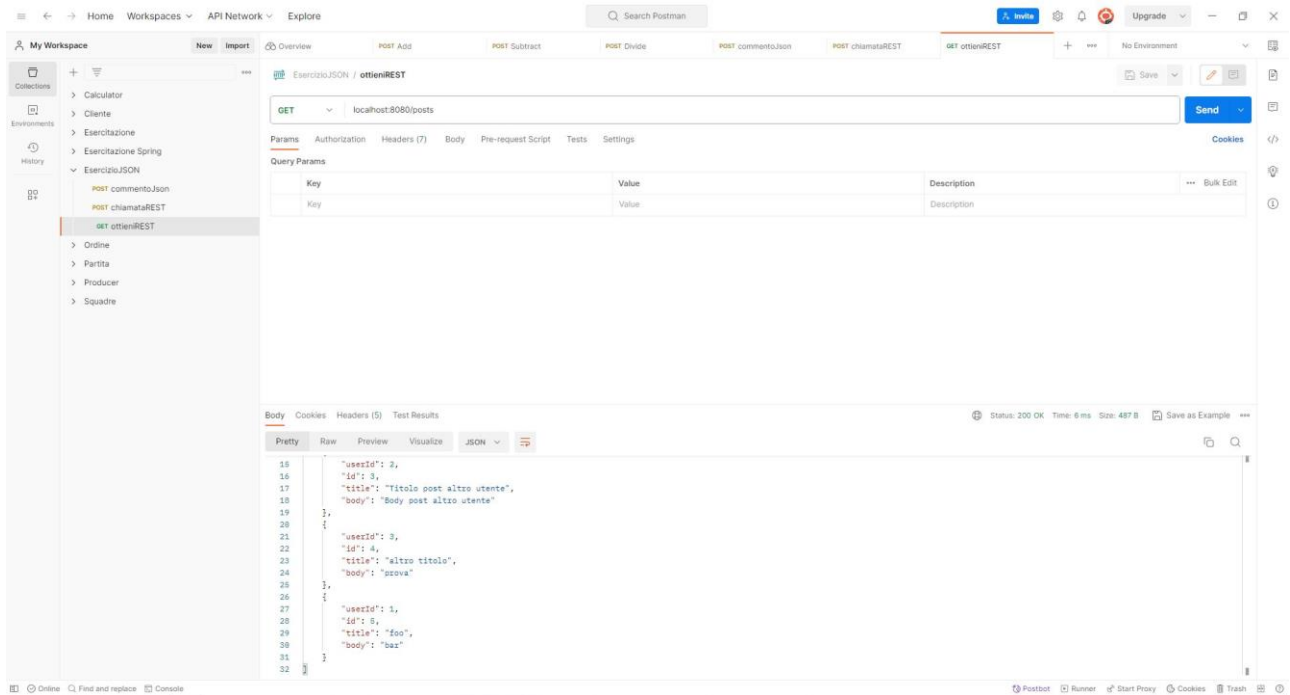
Modificare i metodi di creazione e modifica di un post: sollevare un'eccezione se l'id dell'utente non è valido

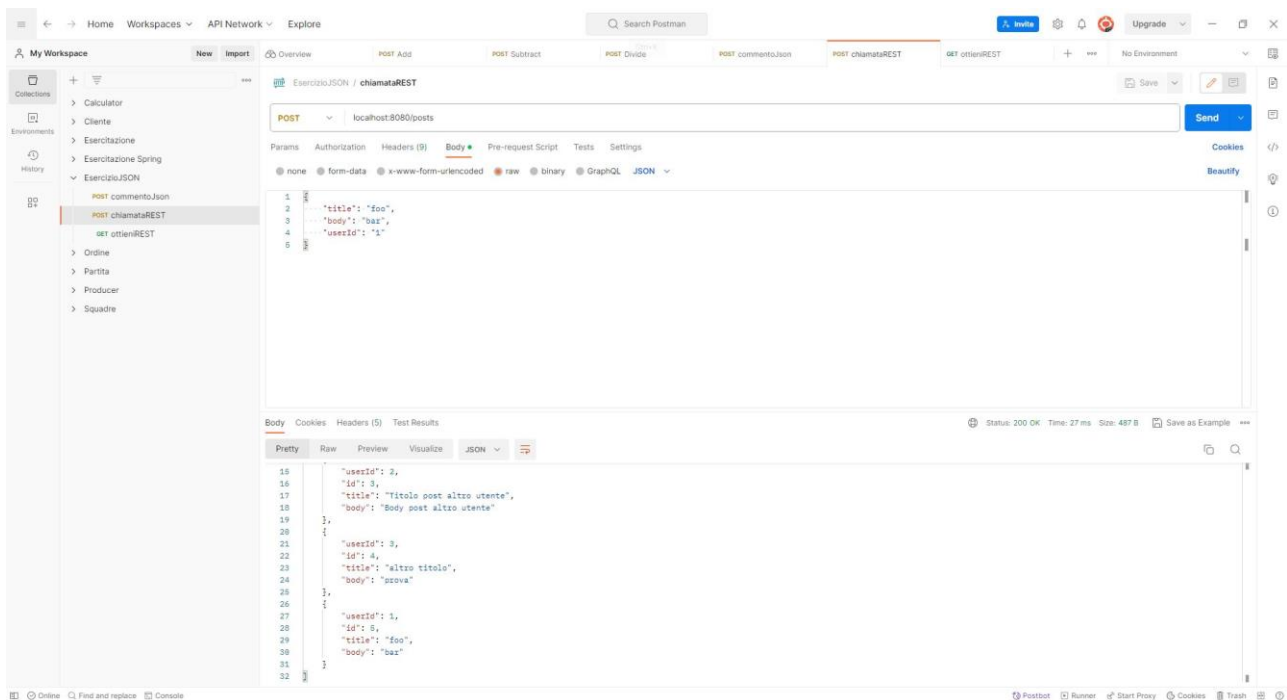


Folder model:



