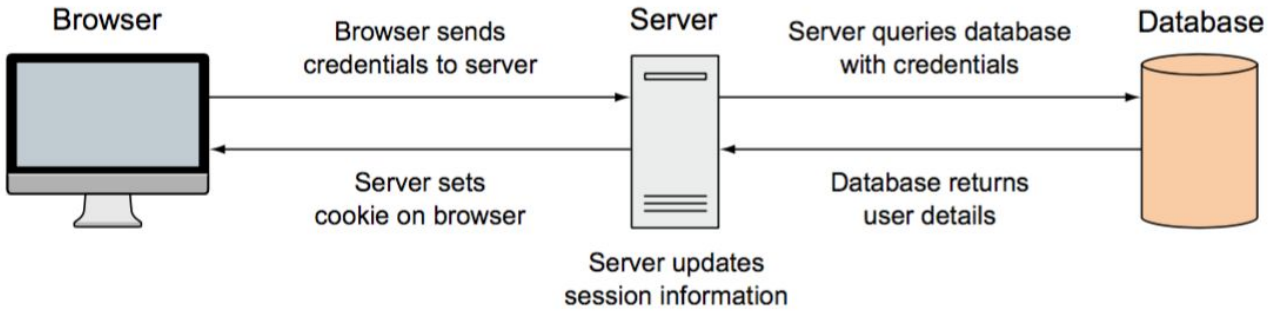
Tecnologie e applicazioni web

JSON Web Token (JWT)

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Anno accademico: 2017/2018

Cookies e le SPA

I meccanismi di autenticazione che agiscono sul protocollo HTTP (ex. cookies) sono tradizionalmente utilizzati in applicazioni web dove il contenuto, o gran parte della business logic, è gestita dal server



Cookies e le SPA

Per loro natura, i cookies sono usati per memorizzare soltanto un “identificativo” di sessione di un determinato client

- Possono essere letti e potenzialmente modificati lato client, pertanto non possono contenere informazioni sensibili che riguardano la business logic gestita dal server

Cookies e le SPA

Esempio: supponiamo di voler realizzare un'applicazione web per gestire gli appelli d'esame.

Consideriamo due tipologie di utenti:

- Professori: possono inserire appelli d'esame e visualizzare la lista degli iscritti
- Studenti: possono iscriversi ad appelli d'esame

Approccio tradizionale

Accesso da parte di uno studente:

- Uno studente inserisce le credenziali per autenticarsi con il server
- Il server verifica le credenziali con le informazioni contenute nel database
- Se le informazioni sono corrette, il server crea una “struttura dati” in memoria che riguarda il determinato client e ritorna un identificativo di sessione che solitamente viene memorizzato attraverso i cookie

Approccio tradizionale

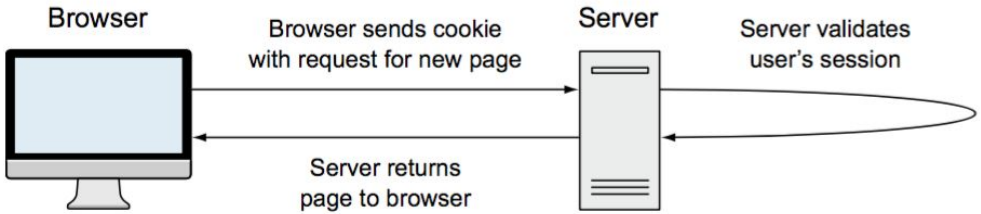
Cosa succede se uno studente vuole accedere alla funzionalità di creazione appelli?:

- Il browser dello studente richiede al server di poter accedere alla risorsa associata alla funzionalità di creazione appelli. Per farlo, invia il proprio identificativo di sessione
- Il server verifica, grazie all'identificativo, se il ruolo dell'utente è compatibile con l'azione richiesta
- In questo caso, viene ritornato un errore al client

Approccio tradizionale

Cosa succede se uno studente vuole accedere alla funzionalità di creazione appelli?

