Jackson serialization Strategy verses problem area

|  |  |  |  |
| --- | --- | --- | --- |
| **Serialization Problem Area** | **Convention Only**  Require that the objects serialize and deserialize with no annotations from the framework | **Custom (De)Serializer**  Custom code that encodes/decodes member properties to/from a Jackson JSON tree document | **Annotations(MixIn Shown)**  Annotations provide hits to non-conventional behavior to the Object Mapper. Hints stored in a interface/abstract class are called MixIns |
| Read only data transfer objects | * Requires the creation of no arg constructors to create objects. May be private. * Requires the creation of setters. May be private. | * Requires maintenance of at least a Desearializer and probably a Serializer. (Large maintenance effort) | * Use the @Constructor/ @JsonCreator annotation to specify property inputs and order. |
| Derived value responses or hidden members. | * Requires that the object mapper be set to not strict mode where extraneous properties are permitted. * Side load get/set must be created. * Getters operating members must be initialized or nullary compliant. (Getter must be public) | * Requires maintenance of at least a Desearializer and probably a Serializer. (Large maintenance effort) * Best strategy for handling these hard to solve problems. | * Special private constructor with @Constructor annotation and additional properties exported. |
| Type erasure for super class lists | * Must implement extraneous property map convention. * Custom code to handle cast compatibility. | * Requires maintenance of at least a Desearializer and probably a Serializer. (Large maintenance effort) | * Use @JsonTypeInfo, @JsonTypeInfo, and @JsonSubTypes to encode class type in string and have jackson automatically process it to the proper sub class. |