Postman:





MODULO 2:

Esercizio TIBCO con il render XML:

- Creare un XML di nome Classe di tipo ClasseType;
- Creare all'interno di ClasseType una alunno [0, *] di tipo AlunnoType;
- Inserire in AlunnoType un nome (string) e sesso (simple type) con enum M e F;
- (Vedere gli schema su TIBCO per ulteriori dubbi);
- Creare il package (tut03), un process (Tut03) e un subprocess (CreazioneClassi);
- Start -> Output Editor -> classe;
- Creo un activity di "Mapping" -> Input Editor -> classi;
- Dirigente -> Input -> tasto destro -> output su classe;
- Input -> Data Source -> \$start -> alunno* -> drag and drop in alunno* a destra -> finish;

- Trascino in alunno "sesso" e gli associo un = "M";
- Faccio la stessa cosa con l'altro alunno e gli associo un = "F";
- End -> Input Editor -> classi;
- Trascino in tutorial il sub e lo collego col timer;
- Seleziono l'activity della sub -> Input -> duplicare alunno -> inserisci nome e sesso;
- Creare un activity "RenderXML" -> add activity -> XML activities;
- RenderXML -> Input Editor -> classi;
- RenderXML -> general -> validate input (check) -> format using default namespace prefix (check);
- Creo un log e lo collego al RenderXML;
- RenderXML -> Input -> \$Creazione -> classe* -> drag and drop -> classe*;
- Log -> input -> creazione -> \$render -> drag and drop -> message;
- End -> input -> \$dirigente -> classe* -> drag and drop -> classe*;
- RenderXML -> input -> for each (check) -> classe* -> drag and drop -> classe*;
- Delete alunni;
- Cliccare sull'icona in alto a destra per verificare gli errori (rettangolino con il check mark);
- Drag and drop alunno* su alunno* collegando un for-each;
- Nome -> Surround with Choose -> 1 (tasto destro);
- Nel when mettere "tns2:sesso="M"";
- Nome -> concat("signore ", tns2:nome);
- Nome -> concat("signora ", tns2:nome);
- Sul sesso -> Surround with -> If;
- Sull'if -> tns2:sesso = "F";
- Sul sesso solo scrivere "tns2:sesso".

Traccia 1:

Esercizio 01.01: General Activities

Sleep

Creare un processo che dopo 2 secondi dall'avvio scriva un log:

Processo partito alle: 11:13:41

Adesso sono le ore: 11:13:43



- Creare un processo (Mod01);
- Nel processo inserire un timer, un log, uno sleep e un altro log;
- Nel timer: Run Once -> mettere la data del giorno successivo;
- Nel primo log: message -> concat("Processo partito alle: " \$Timer/Time) -> Ho trascinato il Time nel message;
- Nello sleep: IntervallInMillisec: 2000 (2 secondi);
- Nel secondo log: message -> concat("Adesso sono le ore: " format-time(current-time(), "[H01]:[m01]:[s01]"));
- Collego tutte le activities ed eseguo il processo.

Traccia 2:

Esercizio 01.02: General Activities

Transitions

Creare un processo che si avvii 10 volte (ogni 3 secondi) e scriva in base al momento in cui parte, un log (in un sottoprocesso) di questo tipo:

```
Dispari
Pari
Dispari
Pari
Dispari
Pari
Dispari
Pari
Dispari
Pari
```



- Creo un processo (Mod02) e due sottoprocessi (SubPari e SubDispari);
- Mod02 avrà un timer collegato ai due sub;
- SubPari e SubDispari avranno entrambi uno starter, un log e un end;
- Nel SubPari mettere il message "SecondoPari";
- Nel SubDispari mettere il message "SecondoDispari";
- Nel timer: Use local time (check) -> Time Interval: 3 -> Interval Unit: Second -> Occurences: 10.

Traccia 3:

Esercizio 01.03: General Activities

Mapper

Utilizzare i due xsd all'interno di un sottoprocesso per trasformare il formato dati in ingresso (del primo xsd) nel formato del secondo xsd.

Completare l'esercizio scrivendo questo log:

Francesco è nato il 20 marzo 2010

expri^{via}



Esercizio 01.03: General Activities

XSD

Creare un secondo xsd simile al primo.

Le differenze riguarderanno 2 elementi:

compleanno al posto di dataDiNascita: non sarà di tipo String ma di tipo Date

sex: dovrà contenere 2 possibili opzioni: maschio e femmina

expri^{via}

- Creo due XSD: persona e compleanno;
- In persona:
 - Add element -> elenco -> ElencoType;
 - Entro in elenco -> ElencoType -> add element -> persona -> PersonaType -> add multiplicity [0, *];
 - Entro in persona -> PersonaType -> creo nome (string), sesso (simple type e anonimo) con enum M e F -> dataNascita (string) con multiplicity [0, 1];
- In compleanno:
 - Add element -> compleanno -> ElencoType;
 - Entro in compleanno -> ElencoType -> add element -> persona -> PersonaType;

