**Definizione di REST**

**REST (Representational State Transfer)** è uno stile architetturale per progettare API basate su protocolli web, in particolare HTTP. È stato ideato per facilitare la comunicazione tra client e server sfruttando operazioni standard del protocollo HTTP, come:

* **GET**: Recuperare risorse.
* **POST**: Creare nuove risorse.
* **PUT**: Aggiornare risorse esistenti.
* **DELETE**: Eliminare risorse.

I dati scambiati tra client e server sono tipicamente in formato **JSON** o **XML**, con JSON spesso preferito per la sua leggerezza e semplicità.

**Domande e Risposte (Argomenti da Esame)**

**1. Che cosa rappresenta REST?**

**Domanda:** Cos'è REST?  
**Risposta:** REST è uno stile architetturale utilizzato per costruire API che consentono la comunicazione tra client e server attraverso operazioni HTTP standard.

**2. Quali formati di dati sono utilizzati in REST?**

**Domanda:** Quali sono i formati principali di dati usati in REST?  
**Risposta:** I formati principali sono **JSON** e **XML**, ma REST supporta anche altri formati come testo semplice o byte.

**3. Differenze tra JSON e XML?**

**Domanda:** Quali sono i vantaggi e gli svantaggi di JSON rispetto a XML?  
**Risposta:**

* **JSON:**
  + **Vantaggi:** Più leggero, più leggibile da parte degli esseri umani, meno verboso rispetto a XML.
  + **Svantaggi:** Mancanza di strumenti integrati per la validazione.
* **XML:**
  + **Vantaggi:** Possibilità di validazione tramite DTD o XSD, struttura più robusta.
  + **Svantaggi:** Più prolisso, meno efficiente in termini di dimensioni e velocità.

**4. Cos'è Postman e perché viene utilizzato?**

**Domanda:** A cosa serve Postman?  
**Risposta:** Postman è uno strumento per testare e verificare le API REST. Permette di inviare richieste HTTP (come GET, POST, PUT, DELETE) e di analizzare le risposte ricevute.

**5. Quali sono i principali codici di stato HTTP?**

**Domanda:** Elenca alcuni codici di stato HTTP utilizzati in REST e i loro significati.  
**Risposta:**

* **200 OK:** La richiesta è stata completata con successo.
* **404 Not Found:** La risorsa richiesta non è stata trovata.
* **500 Internal Server Error:** Si è verificato un errore sul server.
* **301 Moved Permanently:** La risorsa è stata spostata permanentemente.
* **400 Bad Request:** La richiesta non è valida.

**6. Qual è la differenza tra REST e SOAP?**

**Domanda:** In cosa REST è diverso da SOAP?  
**Risposta:**

* **REST:** Leggero, basato su HTTP, utilizza JSON/XML, ideale per servizi semplici.
* **SOAP:** Più complesso, utilizza protocolli personalizzati oltre a HTTP, adatto per applicazioni enterprise con requisiti di sicurezza elevati.

**7. Cos'è un CORS?**

**Domanda:** Spiega cosa sono i CORS (Cross-Origin Resource Sharing).  
**Risposta:** CORS è un meccanismo che consente a un server di specificare chi (quali domini) può accedere alle sue risorse attraverso una richiesta HTTP.

**8. Cos'è una servlet e come si integra con REST?**

**Domanda:** Qual è il ruolo delle servlet in REST?  
**Risposta:** Una servlet è una classe Java che gestisce le richieste HTTP. REST può essere implementato come servlet, che risponde ai verbi HTTP (GET, POST, ecc.) e restituisce dati come JSON o XML.

**9. Quali sono i vantaggi di REST rispetto alle tecnologie precedenti?**

**Domanda:** Perché REST è preferito rispetto ad altre tecnologie?  
**Risposta:** REST è leggero, semplice da implementare, indipendente dalla piattaforma e scalabile. Utilizza operazioni standard HTTP, rendendolo compatibile con qualsiasi sistema che supporti HTTP.

**10. Come vengono serializzati gli oggetti in REST?**

**Domanda:** Come vengono trasferiti i dati in REST?  
**Risposta:** I dati in REST sono serializzati principalmente in formato **JSON** o **XML**, rendendoli leggibili per il client.

**Ulteriori Argomenti da Studiare**

1. **Dynamic Web Project**: Creazione e configurazione di un progetto Java per il web.
2. **EAR (Enterprise Archive)**: Un file che contiene applicazioni enterprise, incluse servlet ed EJB.
3. **Intestazioni HTTP**: Cosa sono e come influenzano le risposte REST.
4. **Utilizzo di Maven per la configurazione del backend.**
5. **Esempio pratico di un'API REST con una servlet.**

Se vuoi, posso fornirti esempi pratici di codice e simulazioni di domande più complesse!

4o

**Hai detto:**

Si magari

**ChatGPT ha detto:**

ChatGPT

**Esempio Pratico di un'API REST con Servlet**

Di seguito trovi un esempio pratico di un'API REST implementata con una servlet. Questa API consente di gestire un elenco di utenti.

