Server-Side _Entity.java Template

```
// $ DO NOT EDIT. Make changes to
${entity.classNameWithOptionalPackage}.java instead.
#if ($entity.superclassPackageName)
package $entity.superclassPackageName;

#end
import com.webobjects.eoaccess.*;
import com.webobjects.eocontrol.*;
import com.webobjects.foundation.*;
import java.math.*;
import java.util.*;
import org.apache.log4j.Logger;

import er.extensions.eof.ERXKey;

@SuppressWarnings( { "serial",
                     "unused" })
public abstract class ${entity.prefixClassNameWithoutPackage} extends #if
($entity.partialEntitySet)er.extensions.partials.ERXPartial<${entity.parti
alEntity.className}>#elseif
($entity.parentSet)${entity.parent.classNameWithDefault}#elseif
($EOGenericRecord)${EOGenericRecord}#else
er.extensions.eof.ERXGenericRecord#end {
#if ($entity.partialEntitySet)
 public static final String ENTITY_NAME = "$entity.partialEntity.name";
#else
 public static final String ENTITY_NAME = "$entity.name";
#end

 // Attributes
#foreach ($attribute in $entity.sortedClassAttributes)
 public static final String ${attribute.uppercaseUnderscoreName}_KEY =
"$attribute.name";
 public static final ERXKey<$attribute.javaClassName>
${attribute.uppercaseUnderscoreName} = new
ERXKey<$attribute.javaClassName>(${attribute.uppercaseUnderscoreName}_KEY);
#end

 // Relationships
#foreach ($relationship in $entity.sortedClassRelationships)
 public static final String ${relationship.uppercaseUnderscoreName}_KEY =
"$relationship.name";
 public static final
ERXKey<$relationship.actualDestination.classNameWithDefault>
${relationship.uppercaseUnderscoreName} = new
ERXKey<$relationship.actualDestination.classNameWithDefault>(${relationshi
p.uppercaseUnderscoreName}_KEY);
#end

 private static Logger LOG =
```

```
Logger.getLogger(${entity.prefixClassNameWithoutPackage}.class.getName());

#if (!$entity.partialEntitySet)
 public $entity.classNameWithOptionalPackage
localInstanceIn(EOEditingContext editingContext) {
  $entity.classNameWithOptionalPackage localInstance =
($entity.classNameWithOptionalPackage)EOUtilities.localInstanceOfObject(ed
itingContext, this);
  if (localInstance == null) {
   throw new IllegalStateException("You attempted to localInstance " + this
+ ", which has not yet committed.");
  }
  return localInstance;
 }

#end
#foreach ($attribute in $entity.sortedClassAttributes)
#if (!$attribute.inherited)
#if ($attribute.userInfo.ERXConstantClassName)
 public $attribute.userInfo.ERXConstantClassName ${attribute.name}() {
  Number value =
(Number)storedValueForKey(${attribute.uppercaseUnderscoreName}_KEY);
  return ($attribute.userInfo.ERXConstantClassName)value;
 }

 public void
set${attribute.capitalizedName}($attribute.userInfo.ERXConstantClassName
${attribute.name}) {
  takeStoredValueForKey(${attribute.name},
${attribute.uppercaseUnderscoreName}_KEY);
 }
#else
 public $attribute.javaClassName ${attribute.name}() {
  /*if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {

${entity.prefixClassNameWithoutPackage}.LOG.debug("${entity.prefixClassNam
eWithoutPackage}.${attribute.name}(): " +
storedValueForKey(${attribute.uppercaseUnderscoreName}_KEY));
  }*/
  return ($attribute.javaClassName)
storedValueForKey(${attribute.uppercaseUnderscoreName}_KEY);
 }

 public void set${attribute.capitalizedName}($attribute.javaClassName
${attribute.name}) {
#if ($attribute.javaClassName == "java.math.BigDecimal")
  if (${attribute.name} != null) {
   ${attribute.name} = ${attribute.name}.setScale(${attribute.scale},
RoundingMode.HALF_UP);
  }
#end
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {
   ${entity.prefixClassNameWithoutPackage}.LOG.debug( "updating
```

```
$attribute.name from " +
storedValueForKey(${attribute.uppercaseUnderscoreName}_KEY) + " to " +
${attribute.name});
  }