- Entro in persona -> PersonaType -> creo nome (string), sesso (simple type e anonimo) con enum maschio e femmina -> dataNascita (date);
- Credo due Process: dataNascita (timer, sub e log) e ricercaDataDiNascita (start, mapper e end);
- Start -> Output Editor -> persona -> ciclo elenco;
- Mapper -> Input Editor -> compleanno -> ciclo compleanno -> input -> persona* -> drag and drop -> persona -> surround with choose -> 1;
- When -> tns1:sesso="M" -> sesso -> maschio;
- Otherwise -> femmina;
- Compleanno -> tib:parse-date("dd/mm/yyyy", tns1:dataNascita);
- End -> Input Editor -> compleanno -> clicca compleanno -> mapper in input -> persona -> drag and drop (copy-of) -> compleanno;
- RicercaDataDiNascita (icona dell'activity del sub con gli ingranaggi) -> persona -> nome "Francesco" sesso "M" -> dataNascita -> "25/12/1995";
- Log -> concat(\$ricercaDataDiNascita/tns2:persona/tns2:nome, " è nato il ", format-date(\$ricercaDataDiNascita/tns2: persona/tns2: compleanno, "[D01]/[M01]/[Y0001]"))
- Persona -> nome -> drag and drop (per scrivere tns2:nome automaticamente);
- Persona -> compleanno -> drag and drop (per scrivere tns2:compleanno automaticamente);
- Constants -> date/time -> [M01]/[D01]/[Y0001] at [H01]:[N001]:[S01] (Ovviamente modificare questa istruzione come quella scritta sopra);
- Output: Francesco è nato il 25/01/1995.

Traccia 5:

Variables

Utilizzo

- Immaginiamo che una classe di alunni voglia fare una colletta per comprare una corona di fiori per il funerale del professore.
- Potremmo creare una struttura di questo tipo:

```
<alunni>
  <alunno>
    <nome>Gino</nome>
    <denaro>5</denaro>
  </alunno>
  <alunno>
    <nome>Pino</nome>
    <denaro>5</denaro>
  </alunno>
  <alunno>
    <nome>Nina</nome>
    <denaro>7</denaro>
  </alunno>
</alunni>
```

Il log:

Sono stati raccolti 5€

Sono stati raccolti 10€

Sono stati raccolti 17€

- Proviamo a ricavare la somma raccolta con le conoscenze che abbiamo fino ad ora

- Creo un xsd chiamato alunniPortafoglio;
- Alunni -> AlunniType -> alunno (AlunnoType) con molteplicità [0, *];
- Nome e denaro;
- Creo un processo con timer, mapper e log (iterator);
- Nel mapper: Input Editor -> alunni;
- In input: duplicate alunno e mettiamo i dati del tag;
- Process -> Tut01 -> empty process -> timer;
- Selezionare lo spazio bianco -> process variables -> ABC+ (variabile semplice) -> EL+ (variabile complessa) -> utilizzo una variabile complessa -> alunni -> ci crea una variabile e la chiamiamo "alunni";
- Creo una variabile semplice e la chiamo somma (Integer) con default while 0;
- Creo una variabile Assign e la collego al timer;
- Seleziono alunni come variabile in general;
- Vado in input, duplico gli alunni e li inserisco;
- Creo un altro assign e lo rinonimo "denaroRaccolto";
- Inserisco un log, vado su input e inserisco il messaggio: concat("Sono stati raccolti ", xsd:string(\$somma), "€");
- Collego tutte le activities;
- Vado nel secondo assign (denaroRaccolto) -> create group -> iterate;
- Iterate -> variable list -> drag and drop -> alunno -> elimina il filter;
- Rinonimo l'iteration element come currentAlunno;
- Nel secondo assign, associo set variable somma;
- Nell'input, aggiungo (drag and drop) \$somma (che è uguale a 0) + il denaro di un alunno a caso e mi uscirà: "\$somma + \$currentAlunno/tns1:denaro".

Traccia 6:

Esercizio 02.01: Properties/Variables

XSD

Creare un xsd di un xml di questo tipo:

```
<tns:autodromo xmlns:tns="http://www.example.org/Es01_Auto">
  <tns:auto>
    <tns:marca>FIAT</tns:marca>
    <tns:modello>500</tns:modello>
    <tns:velocità>130</tns:velocità>
  </tns:auto>
  <tns:auto>
    <tns:marca>FIAT</tns:marca>
    <tns:modello>500x</tns:modello>
    <tns:velocità>180</tns:velocità>
  </tns:auto>
  <tns:auto>
    <tns:marca>Opel</tns:marca>
    <tns:modello>Zafira</tns:modello>
    <tns:velocità>170</tns:velocità>
  </tns:auto>
  ...
  <tns:auto>
    <tns:marca>Opel</tns:marca>
    <tns:modello>Agila</tns:modello>
    <tns:velocità>130</tns:velocità>
  </tns:auto>
  <tns:auto>
    <tns:marca>FIAT</tns:marca>
    <tns:modello>Panda</tns:modello>
    <tns:velocità>120</tns:velocità>
  </tns:auto>
</tns:autodromo>
```

expri^{ia}

Esercizio 02.01: Properties/Variables

Variable

Creare un sottoprocesso Autodromo che raccolga le informazioni delle 4 auto precedenti e sia in grado di ricevere dall'esterno le informazioni di una nuova auto e aggiungerle all'autodromo esistente.