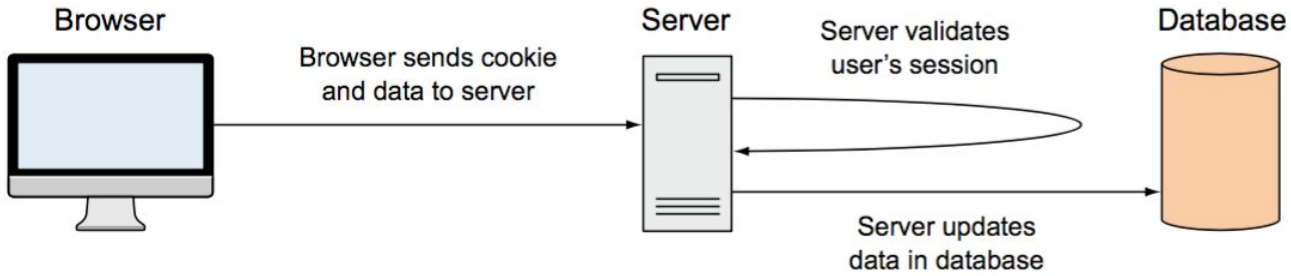
Prima di poter effettuare qualsiasi operazione, il server deve validare l'azione richiesta da parte del client (è l'unico che conosce i dati relativi all'utente)



Approccio tradizionale

Cosa succede se un professore vuole aggiungere un nuovo appello?

Anche in questo caso, prima di effettuare l'operazione, il processo di validazione della sessione deve per forza essere effettuato dal server



Approccio tradizionale

Il meccanismi necessari a rendere il protocollo HTTP stateful hanno due conseguenze negative:

- **Riducono la scalabilità**: il server deve mantenere in memoria le informazioni di sessione per ciascun utente autenticato in un dato momento

- **Aumentano l'accoppiamento tra client e server:** La business logic deve essere gestita in larga parte dal server

Approccio SPA

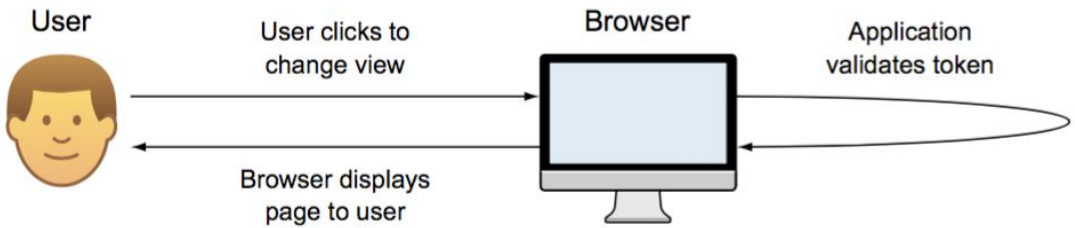
Una Single Page Application è pensata per minimizzare le interazioni con il server, che solitamente riguardano soltanto scambio di dati “fondamentali” per la sua esecuzione

Nell’esempio precedente, la SPA dovrebbe comunicare con il server soltanto per ottenere la lista appelli e/o aggiungere un nuovo appello

Approccio SPA

Cosa succede se uno studente vuole accedere alla funzionalità di creazione appelli?

L'applicazione, che gira nel browser, deve avere già tutte le informazioni per permettere o negare una determinata operazione



Tokens

Un modo efficiente per risolvere il problema è mediante l'utilizzo di tokens

- Il token è generato dal server al momento dell'autenticazione
- Il token contiene non solo un identificativo del client, ma tutte le informazioni che servono alla business logic (ex il ruolo dell'utente)
- Il token è **firmato** dal server e non può quindi essere manomesso

JSON Web Token (JWT)

Standard aperto (RFC-7519) per gestire l'autenticazione basata su token.