  takeStoredValueForKey(${attribute.name},
${attribute.uppercaseUnderscoreName}_KEY);
 }
#end

#end
#end
#foreach ($relationship in $entity.sortedClassToOneRelationships)
#if (!$relationship.inherited)
 public $relationship.actualDestination.classNameWithDefault
${relationship.name}() {
  /*if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {

${entity.prefixClassNameWithoutPackage}.LOG.debug("${entity.prefixClassNam
eWithoutPackage}.${relationship.name}(): " +
storedValueForKey(${relationship.uppercaseUnderscoreName}_KEY));
  }*/
  return
($relationship.actualDestination.classNameWithDefault)storedValueForKey(${
relationship.uppercaseUnderscoreName}_KEY);
 }

 public void
set${relationship.capitalizedName}Relationship($relationship.actualDestina
tion.classNameWithDefault ${relationship.name}) {
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {
   ${entity.prefixClassNameWithoutPackage}.LOG.debug("updating
$relationship.name from " + ${relationship.name}() + " to " +
${relationship.name});
  }
  if (${relationship.name} == null) {
   $relationship.actualDestination.classNameWithDefault
old${relationship.capitalizedName} = ${relationship.name}();
   if (old${relationship.capitalizedName} != null) {

removeObjectFromBothSidesOfRelationshipWithKey(old${relationship.capitaliz
edName}, ${relationship.uppercaseUnderscoreName}_KEY);
   }
  } else {
   addObjectToBothSidesOfRelationshipWithKey(${relationship.name},
${relationship.uppercaseUnderscoreName}_KEY);
  }
 }

#end
#end
#foreach ($relationship in $entity.sortedClassToManyRelationships)
#if (!$relationship.inherited)
 @SuppressWarnings("unchecked")
```

```
 public NSArray<${relationship.actualDestination.classNameWithDefault}>
${relationship.name}() {
  return
(NSArray<${relationship.actualDestination.classNameWithDefault}>)storedVal
ueForKey(${relationship.uppercaseUnderscoreName}_KEY);
 }

 public Integer count${relationship.capitalizedName}() {
     return
countForRelationship(${relationship.uppercaseUnderscoreName}_KEY);
   }

 public EOQualifier qualifierFor${relationship.capitalizedName}() {
  return
qualifierForRelationshipWithKey(${relationship.uppercaseUnderscoreName}_KE
Y);
 }

#if (!$relationship.inverseRelationship || $relationship.flattened ||
!$relationship.inverseRelationship.classProperty)
 public NSArray<${relationship.actualDestination.classNameWithDefault}>
${relationship.name}(EOQualifier qualifier) {
  return ${relationship.name}(qualifier, null);
 }
#else
 public NSArray<${relationship.actualDestination.classNameWithDefault}>
${relationship.name}(EOQualifier qualifier) {
  return ${relationship.name}(qualifier, null, false);
 }

 public NSArray<${relationship.actualDestination.classNameWithDefault}>
${relationship.name}(EOQualifier qualifier, boolean fetch) {
  return ${relationship.name}(qualifier, null, fetch);
 }
#end

 @SuppressWarnings("unchecked")
 public NSArray<${relationship.actualDestination.classNameWithDefault}>
${relationship.name}(EOQualifier qualifier, NSArray<EOSortOrdering>
sortOrderings#if ($relationship.inverseRelationship &&
!$relationship.flattened &&
$relationship.inverseRelationship.classProperty), boolean fetch#end) {
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {

${entity.prefixClassNameWithoutPackage}.LOG.debug("${relationship.name}(qu
alifier " + qualifier + ", sortOrderings " + sortOrderings + "#if
($relationship.inverseRelationship && !$relationship.flattened &&
$relationship.inverseRelationship.classProperty), fetch " + fetch + "#end)
called.");
   }
  NSArray<${relationship.actualDestination.classNameWithDefault}> results;
#if ($relationship.inverseRelationship && !$relationship.flattened &&
$relationship.inverseRelationship.classProperty)
```

```
  if (fetch) {
   EOQualifier fullQualifier;