Infrine loggare le auto e la loro velocità:

```
auto: FIAT 500 - velocità: 130
auto: FIAT 500x - velocità: 180
auto: Opel Zafira - velocità: 170
auto: Opel Agila - velocità: 130
auto: FIAT Panda - velocità: 120
```



expri^{ia}

- Creo un xsd chiamato Autodromo di tipo AutodromoType;
- All'interno di AutodromoType, creo un auto [0, *] di tipo AutoType;
- AutoType -> marca (string) -> modello (string) -> velocità (int);
- Creo un processo (Mod06) e un sottoprocesso (Sub06);
- Nel Sub06:
- Start -> Output Editor -> autodromo;
- Assign -> inserisco i dati -> duplicate -> auto -> drag and drop;
- End -> input editor -> autodromo -> input -> \$variable -> auto -> drag and drop;
- Nell'assign, inserisco l'autodromo nel "process variable set" andando nel general;
- Nel Mod06:

- Creo un activity sub con drag and drop che ci inserisca i dati dell'ultima auto;
- Creo un log con un iterate;
- In iterate: general -> variable list -> \$Sub06/tns1:auto -> iteration element -> currentAuto;
- Infine nel log, scrivo il seguente message: concat("auto: ", \$currentAuto/tns:marca, " ", \$currentAuto/tns:modello, "- velocità: ", xsd:string(\$currentAuto/tns:velocità)).

Traccia 8:

Esercizio 02.03: Properties/Variables

Properties

Inserire nell'autodromo del sottoprocesso precedente una nuova auto (Audi A2) con dati provenienti da 3 Module properties.

Scrivere per ciascuna marca di auto dell'autodromo, tutti i modelli all'interno di 3 file xml:

FIAT.xml

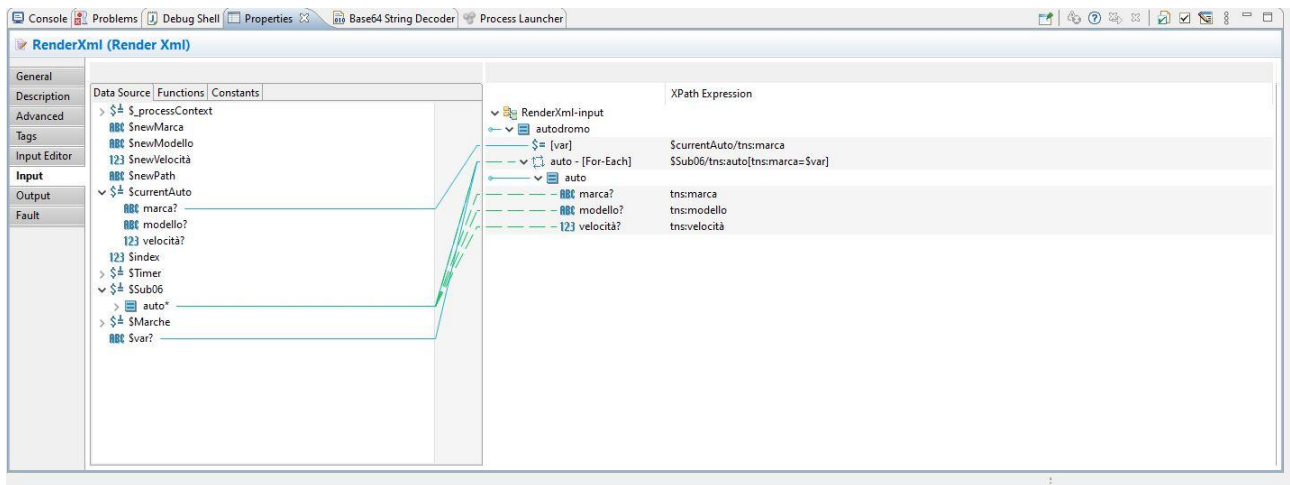
Opel.xml

Audi.xml

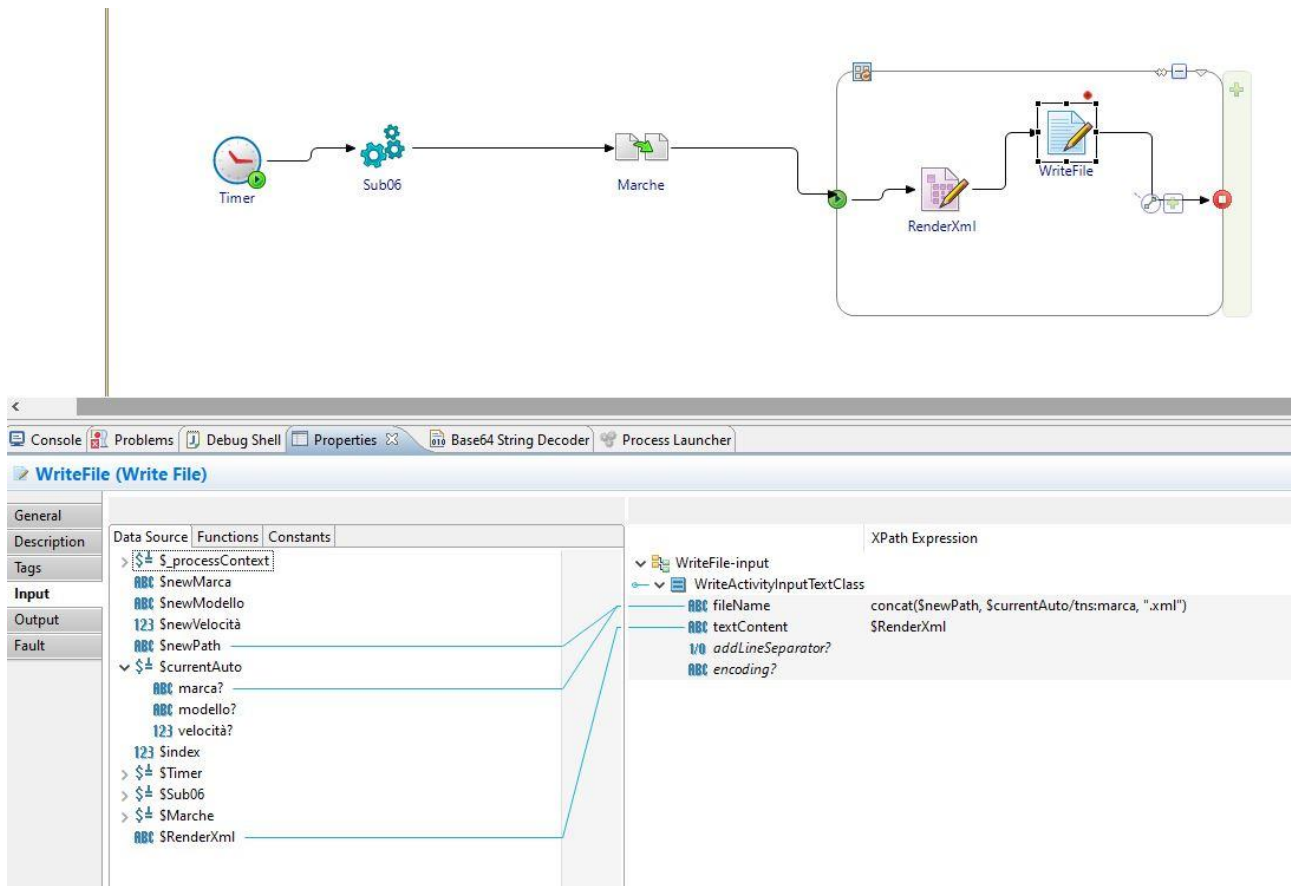


- Riprendere l'esercizio 6 (schema e processi);
- Module descriptors -> module properties;
- New Group -> alunno -> newAuto -> newProperty -> marca (string, Audi), modello (string, A2) e velocità (Integer, 260);
- Andare su process e inserire con il + -> newMarca (string) -> newModello (string) -> newVelocità (Integer)
- Andare sul sub e drag-droppare \$newMarca, \$newModello e \$newVelocità nei rispettivi marca, modello e velocità;
- Creare un mapper e chiamarlo Marche -> input -> auto* -> drag and drop con for each -> tasto destro e Surround with ForEach Group -> auto* -> drag and drop su forEach Group e uscirà così: \$Sub06/tns:auto;
- Marca -> drag and drop alla destra dell'icona del GroupBy;
- Marca -> drag and drop alla destra dell'icona di marca?

- \$Sub06 -> auto* -> drag and drop su auto con forEach;
- Fixare tutto con il comando in alto a destra in modo che esca così:



- Crea un log e in input -> drag and drop \$RenderXML nel message;
- Crea un iterate sul RenderXML e sul Log -> Variable List -> \$Marche -> auto* drag and drop togliendo il filter;
- Iteration Element -> currentAuto;
- Cancellare il log e mettere al suo posto un WriteFile;
- Ritornare sul module properties -> New Group -> filePath -> path (string, inserisci la directory di una cartella creata a tuo piacimento);
- Riandare su process e inserire con il + -> newPath (string) con l'opportuno path in Default Value;
- L'input del WriteFile dovrà avere questi dati:



- WriteFile -> general -> create Non-Existing Directories (check);
- Infine, dopo aver eseguito il programma mi apparirà del contenuto (3 XML relative a delle auto) all'interno del folder creato in precedenza.

Traccia 9:

Esercizio 02.04: Properties/Variables

Job Shared Variable

Sviluppare due processi concorrenti che chiamano entrambi un sottoprocesso Add.
Il sottoprocesso Add dovrà aggiungere 1 al valore di una JobSharedVariable ogni volta che verrà chiamato (il valore iniziale della JSV sarà 0)

I due processi starter dovranno:

Es04A: Cicla per 3 volte e ogni 2 secondi la chiamata del sottoprocesso

Es04B: Cicla per 3 volte e ogni 2 secondi la chiamata del sottoprocesso ma partirà 1 secondo dopo il primo

expri^{via}

12:58 | pok-raxa-fyr

Cerca

- Module Descriptors -> shared variables -> job shared variables -> la chiamo contatore;
- Properties -> Initial Value -> Build Value -> 0 (funge da inizializzazione);
- Credo due processi (Es09A e Es09B) e un sottoprocesso (Add);
- In Add, oltre allo start e all'end, credo GetSharedVariable e SetSharedVariable rispettivamente;
- Nel Get, inserisco il contatore nel general;
- Nel Set, inserisco il contatore nel general -> input -> \$GetSharedVariable -> drag and drop +1 per incrementare: \$GetSharedVariable + 1;
- In Es09A metto il timer -> uno sleep di 2000 ms -> il sub Add -> un GetSharedVariable con il contatore in general -> log con cui farò un drag and drop di \$GetSharedVariable in message;
- Circondare tutti (tranne il timer) con un forEach che cicli per 3 secondi, e quindi Start Counter: 1 e Final Counter: 3;
- In Es09B fare lo stesso procedimento di Es09B con la differenza che bisogna mettere uno sleep dopo il timer e prima dello sleep (e quindi prima del processo di iterazione).

