**JACKSON NOTE:**

**Data Mapping: (Json)**

[**http://www.baeldung.com/jackson**](http://www.baeldung.com/jackson)

**Software support:   
  
Contact – Customer - Group**

**API JACKSON:**

1. CRUD API:
   1. Rest Crud
   2. Rest Verbs
   3. Rest dynamic fields
2. CORS
   1. Test con webapp .
3. JSONP
   1. Test con progetto Ajax
4. Formati e MessageConverters

**NOTE:**

**SPIEGAZIONI:**

**Primitivi:**

**Date:**Decidere il formato della data.

1. Parametro Controller: (String 2 Date)  
   Annotare parametro con @DateTimeFormat(pattern = "dd-MM-yyyy")
2. Serializzazione/Deserializzazione (automatica): (Bean 2 Json)
   1. API GET: Serializzazione date Jackson:
      1. Di default s in timestamp
      2. Impostare formato
         1. Annotation
         2. CustomSerializer
      3. **Formato dinamico (scelto da client con parametro rest api)**
   2. API Create: Deserializzazione (parse on Create)
      1. **Test su Create necessita Ereditarieta prima**

**Enumerations:**

**TUTORIAL (Create/Read)**

1. Create: Basic Post (simmetrico al GET details)
2. Get in formati decisi da headers
   1. MessageConverters (registrati e mappati con headers)
3. Views:
   1. EntityData e Details
4. Field Filters:
   1. Map
   2. OM
   3. Nullable fields
5. **CRUD**:
   1. UC
      * Retrieve (operazioni UI, add, remove Groups -> Group {id,name}
      * Create: POST con groupIds e CustomerId
      * Update (Group Ids), Upload Pic
      * Delete (Contact id)
6. **Hateoas: (Branch edizione 2018)**
   1. Importare dependency spring-hateoas (ultima versione)
   2. Modificare model ed aggiungere per ogni dto extend ResourceSupport (**invasivo**)
      * Invasivo perche’ l Id e’ gia nella superclasse e si riferisce al Link
   3. Verificare funzionamento linkTo e methodOn
7. AJAX:
   1. GET /POST
   2. **CORS**:
      * Sito esterno per chiamate CORS
   3. Deferred result
   4. JSONP

**RICERCA:  
  
Jersey soluzione filters dinamici:**https://github.com/jersey/jersey/tree/2.22.2/examples/entity-filtering-selectable