#if (${relationship.actualDestination.genericRecord})
   EOQualifier inverseQualifier = new
EOKeyValueQualifier("${relationship.inverseRelationship.name}",
EOQualifier.QualifierOperatorEqual, this);
#else
   EOQualifier inverseQualifier = new
EOKeyValueQualifier(${relationship.actualDestination.classNameWithDefault}
.${relationship.inverseRelationship.uppercaseUnderscoreName}_KEY,
EOQualifier.QualifierOperatorEqual, this);
#end

   if (qualifier == null) {
    fullQualifier = inverseQualifier;
   } else {
    NSMutableArray<EOQualifier> qualifiers = new
NSMutableArray<EOQualifier>();
    qualifiers.addObject(qualifier);
    qualifiers.addObject(inverseQualifier);
    fullQualifier = new EOAndQualifier(qualifiers);
   }

#if (${relationship.actualDestination.genericRecord})
   EOFetchSpecification fetchSpec = new
EOFetchSpecification("${relationship.actualDestination.name}", qualifier,
sortOrderings);
   fetchSpec.setIsDeep(true);
   results =
(NSArray<${relationship.actualDestination.classNameWithDefault}>)editingCo
ntext().objectsWithFetchSpecification(fetchSpec);
#else
   results =
${relationship.actualDestination.classNameWithDefault}.fetch${relationship
.actualDestination.pluralName}(editingContext(), fullQualifier,
sortOrderings);
#end
  } else {
#end
   results = ${relationship.name}();
   if (qualifier != null) {
    results =
(NSArray<${relationship.actualDestination.classNameWithDefault}>)EOQualifi
er.filteredArrayWithQualifier(results, qualifier);
   }
   if (sortOrderings != null) {
    results =
(NSArray<${relationship.actualDestination.classNameWithDefault}>)EOSortOrd
ering.sortedArrayUsingKeyOrderArray(results, sortOrderings);
   }
#if ($relationship.inverseRelationship && !$relationship.flattened &&
$relationship.inverseRelationship.classProperty)
  }
```

```
#end
  return results;
 }

 public void
addTo${relationship.capitalizedName}Relationship($relationship.actualDesti
nation.classNameWithDefault
${relationship.actualDestination.initialLowercaseClassNameWithoutPackage})
{
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {
   ${entity.prefixClassNameWithoutPackage}.LOG.debug("adding " +
${relationship.actualDestination.initialLowercaseClassNameWithoutPackage} +
" to ${relationship.name} relationship");
  }

addObjectToBothSidesOfRelationshipWithKey(${relationship.actualDestination
.initialLowercaseClassNameWithoutPackage},
${relationship.uppercaseUnderscoreName}_KEY);
 }

 public void
removeFrom${relationship.capitalizedName}Relationship($relationship.actual
Destination.classNameWithDefault
${relationship.actualDestination.initialLowercaseClassNameWithoutPackage})
{
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {
   ${entity.prefixClassNameWithoutPackage}.LOG.debug("Removing " +
${relationship.actualDestination.initialLowercaseClassNameWithoutPackage} +
" from ${relationship.name} relationship");
  }

removeObjectFromBothSidesOfRelationshipWithKey(${relationship.actualDestin
ation.initialLowercaseClassNameWithoutPackage},
${relationship.uppercaseUnderscoreName}_KEY);
#if ($relationship.ownsDestination)
  //
editingContext().deleteObject(${relationship.actualDestination.initialLowe
rcaseClassNameWithoutPackage});
#end }

 public $relationship.actualDestination.classNameWithDefault
create${relationship.capitalizedName}Relationship() {
  EOClassDescription eoClassDesc =
EOClassDescription.classDescriptionForEntityName("${relationship.actualDes
tination.name}");
  EOEnterpriseObject eo =
eoClassDesc.createInstanceWithEditingContext(editingContext(), null);
  editingContext().insertObject(eo);
  addObjectToBothSidesOfRelationshipWithKey(eo,
${relationship.uppercaseUnderscoreName}_KEY);
  return ($relationship.actualDestination.classNameWithDefault) eo;
 }
```

```
 public void
delete${relationship.capitalizedName}Relationship($relationship.actualDest