Traccia 10:

Esercizio 03.01: Files

Read & Write file

Leggi un file, scrivi il suo contenuto in un log e archivia nella stessa cartella il suo backup (stesso nome file ma con estensione .bck)

SPECIFICHE

- 1) La property del filePath deve comprendere path, nome file ed estensione
- 2) Dobbiamo inserire una seconda property che permetta di decidere quale possa essere l'estensione del backup. Nel nostro caso sarà .bck
- 3) Se il file da backuppare sarà:
C:\pippo.txt
Il file backuppato dovrà essere
C:\pippo.bck



- Module Descriptors -> Module Properties -> New Group -> File;
- Creo due properties: Path(string + directory) e Estensione(string + .bck);
- Process Properties -> newPath (string + directory) -> newEstensione(string + directory);
- Creo un processo (Es10) con Timer, ReadFile, Log e WriteFile;
- Nell'input del ReadFile trascino il \$newPath in fileName;
- Nell'input del Log scrivo nel message: concat("Contenuto del file: ", \$ReadFile/fileContent/textContent);
- Nell'input del WriteFile scrivo nel fileName: concat(tib:substring-before-last(\$newPath, "."), \$newEstensione) e nel textContext: \$ReadFile/fileContent/textContent;
- Infine, eseguo il programma.

Traccia 11:

Esercizio 03.02: Files

ParseXml

Andiamo a cercare nei file xml delle marche auto creati negli esercizi precedenti, le informazioni sulla Fiat 500 e logghiamo la velocità.













auto: Fiat 500 – velocità: 130



- Module Descriptors -> Module Properties -> New Group -> Autodromo;
- Creo una property: Path(string + directory con *.xml tipo così:
C:\Users\utente\Desktop\IGNACIO*.xml);
- Creo un processo (Es11) con Timer, ListFiles, ReadFile, ParseXml e Log dopo aver creato il newPath in Process Properties e inseriscilo nel fileName di ListFiles;
- Gli ultimi 4 li cirondo con un iterate e creo un iterate annidato sul log;
- Vado sull'input del ReadFile e trascino \$currentAuto -> fullName nell'icona del fileName;
- ParseXml -> Output Editor -> Autodromo -> Input -> trascino il fileContent che trovo nel \$ReadFile nell'icona dell'xmlString;
- Nel primo iterate che chiamerò currentAuto scriverò questa Variable List: \$ListFiles/files/fileInfo;
- Nel secondo iterate che chiamerò currentAuto2 scriverò questa Variable List:
\$ParseXml/ns8:auto[ns8:marca="FIAT" and ns8:modello="500"];
- Nell'input del message scriverò la seguente istruzione: concat("&cr;", "Marca: ",
\$currentAuto2/tns4:marca, "&cr;", "Modello: ", \$currentAuto2/tns4:modello, "&cr;", "Velocità: ",
xsd:string(\$currentAuto2/tns4:velocità));
- Infine, eseguo il programma.

Collegamento al database:

- Creo un database db_anagrafiche e creo una table anagrafica con questi attributi:

Columns							
	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
 	id	integer v			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('an
 	nome	character varying v			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 	cognome	character varying v			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 	citta_di_nascita	character varying v			<input type="checkbox"/>	<input type="checkbox"/>	
 	data_di_nascita	date v			<input type="checkbox"/>	<input type="checkbox"/>	
 	telefono	character varying v			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

- Creo un module (Tutorial04_JDBC);
- Creo un package (tut04) e un process (Tut04);
- Ci metto un timer e un JDBCQuery;
- Vado su general -> Shared Resource -> jdbcProperty -> lente di ingrandimento -> tutorial04_jdbc.JDBCConnectionResource;
- Resources -> tutorial04_jdbc -> JDBCConnectionResource;
- Module Properties -> newGroup: JDBCConnection e 4 newProperties;
- Le seguenti properties: username (string, andare sull'icona dell'elefante su postgresql e andare su connection per vedere l'username) -> password (password, inserisci la password) -> url (string, jdbc:tibcosoftwareinc:postgresql://localhost:5432;DatabaseName=db_anagrafiche, ovviamente mettendo l'host e il nome del db giuste) -> driver (string, tibcosoftwareinc.jdbc.postgresql.PostgreSQLDriver) con cui indico il DBMS che utilizzo;
- Ritorno su JDBCConnectionResource e inserisco i module -> username, password, drive e url;
- Faccio Test Connection.
- JDBCQuery -> Statement -> copio e incollo la seguente query: select id, nome, cognome, citta_di_nascita, data_di_nascita, telefono FROM anagrafica;
- Effettuo il fetch, vado nell'output e vedo se i campi della query mi sono usciti;
- Eseguo la mia query sperando esca un succeed;
- Eseguo il programma;
- Module Descriptors -> Overview -> Activator Process -> activator, Activator;
- Statement -> Inserisco nella query un WHERE tipo questo: where id = ?;
- Creo un parameter (+) id di tipo integer;
- Vado nell'input del JDBCQuery e inserisco nell'id un id già esistente nella table anagrafica con l'insert, per esempio: insert into anagrafica (id, nome, cognome, citta_di_nascita, data_di_nascita, telefono) values(1, 'Pino', 'Giordano', 'Milano','25/04/1978', '2637283987');

- Nello statement inserisco anche nome e città nel where e nei parameters (where id = ? and nome = ? and citta_di_nascita = ?);
- Eseguo il programma;
- Esempio completo di Insert: INSERT INTO anagrafica (Nome, Cognome, Citta_di_nascita, Data_di_nascita, Telefono) VALUES ('Mario', 'Rossi', 'Roma', '1990-05-15', '+39 123-456-7890'), ('Laura', 'Bianchi', 'Milano', '1985-08-25', '+39 987-654-3210'), ('Giovanni', 'Verdi', 'Napoli', '1992-03-10', '+39 345-678-9012'), ('Anna', 'Esposito', 'Torino', '1988-11-03', '+39 567-890-1234'), ('Luca', 'Ricci', 'Firenze', '1995-07-20', '+39 789-012-3456');
- Creo un database auto e restorare col tasto destro il filename con la directory in cui si trova il file auto.