E' una stringa (codificata base-64) composta da 3 parti:

1. Un header che specifica il tipo di token e l'algoritmo di firma utilizzato
2. Un payload, contenente dati arbitrari in formato JSON
3. La firma digitale di header e payload

JSON Web Token (JWT)

```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpZCI6I
jEyMzQ1Njc4OTAiLCJuYW1l
IjoiRmlsaXBwbyBCZXJnYW1
hc2NvIiwicm9sZXMiOlsicH
JvZmVzc29yIl19.41ekRzJT
TbzdgBAuQKn-Jv6CV9h4NdB
9i5t0Cvjmg1g
```

1. Composto da 3 sotto-stringhe separate da “.”
2. Ciascun elemento codificato Base-64
3. Ultimo elemento è la firma digitale della sottostringa composta dai primi due

JSON Web Token (JWT)

```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpZCI6I
jEyMzQ1Njc4OTAiLCJuYW1l
IjoiRmlsaXBwbyBCZXJnYW1
hc2NvIiwicm9sZXMiOlsicH
JvZmVzc29yIl19.41ekRzJT
TbzdgBAuQKn-Jv6CV9h4NdB
9i5t0Cvjmg1g
```

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

JSON Web Token (JWT)

```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpZCI6I
jEyMzQ1Njc4OTAiLCJuYW1l
IjoiRmlsaXBwbyBCZXJnYW1
hc2NvIiwicm9sZXMiOlsicH
JvZmVzc29yIl19.41ekRzJT
TbzdgBAuQKn-Jv6CV9h4NdB
9i5t0Cvjmg1g
```

Payload

```
{
  "id": "1234567890",
  "name": "Filippo
Bergamasco",
  "roles": [
    "professor"
  ]
}
```

JSON Web Token (JWT)

```
eyJhbGciOiJIUzI1NiIsInR
5cCI6IkpXVCJ9.eyJpZCI6I
jEyMzQ1Njc4OTAiLCJuYW1l
IjoiRmlsaXBwbyBCZXJnYW1
hc2NvIiwicm9sZXMiOlsicH
JvZmVzc29yIl19.41ekRzJT
TbzdgBAuQKn-Jv6CV9h4NdB
9i5t0Cvjmg1g
```

Signature

```
HMACSHA256(
base64(header) + "."
+ base64(payload),
secretkey
)
```

JWT: Vantaggi

Il sistema di autenticazione torna ad essere stateless:

- JWT contiene non solo un “id” di autenticazione ma anche qualsiasi dato aggiuntivo utile alla business logic
- I dati **non sono cifrati** ma sono **firmati** dal generatore del token. **Non è possibile modificarli senza invalidare la firma.**

JWT: Vantaggi

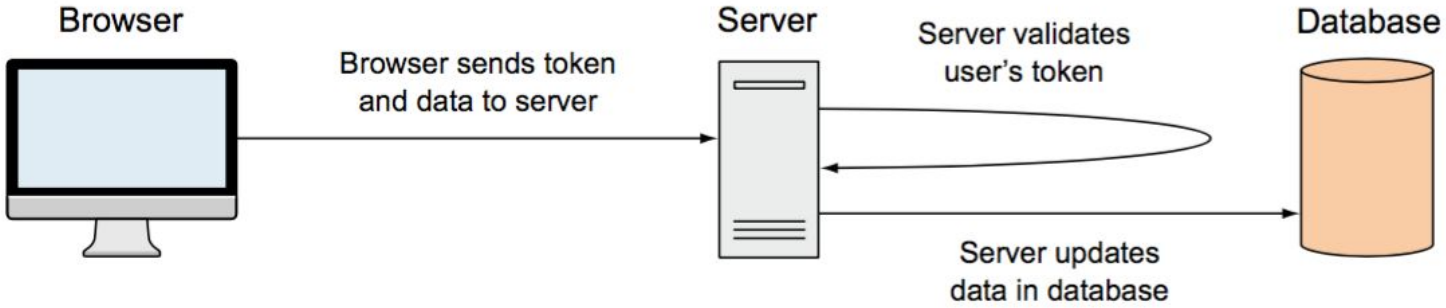
L'intero token può essere inviato al server come si trattasse di un cookie (header HTTP)

```
Authorization: Bearer <token>
```

Dato che contiene tutte le informazioni associate ad un utente, il server non deve tenere in memoria i dati relativi alla sessione

Tradeoff: banda vs. utilizzo memoria del server

JWT: Vantaggi



Il server può "recuperare" la sessione utente senza mantenerla in memoria e senza effettuare query aggiuntive al database.

