BusinessObjectProvider

- Allgemein
- Funktionsumfang

Allgemein

Die BusinessObjectProvider-Klasse stellt allgemeine Funktionen zur Erstellung und Weiterverarbeitung von BusinessObjekten zur Verfügung. Die folgende Liste zeigt den Funktionsumfang der Provider-Klasse und bietet Beispiele für deren Nutzung an.

Bereits in Masken dargestellte Daten werden über den Kontext in eine Regel geladen. Das parallele Laden und Weiterverarbeiten der selben Daten über den BusinessObjectProvider (update) innerhalb der Regel kann aufgrund unterschiedlicher Bearbeitungsversionen zu Fehlern führen.

Funktionsumfang

Methode	Beschreibung
@Deprecated insert neu: Interface Modifiable	Die Insert-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel ein neues Kontext-fremdes, also nicht zum aktuellen Ablauf gehörendes BusinessObjekt anzulegen, zu befüllen und in der Datenbank abzulegen. `public static <T extends BusinessObject> void insert(T type) throws BusinessException;` Ein Beispiel dazu gibt es hier.
insertAll	Die InsertAll-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel mehrere neue Kontext-fremde, also nicht zum aktuellen Ablauf gehörende BusinessObjekte anzulegen, zu befüllen und in der Datenbank abzulegen. Die übergebenen Einträge werden sequentiell abgearbeitet. `public static <T extends BusinessObject> void insertAll(Collection<T> type) throws BusinessException;` Ein Beispiel dazu gibt es hier.
@Deprecated update neu Interface Modifiable	Die Update-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel ein existierendes Kontext-fremdes, also nicht zum aktuellen Ablauf gehörendes BusinessObjekt zu verändern und in der Datenbank abzulegen. `public static <T extends BusinessObject> void update(T type) throws BusinessException;` Ein Beispiel dazu gibt es hier.
updateAll	Die Update-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel existierende Kontext-fremde, also nicht zum aktuellen Ablauf gehörende BusinessObjekte zu verändern und in der Datenbank abzulegen. Die übergebenen Einträge werden sequentiell abgearbeitet. `public static <T extends BusinessObject> void updateAll(Collection<T> type) throws BusinessException;` Ein Beispiel dazu gibt es hier.

@Deprecated delete neu Interface Modifiable	Die delete-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel ein existierendes Kontext-fremdes, also nicht zum aktuellen Ablauf gehörendes BusinessObjekt physikalisch aus der Datenbank zu löschen. `public static <T extends BusinessObject> void delete(T type) throws BusinessException;` Ein Beispiel dazu gibt es hier.
deleteAll	Die deleteAll-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel existierende Kontext-fremde, also nicht zum aktuellen Ablauf gehörende BusinessObjekte physikalisch aus der Datenbank zu löschen. Die übergebenen Einträge werden sequentiell abgearbeitet. `public static <T extends BusinessObject> void deleteAll(Collection<T> type) throws BusinessException;` Ein Beispiel dazu gibt es hier.
deleteLogical	Die deleteLogical-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel ein existierendes Kontext-fremdes, also nicht zum aktuellen Ablauf gehörendes BusinessObjekt logisch aus der Datenbank zu löschen. `public static <T extends LogicalDeletable> void deleteLogical(T type) throws BusinessException;` Ein Beispiel dazu gibt es hier.
deleteLogicalAll	Die deleteLogicalAll-Methode erlaubt es dem Regelprogrammierer während der Abarbeitung einer Regel existierende Kontext-fremde, also nicht zum aktuellen Ablauf gehörende BusinessObjekte logisch aus der Datenbank zu löschen. Die übergebenen Einträge werden sequentiell abgearbeitet. `public static <T extends LogicalDeletable> void deleteLogicalAll(Collection<T> type) throws BusinessException;` Ein Beispiel dazu gibt es hier.