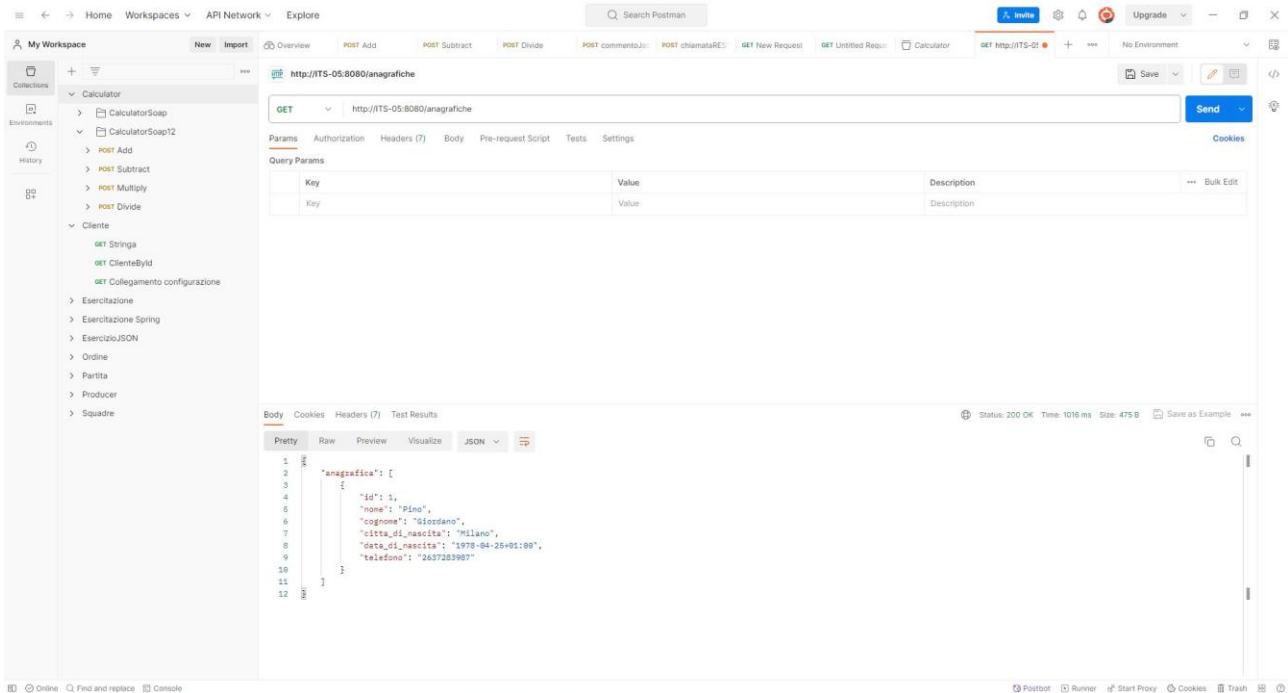
Collegamento al database con un WriteFile:

- Creare un processo (Es13) con Timer, JDBCQuery, log (iterate) e WriteFile (iterate);
- Seguire i passaggi sul collegamento al database;
- Creare un iterate, un log e un WriteFile;
- Nell'iterate mettere in variable list: "\$JDBCQuery/Record" con un iteration element: "currentQuery";
- Nel log, in message: concat("Box: ", xsd:string(\$currentQuery/box), " - Marca: ", \$currentQuery/marca, " - Modello: ", \$currentQuery/modello, " - Velocità: ", xsd:string(\$currentQuery/velocita));
- Nel WriteFile, in fileName: "C:\Users\utente\Desktop\EsempioDB\FiatCars.txt" (non necessariamente questa directory), nel textContent: concat("Box: ", xsd:string(\$currentQuery/box), " || ", " Marca: ", \$currentQuery/marca, " || ", "Modello: ", \$currentQuery/modello, " || ", "Velocità: ", xsd:string(\$currentQuery/velocita), "&cr;").