Diminuisce l'accoppiamento client-server e aumenta la scalabilità

JWT: Vantaggi

Il token può essere anche scambiato tra domini differenti in contesti di single-sign-on.

- Un utente può autenticarsi su un determinato dominio A e ricevere un JWT firmato da A
- Il JWT può essere inviato al dominio B che può verificare (attraverso la firma digitale) se è stato effettivamente generato da A (di cui si fida)
- Il dominio B opera considerando l'utente come autenticato perché il processo di autenticazione è già stato gestito da A

JWT: Vantaggi

In ambito mobile, dove sono ugualmente utilizzate applicazioni web che girano sul browser e quelle native l'utilizzo dei cookie può generare problemi

- Difficile gestire i cookie da app native e non sono interscambiabili con quelli gestiti dal browser di sistema

Gli stessi JWT possono essere utilizzati indifferentemente dal browser e da app native

JSON

JavaScript Object Notation (JSON) è un formato di interscambio dati “leggero”, basato su su convenzioni comunemente usate in linguaggi come C++, Javascript, Java, etc.

Perché è così diffuso oggi giorno?

Semplice da leggere per umani e al tempo stesso facile effettuare il parsing in modo automatico

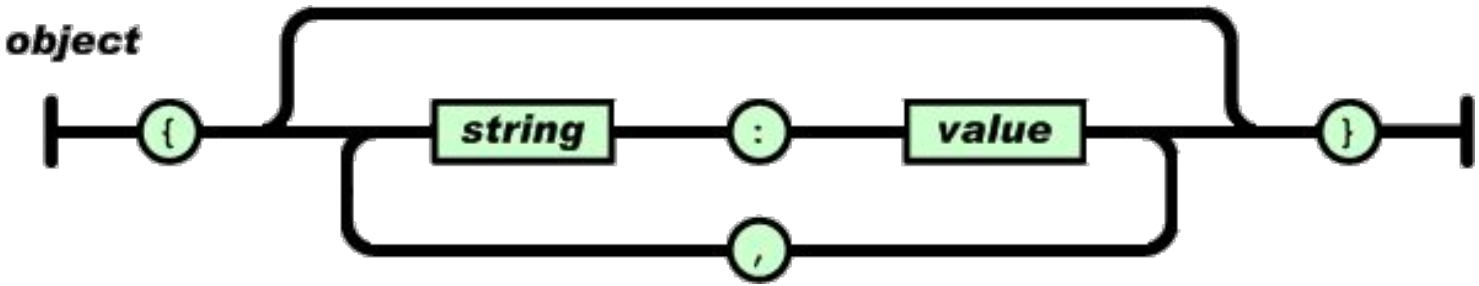
JSON

Basato sul linguaggio Javascript, anche se il tipo di strutture dati che permette di codificare sono comune alla maggior parte dei linguaggi oggi esistenti

Permette in particolare di rappresentare:

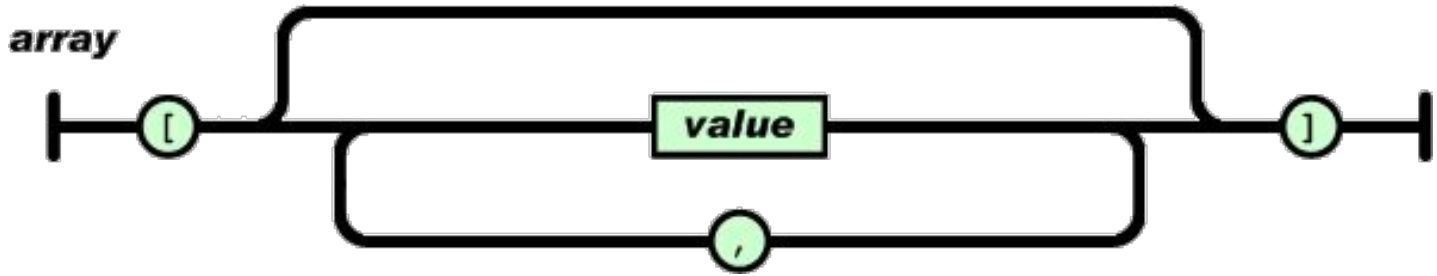
- Collezioni di coppie chiave-valore (comunemente usate per definire dizionari, oggetti, hashmap,etc)
- Liste ordinate di elementi (comunemente definiti come array, vettore, liste, etc)