ination.classNameWithDefault
${relationship.actualDestination.initialLowercaseClassNameWithoutPackage})
{
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {
   ${entity.prefixClassNameWithoutPackage}.LOG.debug("Delete " +
${relationship.actualDestination.initialLowercaseClassNameWithoutPackage});
  }

removeObjectFromBothSidesOfRelationshipWithKey(${relationship.actualDestin
ation.initialLowercaseClassNameWithoutPackage},
${relationship.uppercaseUnderscoreName}_KEY);
 #if (true || !$relationship.ownsDestination)

editingContext().deleteObject(${relationship.actualDestination.initialLowe
rcaseClassNameWithoutPackage});
#end }

 public void deleteAll${relationship.capitalizedName}Relationships() {
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {
   ${entity.prefixClassNameWithoutPackage}.LOG.debug("Delete all " + this +
".${relationship.name}() objects");
  }
  Enumeration<${relationship.actualDestination.classNameWithDefault}>
objects = ${relationship.name}().immutableClone().objectEnumerator();
  while (objects.hasMoreElements()) {

delete${relationship.capitalizedName}Relationship(($relationship.actualDes
tination.classNameWithDefault)objects.nextElement());
  }
 }

#end
#end

#if (!$entity.abstractEntity)
 public #if (!$entity.partialEntitySet)static
#end${entity.classNameWithOptionalPackage}#if (!$entity.partialEntitySet)
create#else init#end${entity.name}(EOEditingContext editingContext
#foreach ($attribute in $entity.sortedClassAttributes)
#if (!$attribute.allowsNull)
#set ($restrictingQualifierKey = 'false')
#foreach ($qualifierKey in $entity.restrictingQualifierKeys)
#if ($attribute.name == $qualifierKey)#set ($restrictingQualifierKey =
'true')
#end
#end
#if ($restrictingQualifierKey == 'false') #if
($attribute.userInfo.ERXConstantClassName)
    , ${attribute.userInfo.ERXConstantClassName} #else
    , ${attribute.javaClassName} #end
${attribute.name}
```

```
#end
#end
#end
#foreach ($relationship in $entity.sortedClassToOneRelationships) #if
($relationship.mandatory && !$relationship.flattened &&
!($relationship.ownsDestination && $relationship.propagatesPrimaryKey))
    , ${relationship.actualDestination.classNameWithDefault}
${relationship.name}
#end
#end
     ) {
  if(LOG.isDebugEnabled()) {
   LOG.debug("${entity.classNameWithOptionalPackage}#if
(!$entity.partialEntitySet).create#else.init#end${entity.name}(EOEditingCo
ntext " + editingContext
#foreach ($attribute in $entity.sortedClassAttributes)
#if (!$attribute.allowsNull)
#set ($restrictingQualifierKey = 'false')
#foreach ($qualifierKey in $entity.restrictingQualifierKeys)
#if ($attribute.name == $qualifierKey)#set ($restrictingQualifierKey =
'true')
#end
#end
#if ($restrictingQualifierKey == 'false') #if
($attribute.userInfo.ERXConstantClassName)
    + ", ${attribute.userInfo.ERXConstantClassName} #else
    + ", ${attribute.name} #end
" + ${attribute.name}
#end
#end
#end
#foreach ($relationship in $entity.sortedClassToOneRelationships) #if
($relationship.mandatory && !$relationship.flattened &&
!($relationship.ownsDestination && $relationship.propagatesPrimaryKey))
    + ", ${relationship.name} " + ${relationship.name}
#end
#end    );
  }
   ${entity.classNameWithOptionalPackage} new${entity.name} =
(${entity.classNameWithOptionalPackage})#if
($entity.partialEntitySet)this;#else
EOUtilities.createAndInsertInstance(editingContext,
${entity.name}.ENTITY_NAME);#end

   // Set Required Attributes
#foreach ($attribute in $entity.sortedClassAttributes)
#if (!$attribute.allowsNull)
#set ($restrictingQualifierKey = 'false')
#foreach ($qualifierKey in $entity.restrictingQualifierKeys)
#if ($attribute.name == $qualifierKey)
#set ($restrictingQualifierKey = 'true')
#end
#end
```

```
#if ($restrictingQualifierKey == 'false')