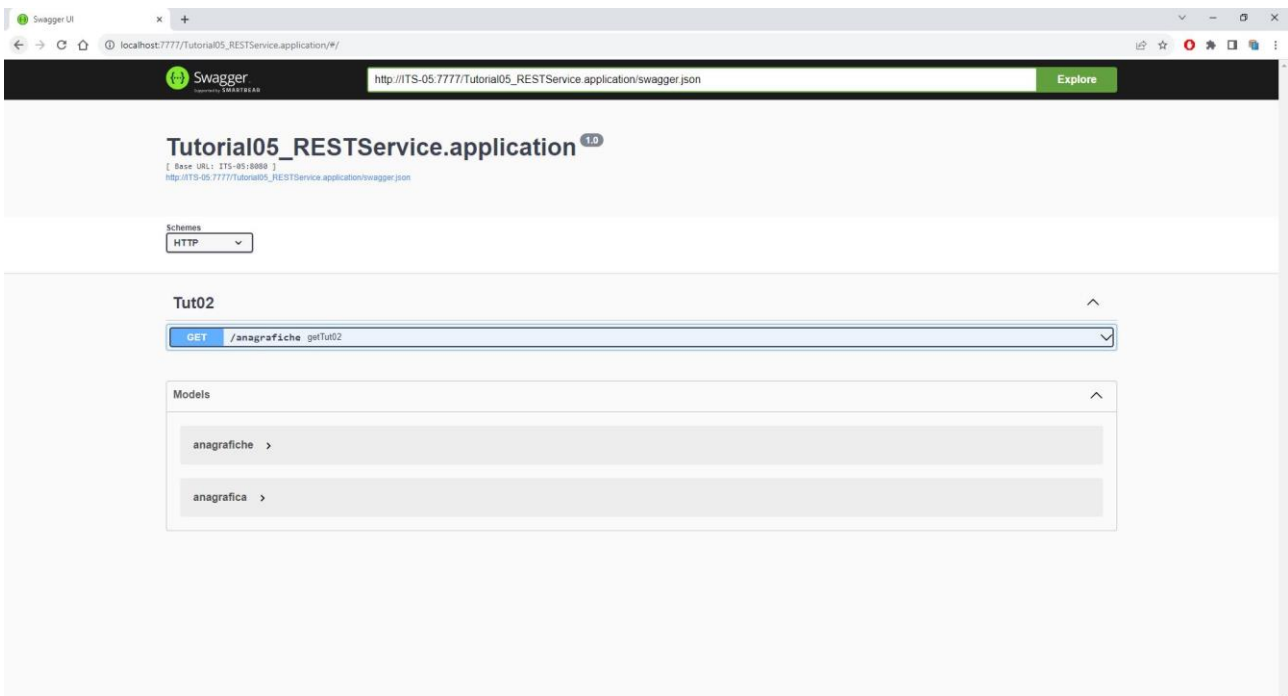
Esempio di REST:

- New -> Business Work REST Resource -> Tut02 -> Resource Service Path: /anagrafiche -> Operations: GET and uncheckare POST -> Resource Definition: Browse: inserimento dell'xsd -> next -> finish;
- Trascinare lo shared01 nella pagina REST e collegarlo al getOut;
- Nel getOut andare in input e trascinare anagrafica* in anagrafica;
- Eseguire il programma;
- Premere invio e scrivere "lendpoints";

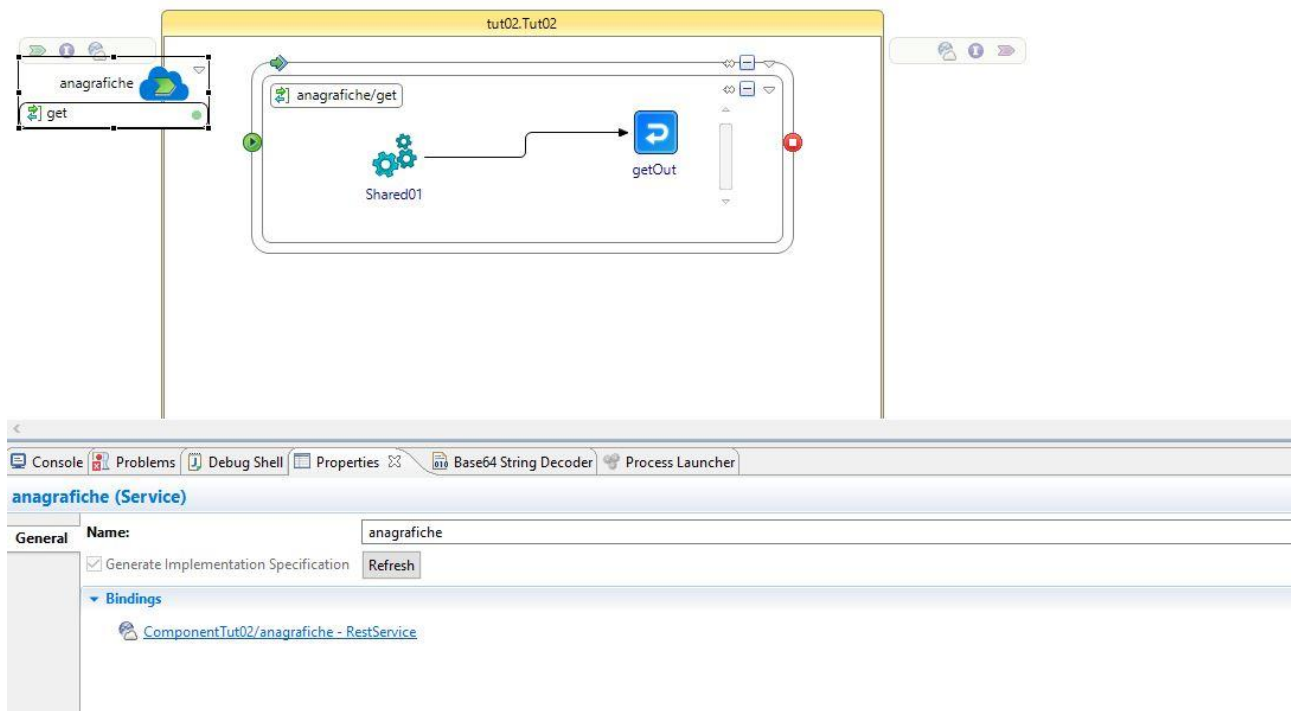
- Postman:



- Swagger tramite url browser (http://localhost:7777/Tutorial05_RESTService.application):



- Cliccare sull'icona a sinistra:



- Andare su bindings per effettuare le varie operazioni di request, response, etc...

Insert, update e delete:

- Usare un JDBCUpdate al posto di JDBCQuery con la struttura dello statement corretto (insert, update e delete).

