**http://docs.oracle.com/javaee/6/api/javax/xml/bind/annotation/package-summary.html**

**JAXB Annotations - Contents:**

|  |  |
| --- | --- |
| **Annotation** | **Package Detail/Import statement** |
| [@XmlRootElement](http://www.techferry.com/articles/jaxb-annotations.html#XmlRootElement) | import javax.xml.bind.annotation.XmlRootElement; |
| [@XmlElement](http://www.techferry.com/articles/jaxb-annotations.html#XmlElement) | import javax.xml.bind.annotation.XmlElement; |
| [@XmlType](http://www.techferry.com/articles/jaxb-annotations.html#XmlType) | import javax.xml.bind.annotation.XmlType; |
| [@XmlTransient](http://www.techferry.com/articles/jaxb-annotations.html#XmlTransient) | import javax.xml.bind.annotation.XmlTransient; |
| [@XmlSeeAlso](http://www.techferry.com/articles/jaxb-annotations.html#XmlSeeAlso) | import javax.xml.bind.annotation.XmlSeeAlso; |
| [Using JAXB and JPA Annotations in Conjunction](http://www.techferry.com/articles/jaxb-annotations.html#Using-JAXB-JPA-Annotations-in-conjunction) | |
| [Using JAX-RS Annotations with JAXB and JPA Annotations](http://www.techferry.com/articles/jaxb-annotations.html#Using-JAX-RS-Annotations-with-JAXB-JPA-Annotations) | |

As stated earlier in [Example Application](http://www.techferry.com/articles/JEE-annotations.html#exampleApp), we are using JAXB to convert our Entities to XML or JSON format, so our rich clients like Ext-js or jQuery can easily process and present the data.

**@XmlRootElement**

Define the root element for the XML to be produced with @XmlRootElement JAXB annotation. The name of the root XML element is derived from the class name.

|  |  |
| --- | --- |
|  | @XmlRootElement  public class Contact implements Serializable {  ...  } |

You can also specify the name of the root element of the XML using its name attribute, for example @XmlRootElement(name = "CompanyContact")

**@XmlElement**

Annotate all fields that needs to be included in XML/JSON output with @XMLElement.

|  |  |  |
| --- | --- | --- |
|  | @XmlElement  public String getName() {    return name;  } | |
| http://www.techferry.com/articles/images/tips.gif | Either annotate all fields or all getter methods in your Entity bean. A mix of both is not supported. Add @XmlAccessorType(XmlAccessType.FIELD) at the class level if you want to annotate private fields instead of getter methods. |

**@XmlType**

Specify the order in which XML elements or JSON output will be produced.

|  |  |
| --- | --- |
|  | @XmlRootElement  @XmlType(propOrder = { "id", "firstName", "lastName", "email", "telephone" })  public class Contact implements Serializable {  ...  } |

The above @XmlType annotation will produce the following XML.

|  |  |
| --- | --- |
|  | <contact>      <id>38</id>      <firstname>FirstName</firstname>      <lastname>LastName</lastname>      <email>dummyEmail@techferry.com</email>      <telephone>1111111111</telephone>  </contact> |

Similarly, it will produce the following JSON.

|  |  |
| --- | --- |
|  | {"id":"38","firstName":"FirstName","lastName":"LastName",  "email":"dummyEmail@techferry.com","telephone":"1111111111"} |

**@XmlTransient**

Annotate fields that we do not want to be included in XML or JSON output with @XMLTransient.

|  |  |
| --- | --- |
|  | @XmlTransient  public Date getVersion() {    return version;  } |

**@XmlSeeAlso**

Use @XmlSeeAlso annotation when we want another Entity bean included in the XML output. In our example below, CompanyList bean refers to Company bean and the XML output should include XML generated from Company Entity too.