**1. Configurazione del Progetto**

1. Crea un **Dynamic Web Project** in Eclipse o IntelliJ.
2. Configura il file web.xml per mappare la servlet.

**2. Codice della Servlet**

**package** com.example;

**import** jakarta.servlet.ServletException;

**import** jakarta.servlet.annotation.WebServlet;

**import** jakarta.servlet.http.HttpServlet;

**import** jakarta.servlet.http.HttpServletRequest;

**import** jakarta.servlet.http.HttpServletResponse;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

// Mappatura della servlet

@WebServlet("/users")

**public** class UserServlet extends HttpServlet {

**private** List<**String**> users;

@Override

**public** **void** init() **throws** ServletException {

// Inizializza una lista di utenti

users = **new** ArrayList<>();

users.add("Alice");

users.add("Bob");

users.add("Charlie");

}

@Override

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

// Imposta il tipo di risposta

response.setContentType("application/json");

response.setCharacterEncoding("UTF-8");

// Trasforma la lista degli utenti in JSON

StringBuilder json = **new** StringBuilder("[");

**for** (int i = 0; i < users.size(); i++) {

json.append("\"").append(users.get(i)).append("\"");

**if** (i < users.size() - 1) {

json.append(",");

}

}

json.append("]");

// Invia la risposta al client

response.getWriter().write(json.toString());

}

@Override

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** IOException {

// Leggi il nuovo utente dai parametri della richiesta

**String** newUser = request.getParameter("name");

**if** (newUser != **null** && !newUser.isEmpty()) {

users.add(newUser);

// Risposta di successo

response.setStatus(HttpServletResponse.SC\_CREATED);

response.getWriter().write("{\"message\": \"User added successfully\"}");

} **else** {

// Risposta di errore

response.setStatus(HttpServletResponse.SC\_BAD\_REQUEST);

response.getWriter().write("{\"error\": \"Invalid user name\"}");

}

}

@Override

**protected** **void** doDelete(HttpServletRequest request, HttpServletResponse response) **throws** IOException {

// Leggi il nome dell'utente da eliminare

**String** userToDelete = request.getParameter("name");

**if** (users.remove(userToDelete)) {

response.getWriter().write("{\"message\": \"User deleted successfully\"}");

} **else** {

response.setStatus(HttpServletResponse.SC\_NOT\_FOUND);

response.getWriter().write("{\"error\": \"User not found\"}");

}

}

}

**3. Configurazione di web.xml**

Se non utilizzi annotazioni come @WebServlet, aggiungi la mappatura nel file web.xml:

<**web-app** xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">

<**servlet**>

<**servlet-name**>UserServlet</**servlet-name**>

<**servlet-class**>com.example.UserServlet</**servlet-class**>

</**servlet**>

<**servlet-mapping**>

<**servlet-name**>UserServlet</**servlet-name**>

<**url-pattern**>/users</**url-pattern**>

</**servlet-mapping**>

</**web-app**>

**Test dell'API con Postman**

1. **GET /users**
   * **Metodo:** GET
   * **URL:** http://localhost:8080/NomeProgetto/users
   * **Risultato:**

["Alice", "Bob", "Charlie"]

1. **POST /users**
   * **Metodo:** POST
   * **URL:** http://localhost:8080/NomeProgetto/users
   * **Body (form-data):**
     + Key: name, Value: Diana
   * **Risultato:**

{"message": "User added successfully"}

1. **DELETE /users**
   * **Metodo:** DELETE
   * **URL:** http://localhost:8080/NomeProgetto/users
   * **Body (form-data):**
     + Key: name, Value: Alice
   * **Risultato:**

{"message": "User deleted successfully"}

**Domande da Esame con Risposte**

**1. Qual è la differenza tra doGet e doPost?**

* **Domanda:** A cosa servono doGet e doPost in una servlet?
* **Risposta:**
  + doGet gestisce richieste HTTP di tipo GET, utilizzate per recuperare dati.
  + doPost gestisce richieste HTTP di tipo POST, utilizzate per inviare dati al server.

**2. Come si configura una servlet per REST?**

* **Domanda:** Quali annotazioni o configurazioni sono necessarie per creare un'API REST con servlet?
* **Risposta:** È possibile utilizzare l'annotazione @WebServlet per mappare una servlet a un URL specifico. In alternativa, si può configurare il mapping nel file web.xml.

**3. Quali sono le principali operazioni HTTP supportate da REST?**

* **Domanda:** Elenca le operazioni HTTP fondamentali di REST.
* **Risposta:**
  + **GET:** Recuperare dati.
  + **POST:** Creare nuove risorse.
  + **PUT:** Aggiornare risorse esistenti.
  + **DELETE:** Eliminare risorse.

**4. Cos'è un client REST come Postman?**

* **Domanda:** Perché si utilizza Postman?
* **Risposta:** Postman è uno strumento per inviare richieste HTTP e testare API REST, utile per verificare che il server risponda correttamente.



Cosa sono i CORS?  
  
Come formato di trasporto si utilizza Json o HTML, ad ogni modo sembra che Rest si utilizzi per serializzare degli oggetti in modo da rendere trasportabili degli oggetti,

Dynamic Web Project  
EAR  
Quando faccio l’EJB non devo creare un nuovo EAR.  
  
Il BE lo faccio diventare Maven, e dopo di ché creiamo una interfaccia remota. Il name in HelloService bisogna scriverlo, scrivendo Remote.  
  
Poi nel frontend  
  
Nel pacchetto service dobbiamo avere il nome dell’EAR e poi del BE.  
  
Package > Rest = è il backend.

[package: contoller, model,  
Package > BE = è il frontend