Explications liées au programme DemoTest6_ClientRest

@Data **@AllArgsConstructor** **@NoArgsConstructor** **public class Info {** **private String message;** **private Date date;** **}**	Classe qui sera utilisée pour passer des objets au Web service ou en provenance du Web Service

@Data **public class ListInfo implements Serializable {** **private List<Info> list;** **public ListInfo()** **{list=new ArrayList<>();}** **}**	Classe qui encapsule une ArrayList<Info> pour permettre d'échanger une ArrayList<Info> avec le Web Service Impossible, coté client, de récupérer dans du code les différents objets Info de la liste ! → on doit encapsuler cette liste dans une classe

ClientRest des différents Web Services qui produisent ou consomment des objets au format Json (voir DemoTest6)

@Component
public class ClientRest
implements CommandLineRunner {
 	
@Override
public void run(String... args) throws Exception {
 	
RestTemplate rst=new RestTemplate();
Info info=
rst.getForObject("http://localhost:8080/apijson", Info.class);
// Traitement qcq de l'objet ....
System.out.println(info.toString());
 	
ListInfo listInfo =
 	
rst.getForObject("http://localhost:8080/apijson/listinfo",ListInfo.class);
for(Info inf: listInfo.getList())
	Une classe annotée par **@Component** et qui implémente **CommandLineRunner** doit définir une méthode run() qui sera automatiquement exécutée au lancement de l'application. On n'est bien sûr pas obligé d'implémenter le Client Rest dans une telle méthode !
	
	Ce code reçoit un objet Info envoyé par un service Rest et que l'on appelle par un GET (getForObject)
	
	Ici on reçoit un objet ListInfo qui encapsule des objets Info

System.out.println(inf.toString()+" from list");
 	
ResponseEntity<Info[]> rep=
rst.getForEntity ("http://localhost:8080/apijson/tabinfo",Info[].class);
Info[] tab=rep.getBody();
for(Info inf: tab)
System.out.println(inf.toString()+" from tab");
 	
// Envoyer une collection d'objets info
ListInfo listInfo2=new ListInfo();
List<Info> list=new ArrayList<>();
list.add(new Info("Item a", new Date()));
list.add(new Info("Item b", new Date()));
list.add(new Info("Item c", new Date()));
listInfo2.setList(list);
rst.postForObject("http://localhost:8080/apijson/postlistinfo", listInfo2, ListInfo.class);
 	
}
}
	Ici on reçoit tableau Info[] . On passe par un getForEntity qui renvoie un ResponseEntity<Info[]>
	
	On extrait le tableau via getBody()
	
	Ce client envoie un objet ListInfo via un POST à un WebService qui va le consommer.
	
	Utilisation de **postForObject**