  new${entity.name}.set${attribute.capitalizedName}(${attribute.name});
#end
#end
#end
  // Set Required Relationships
#foreach ($relationship in $entity.sortedClassToOneRelationships)
#if ($relationship.mandatory && !$relationship.flattened &&
!($relationship.ownsDestination && $relationship.propagatesPrimaryKey))

new${entity.name}.set${relationship.capitalizedName}Relationship(${relatio
nship.name});
#end
#end
  return new${entity.name};
 }
#end

#if (!$entity.partialEntitySet)

 public static NSArray<${entity.classNameWithOptionalPackage}>
fetchAll${entity.pluralName}(EOEditingContext editingContext) {
  return
${entity.prefixClassNameWithoutPackage}.fetchAll${entity.pluralName}(editi
ngContext, null);
 }

 public static NSArray<${entity.classNameWithOptionalPackage}>
fetchAll${entity.pluralName}(EOEditingContext editingContext,
NSArray<EOSortOrdering> sortOrderings) {
  return
${entity.prefixClassNameWithoutPackage}.fetch${entity.pluralName}(editingC
ontext, null, sortOrderings);
 }

 @SuppressWarnings("unchecked")
 public static NSArray<${entity.classNameWithOptionalPackage}>
fetch${entity.pluralName}(EOEditingContext editingContext, EOQualifier
qualifier, NSArray<EOSortOrdering> sortOrderings) {
  if(${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {

${entity.prefixClassNameWithoutPackage}.LOG.debug("fetch${entity.pluralNam
e}(editingContext " + editingContext + ", qualifier " + qualifier + ",
sortOrderings " + sortOrderings + ")");
  }
  EOFetchSpecification fetchSpec = new
EOFetchSpecification(${entity.name}.ENTITY_NAME, qualifier, sortOrderings);
  fetchSpec.setIsDeep(true);
  NSArray<${entity.classNameWithOptionalPackage}> eoObjects =
(NSArray<${entity.classNameWithOptionalPackage}>)editingContext.objectsWit
hFetchSpecification(fetchSpec);
  return eoObjects;
 }
```

```
 public static ${entity.classNameWithOptionalPackage}
fetch${entity.name}(EOEditingContext editingContext, String keyName, Object
value) {
  return
${entity.prefixClassNameWithoutPackage}.fetch${entity.name}(editingContext,
new EOKeyValueQualifier(keyName, EOQualifier.QualifierOperatorEqual,
value));
 }

 public static ${entity.classNameWithOptionalPackage}
fetch${entity.name}(EOEditingContext editingContext, EOQualifier qualifier)
{
  NSArray<${entity.classNameWithOptionalPackage}> eoObjects =
${entity.prefixClassNameWithoutPackage}.fetch${entity.pluralName}(editingC
ontext, qualifier, null);
  ${entity.classNameWithOptionalPackage} eoObject;
  int count = eoObjects.count();
  if (count == 0) {
   eoObject = null;
  }
  else if (count == 1) {
   eoObject =
(${entity.classNameWithOptionalPackage})eoObjects.objectAtIndex(0);
  }
  else {
   throw new IllegalStateException("There was more than one ${entity.name}
that matched the qualifier '" + qualifier + "'.");
  }
  return eoObject;
 }

 public static ${entity.classNameWithOptionalPackage}
fetchRequired${entity.name}(EOEditingContext editingContext, String
keyName, Object value) {
  return
${entity.prefixClassNameWithoutPackage}.fetchRequired${entity.name}(editin
gContext, new EOKeyValueQualifier(keyName,
EOQualifier.QualifierOperatorEqual, value));
 }

 public static ${entity.classNameWithOptionalPackage}
fetchRequired${entity.name}(EOEditingContext editingContext, EOQualifier
qualifier) {
  ${entity.classNameWithOptionalPackage} eoObject =
${entity.prefixClassNameWithoutPackage}.fetch${entity.name}(editingContext,
qualifier);
  if (eoObject == null) {
   throw new NoSuchElementException("There was no ${entity.name} that
matched the qualifier '" + qualifier + "'.");
  }
  return eoObject;
 }
```

```
 public static ${entity.classNameWithOptionalPackage}
localInstanceIn(EOEditingContext editingContext,
${entity.classNameWithOptionalPackage} eo) {
  ${entity.classNameWithOptionalPackage} localInstance = (eo == null) ?