JSON: Oggetti



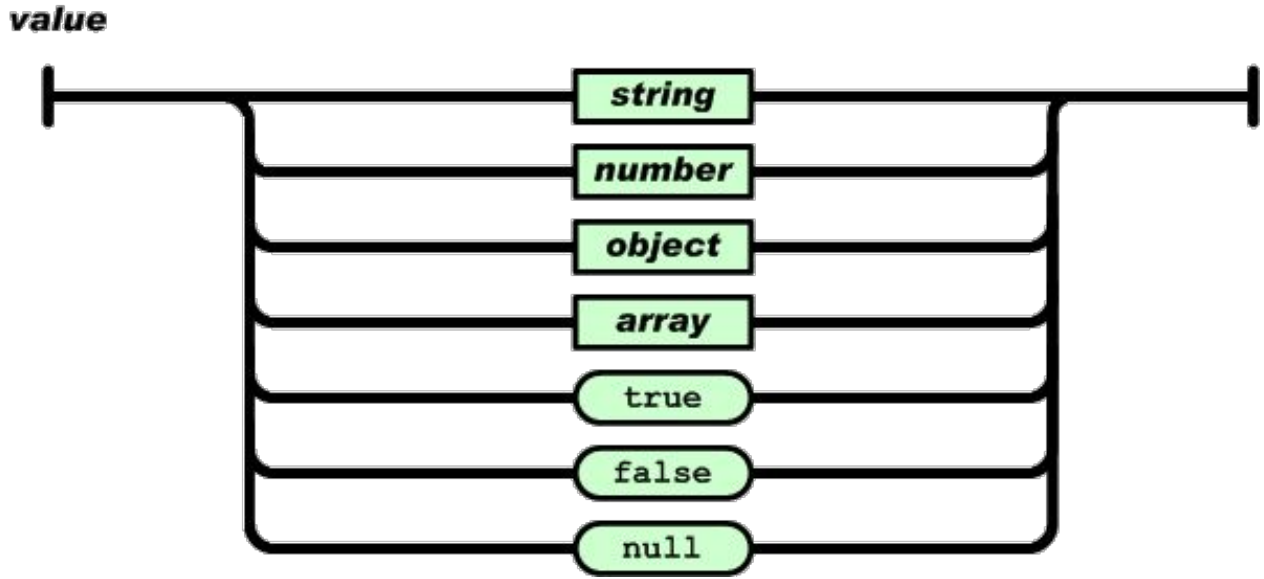
http://www.json.org/

JSON: Array



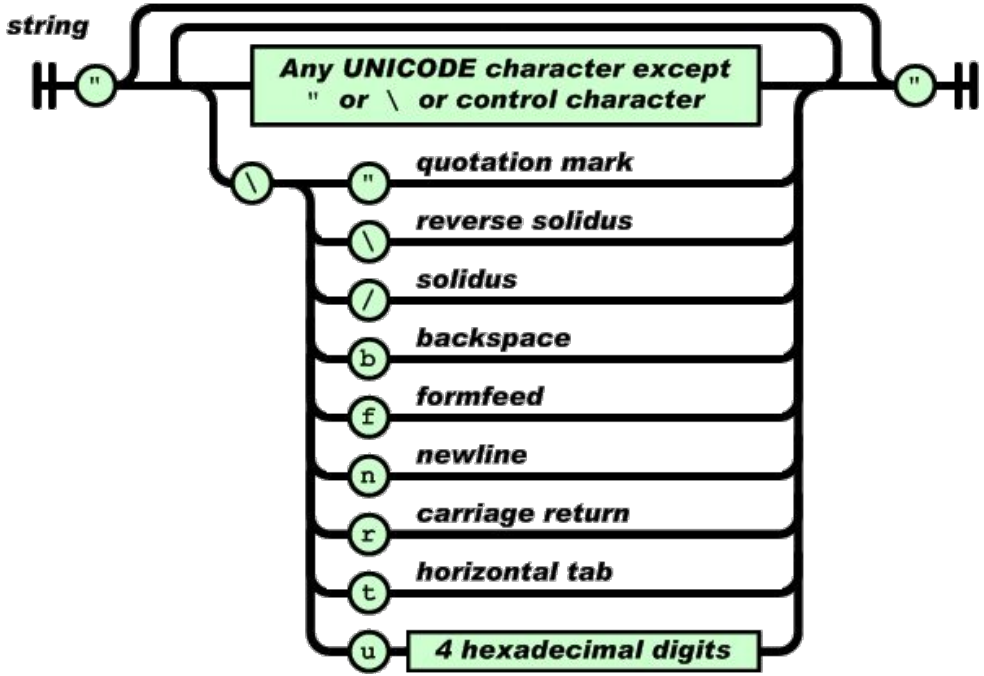
http://www.json.org/

JSON: Value



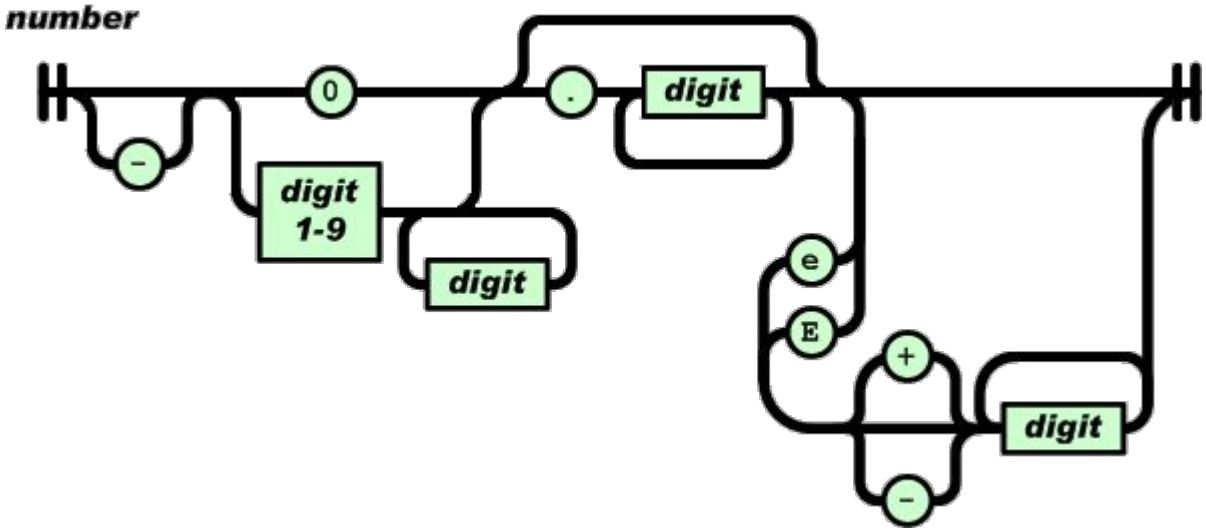
http://www.json.org/

JSON: string



http://www.json.org/

JSON: number



http://www.json.org/