|  |  |
| --- | --- |
|  | @XmlRootElement(name = "List")  @XmlSeeAlso(Company.class)  public class CompanyList {      @XmlElement(name = "companyList")    public List<Company> getList() {      return list;    }  ...  } |

|  |  |
| --- | --- |
| http://www.techferry.com/articles/images/tips.gif | To include more than 1 classes, we can use @XmlSeeAlso JAXB annotation as:  @XmlSeeAlso({ A.class, B.class }) |

**Using JAXB and JPA Annotations in Conjunction**

If you have reviewed both [Hibernate - JPA Annotations](http://www.techferry.com/articles/hibernate-jpa-annotations.html" \o "Hibernate Annotations) and [JAXB Annotations](http://www.techferry.com/articles/jaxb-annotations.html), the following snippet illustrates usage of both JAXB and JPA annotations in the same entity Contact.

|  |  |
| --- | --- |
|  | @Entity  @Table(name = "CONTACT")  @XmlRootElement  @XmlType(propOrder = { "id", "firstName", "lastName", "email", "telephone" })  public class Contact implements Serializable {      @Id    @Column(name = "ID")    @GeneratedValue    private Integer id;      @Column(name = "firstName")    private String firstName;      @Column(name = "lastName")    private String lastName;      @Column(name = "EMAIL")    private String email;      @Column(name = "TELEPHONE")    private String telephone;      @Version    @Column(name = "version")    private Date version;      @ManyToOne    @JoinColumn(name = "companyId")    private Company company;      @OneToOne(mappedBy = "contact", cascade = CascadeType.ALL)    private ContactDetail contactDetail;      @XmlTransient    public Company getCompany() {      return company;    }      public void setCompany(Company company) {      this.company = company;    }      @XmlTransient    public ContactDetail getContactDetail() {      return contactDetail;    }      public void setContactDetail(ContactDetail contactDetail) {      this.contactDetail = contactDetail;    }      @XmlTransient    public Date getVersion() {      return version;    }      public void setVersion(Date version) {      this.version = version;    }      @XmlElement    public Integer getId() {      return id;    }      public void setId(Integer id) {      this.id = id;    }      @XmlElement    public String getFirstName() {      return firstName;    }      public void setFirstName(String firstName) {      this.firstName = firstName;    }      @XmlElement    public String getLastName() {      return lastName;    }      public void setLastName(String lastName) {      this.lastName = lastName;    }      @XmlElement    public String getEmail() {      return email;    }      public void setEmail(String email) {      this.email = email;    }      @XmlElement    public String getTelephone() {      return telephone;    }      public void setTelephone(String telephone) {      this.telephone = telephone;    }    } |

**Using JAX-RS Annotations with JAXB and JPA Annotations**

This section assumes that you have reviewed [RESTful JAX-RS Annotations](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html" \o "Jersey Annotations), [Hibernate - JPA Annotations](http://www.techferry.com/articles/hibernate-jpa-annotations.html) and [JAXB Annotations](http://www.techferry.com/articles/jaxb-annotations.html). Also see the section above on using [JAXB and JPA Annotations in Conjunction.](http://www.techferry.com/articles/jaxb-annotations.html#Using-JAXB-JPA-Annotations-in-conjunction)  
  
Now that you have an entity bean containing both JAXB and JPA Annotations which is capable of doing data exchange with database and coverting it to required JSON/XML format, the next step is to send this data to rich clients using jQuery or Ext-js. In your REST based web-service methods, return the Contact entity bean as shown below. Jersey, JAXB will take care of data conversion and appropriate response generation.

|  |  |
| --- | --- |
|  | @GET  @Produces("application/xml")  @Path("xml/{firstName}")  public Contact getXML(@PathParam("firstName") String firstName) {    Contact contact = contactService.findByFirstName(firstName);    return contact;  } |
|  | @GET  @Produces("application/json")  @Path("json/{firstName}")  public Contact getJSON(@PathParam("firstName") String firstName) {    Contact contact = contactService.findByFirstName(firstName);    return contact;  } |