null :
(${entity.classNameWithOptionalPackage})EOUtilities.localInstanceOfObject(
editingContext, eo);
  if (localInstance == null && eo != null) {
   throw new IllegalStateException("You attempted to localInstance " + eo +
", which has not yet committed.");
  }
  return localInstance;
 }
#end

#foreach ($fetchSpecification in $entity.sortedFetchSpecs)
 @SuppressWarnings("unchecked")
 public static NSArray<${entity.className}>
fetch${fetchSpecification.capitalizedName}(EOEditingContext ec#foreach
($binding in $fetchSpecification.distinctBindings),
              ${binding.javaClassName} ${binding.name}Binding#end) {
  if (${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {

${entity.prefixClassNameWithoutPackage}.LOG.debug("fetch${fetchSpecificati
on.capitalizedName}(EOEditingContext " + ec + "#foreach ($binding in
$fetchSpecification.distinctBindings), ${binding.javaClassName} " +
${binding.name}Binding + "#end)");
  }
  EOFetchSpecification spec =
EOFetchSpecification.fetchSpecificationNamed("${fetchSpecification.name}",
${entity.name}.ENTITY_NAME);
#if ($fetchSpecification.distinctBindings.size() > 0)
  NSMutableDictionary<Object, String> bindings = new
NSMutableDictionary<Object, String>();
#foreach ($binding in $fetchSpecification.distinctBindings)
  bindings.takeValueForKey(${binding.name}Binding, "${binding.name}");
#end
  spec = spec.fetchSpecificationWithQualifierBindings(bindings);
#end
  return ec.objectsWithFetchSpecification(spec);
 }

 public static ${entity.classNameWithOptionalPackage}
fetchOne${fetchSpecification.capitalizedName}(EOEditingContext ec#foreach
($binding in $fetchSpecification.distinctBindings),
                          ${binding.javaClassName}
${binding.name}Binding#end) {
  if(${entity.prefixClassNameWithoutPackage}.LOG.isDebugEnabled()) {

${entity.prefixClassNameWithoutPackage}.LOG.debug("fetchOne${fetchSpecific
ation.capitalizedName}(ec " + ec#foreach ($binding in
$fetchSpecification.distinctBindings) + ","
```

```
                        + "${binding.name}Binding " +
${binding.name}Binding#end + ")");
  }
  ${entity.classNameWithOptionalPackage} result = null;
#if ($fetchSpecification.distinctBindings.size() > 0)
  NSMutableDictionary<Object, String> bindings = new
NSMutableDictionary<Object, String>();
#foreach ($binding in $fetchSpecification.distinctBindings)
  bindings.takeValueForKey(${binding.name}Binding, "${binding.name}");
#end
#end
  try {
   result = (${entity.classNameWithOptionalPackage})
EOUtilities.objectWithFetchSpecificationAndBindings(ec,
${entity.name}.ENTITY_NAME, "${fetchSpecification.name}", #if
($fetchSpecification.distinctBindings.size() > 0) bindings #else null
#end);
  } catch ( EOObjectNotAvailableException e ) {
   //Do Nothing.
  }
  return result;
 }
```

```
#end
}
```